

Technical Document

Getting Started with Niagara

niagara

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

This document provides basic information about the Niagara Framework® and Workbench. Included are basic descriptions and concepts as well as reference information to help Systems Integrators and Engineers get started with Niagara. These topics also provide information that is not covered in other specific Niagara 4 Guide documentation.

- [Document change log](#)
Updates to this document are listed below.
- [Related documentation](#)
The table in this topic contains related documents with a short description of each document.

Updates to this document are listed below.

May 16, 2024

- Added store size limit information to “Configuring the web browser allowlist (system.properties)” topic.
- Updated “Change of State Count” and “Elapsed Active Time” property descriptions in the “control-DiscreteTotalizerExt” component topic of the “Components in control module” chapter (as of Niagara 4.14).
- Added “Setup tab” section to “History Chart view” in the “Getting to know Workbench” chapter.

November 28, 2023

Added new property “SAML Authentication Scheme Rank” and “SAML Authentication Scheme Rank –With Tls and Without Tls” in the “Digest Authentication Scheme” topic to the “Components guides” chapter.

July 7, 2023

- Clarified instructions for adding custom fonts in the “Embedded Device Font Tool” topic.
- Updated Mode property description in “Category (baja-Category)” component reference topic.
- Edited “Palette” glossary term.

- Added “Maximum Length” property (as of Niagara 4.13) to “Password Strength” container in “bajaGlobalPasswordConfiguration” component reference topic.
- Added Https Cert (Alias, Password) and Server Certificate Health (Requested Cert, Returned Cert, Cert Status) properties to the **WebService** (“web-WebService) component” reference topic.
- Added references to JACE-9000 where appropriate.

November 15, 2022

- Added “Configuring the display of the Source Ord.”
- Added a glossary entry: handle Ord.
- Cleaned up links following migration from content management system.

October 6, 2022

- Added information about the new functionality of the “Refresh” menu item in the “About the File menu” topic.
- Added Group Nav Tree Host topic to “Getting to know Workbench” chapter.

August 25, 2022

Added new information or new topics about the following:

- Configuring settings for and viewing local Workbench logs.
- How to modify default actions.
- HTML 5 Bql Query Builder view

Added new topics to the “Plugins” chapter

- WebEditors-Multisheet
- Relation Sheet

March 8, 2022

- Added “Viewing the station log” topic to the “Key tasks” chapter.
- Added section “Using Web Wire Sheet” to “Wire Sheet object management” chapter.
- Added component “Prototype Merge Policy (baja-UserPrototypeMergePolicy)”.
- Added property description to WebService component Https Min Protocol property regarding addition of TLSv1.3 option.
- Added the following topics:
 - “Backing up a station using Workbench”

- “Restoring a controller station from a distribution file using Workbench”
- “Adding a new log category”
- “Changing the severity level of an existing log category”
- “Removing an existing log category”
- “Changing the name of a point”
- “Modifying an Action Name”
- “Configuring the web browser whitelist
- “Creating additional windows”
- “Opening a new tab”
- “Changing the time format using the Lexicon”
- “Changing the time format using Facets”
- Reorganized the document, combined topics, added missing topics, edited and rewrote multiple topics.

June 2, 2021

- Added wiresheet-WebWiresheet plugin topic.
- Added the Same Site property in the web-WebService component topic.

December 2, 2020

Updated document to Niagara 4.10 which includes the following changes:

- Added baja-SMANotificationSettings component topic.
- Added content on the Workbench **Certificate Signer Multiple Selection Tool**, available in Niagara 4.10 and later.

May 12, 2020

Updated document to Niagara 4.9 which includes the following changes:

- Added a new component Http Header Providers in web-WebService topic.
- Added new topics for workbench-WebBrowser and jxBrowser-JxWebBrowserImpl to support online help.
- Updated the “Px Editor” Workbench Options topic with added information on the “Neqlize Mode” and related properties.
- Minor changes throughout to update for Niagara 4.9. Replaced references to “applet” and “WebStart” with “Web Launcher”.

- Added a new topic for workbench-WebBrowserView plugin.

August 12, 2019

Updated document to Niagara 4.8 which includes the following changes:

- Edited the procedure, “Configuring the web browser whitelist” to reflect latest changes in functionality.
- In the section, “Customizing the Workbench environment”, added a procedure for “Configuring the web browser whitelist”.
- Edited the “wbutil-PermissionsBrowser” plugin topic, to add information on improvements to the **Permissions Browser** view
- In the section on “Workbench Tools,” added topics on the “Jar Signer” and “Module Info” tools.

October 8, 2018

In the topic, "About history extensions", added a new functionality for history extensions.

April 12, 2018

In the topic, “About the New Station Wizard”, added a note on a change in the station templates for the New Station Wizard in Niagara 4.6 and later. Also updated the Chart commands topic with information on the Ord Type and Base Ord properties.

March 5, 2018

Added the component topic, “baja-UserPrototype”.

January 25, 2018

Clarified the description of the Auto Logoff parameters in several topics.

December 11, 2017

Added SMA Notification Settings configurable properties to baja-UserService component topic.

November 13, 2017

- Update related to FIPS support available in Niagara 4.6 and later. Under the topic, “Types of Workbench Options”, replaced image and added the FIPS Options topic.
- In the Glossary, removed references to “tagged categories” from the “category” definition.

October 20, 2017:

In the section on “Customizing the Workbench environment” in the “Platform Connections options” topic, added information about Workbench enforcing the requirement to change platform defaults and the **Change Platform Defaults Wizard**.

October 12, 2017

Updates related to the Niagara 4.4 release include the following changes:

- In the section on “Customizing the Workbench environment”, added content on new types of Workbench options: added “Platform Connections Options” and “Code Signing Options,” renamed “Signing code in batch offline bogs” to read Batch signing code in offline bogs, replaced “Signing the code-signing certificate” with Signing a certificate
- In the section on “Workbench Tools,” added a topic on the “Workbench Library Service”.
- In the component topic, baja-UserService, added the Default Auto Logoff property description; and in “User Prototypes” added Auto Logoff property descriptions
- Added the component topic, “baja-SSOConfiguration” including property descriptions
- Added the component topics, “saml-SAMLAuthenticationScheme” and “saml-SAMLAttributeMapper” property descriptions
- In the component topic, “web-WebService,” added the Cache Config property description

October 2, 2017:

Code signing was implemented originally in Niagara 4.2 and is documented for the first time in Niagara 4.4. In the section, “Types of Workbench Options”, see “Signing program objects” and following topics.

August 9, 2017:

In the topic baja-UserService, added the description about “Effect of property changes on user session”

July 11, 2017

Added property descriptions in baja-UserService component topic.

June 13, 2017

Added six baja-Authentication component topics to support context-sensitive help.

March 30, 2017

Minor corrections throughout.

February 28, 2017

In the chapter, “About Workbench Tools,” updated the NDIO to NRIO Conversion Tool description with functionality changes for Niagara 4.3/AX-3.8U3. Also updated the procedure, “Converting NDIO modules to NRIO.”

December 12, 2016

- In the chapter, “About Workbench Tools”, added topics on the Embedded Device Font Tool, and Kerberos Configuration Tool.

- In the chapter, “About Workbench Tools”, updated the “New Station wizard” topic to include information on the NewSupervisorStationLinux template added in Niagara 4.2. Also, added a procedure on “Creating a new Linux Supervisor Station”.
- Removed several topics referencing the timesync module, its components and plugin view.

August 3, 2016

In the Tools chapter, described new functionality in the NDIO to NRIO Conversion Tool.

December 8, 2015

Corrections to the section on the NDIO to NRIO Conversion Tool.

November 9, 2015

- Niagara 4.1 release version.
- Updated the chapter on Workbench Tools to include information on the NDIO to NRIO Conversion Tool.
- Added Applet Module Caching Type property description in Components chapter, web-WebServices component.

August 31, 2015

Initial release document.

Parent topic: [About this Guide](#)

The table in this topic contains related documents with a short description of each document.

Related documents

Document Title	Description
<i>Niagara AX to N4 Migration Guide</i>	Describes processes and considerations for users that want to migrate NiagaraAX stations to Niagara 4.
<i>Niagara Hierarchies Guide</i>	This document provides user information for working with the Hierarchies feature.
<i>Niagara Histories Guide</i>	This document covers the concepts of histories, history services, history components and plugins, and describes common histories-related tasks.
<i>Niagara Lexicon Guide</i>	This document explains the concepts of lexicons, and describes how to use Workbench lexicon tools.
<i>Niagara Relations Guide</i>	This document explains the concepts of entity-relationships, creating, editing, and tagging relations.
<i>Niagara Scheduling Guide</i>	This document explains the concepts of scheduling and describes how to add and configure different types of schedules. Also covered descriptions about linking and importing schedules.

Document Title	Description
<i>Niagara Station Security Guide</i>	This document introduces and provides procedures for using: secure communication (TLS/SSL), email security, user authentication (AuthenticationService), and component authorizations (Categories, Roles and a bit about hierarchies and how they provide security).
<i>Niagara Tagging Guide</i>	This document describes tags, tag dictionaries, tag groups, direct and implied tags, and other concepts along with common tagging tasks.
<i>Niagara Templates Guide</i>	This document explains template concepts and includes common template tasks.
<i>Niagara Web Charts Guide</i>	This document describes and illustrates use of web charting tools.
<i>Niagara 4 Installation Guide</i>	This document describes how to install Niagara 4 on various platforms.
<i>Niagara Platform Guide</i>	Describes Workbench views that are available when you have a platform connection to a host Niagara 4 controller. It also describes PlatformServices that are available in the Niagara 4 station.
<i>Niagara Provisioning Guide</i>	This document provides concepts and procedures related to using the Niagara provisioning tools.
<i>Niagara Data Recovery Service Guide</i>	This document describes how to use the data recovery service to manage station backups.
<i>JACE-8000 Backup and Restore Guide</i>	This document describes how to make a backup and restore a backup using the USB port on the JACE-8000.
<i>JACE Niagara 4 Install and Startup Guide</i>	This document covers the initial software installation and configuration for a QNX-based JACE controller using Workbench. Applicable controllers include the JACE-3, -6, -7 series controllers.

Parent topic: [About this Guide](#)

Software frameworks provide a platform to allow businesses to more easily build their end-product offerings. The patented Niagara Framework® is targeted at solving the challenges associated with managing diverse smart devices, unifying their data, and connecting them to enterprise applications. Examples of smart devices include: monitoring and control systems, sensors, metering systems, and embedded controls on packaged equipment systems.

framework, n. something composed of parts fitted together and united; a structural frame; a basic structure (as of ideas); in object-oriented programming, a reusable basic design structure, consisting of abstract and concrete classes, that assists in building applications.

Niagara Framework, n. a universal software infrastructure that allows companies to build custom, web-enabled applications for accessing, automating, and controlling smart devices in real time over the Internet.

Using Niagara 4, integrators and developers can build device-to-enterprise solutions and Internet-enabled control and monitoring products.

The framework:

- Integrates diverse systems and devices (regardless of manufacturer or communication protocol) into a unified platform that can be easily managed in real time over the Internet (or intranet) using a late version HTML5-capable web browser.
- Supports the use of tags that can be queried, thus providing a foundation for many of the new features (search, tagging, relations, templates, hierarchies).
- Includes a cutting-edge toolset that enables non-programmers to build rich applications in a drag-and-drop environment.

Most features are designed for dual use (for programmers as well as non-programmers). These features are built around a set of Java APIs to be accessed by developers writing Java code. At the same time, most features are also designed to be used through high level graphical programming and configuration tools. This vastly increases the number of users capable of building applications on the Niagara platform.

Niagara is fully scalable, meaning that it can be run on platforms spanning the range from small, embedded devices to enterprise class servers. It is successfully applied globally in energy-services, building-automation, industrial-automation and M2M applications.

- **About the user experience**
The user experience (UX) is at the heart of Niagara 4 . Its “bajaux” framework is based on open web technologies, such as HTML5, CSS3, JSON, and Bajascript v2.0.
- **About systems capabilities**
Niagara is designed to run across a range of embedded systems, and to provide scalability to highly distributed systems.
- **Software architecture**
The framework’s architecture is organized around a set of subsystems.
- **Networking and Internet protocols**
The framework does not assume one standard network protocol, distributed architecture, or fieldbus. The design goal is to integrate cleanly with all networks and protocols. The framework standardizes what is inside the box, not what the box talks to.
- **Software building blocks**
This section describes some of the fundamental building blocks of the framework. It is important to understand these concepts and associated terminology in order to fully benefit from the use of the Workbench.

- **Formats (BFormat)**

A Baja Format (BFormat), is a class that returns object strings using a standardized formatting pattern language (script).

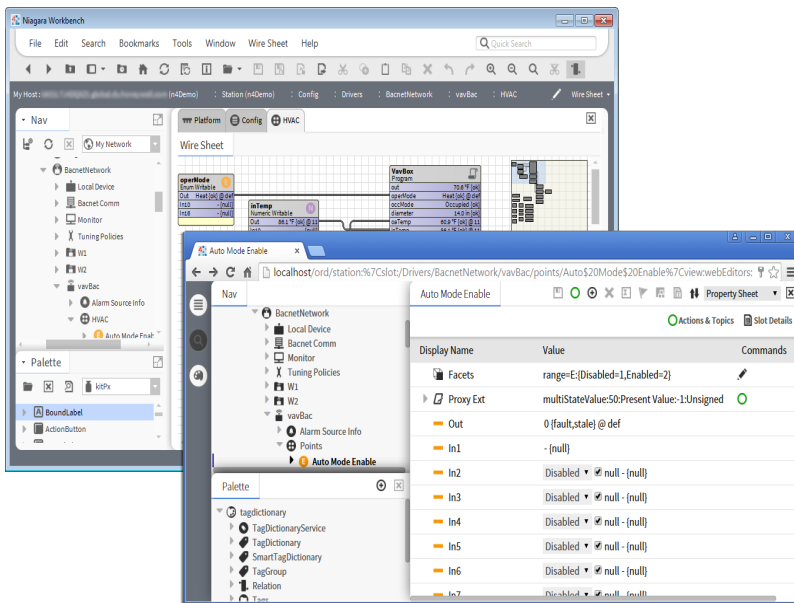
The user experience (UX) is at the heart of Niagara 4 . Its “bajaux” framework is based on open web technologies, such as HTML5, CSS3, JSON, and Bajascript v2.0.

This framework provides ease of use and efficiency, as well as the capability to produce feature-rich charts and dashboards. Niagara 4 is designed to be dynamic and responsive to accommodate changing usage circumstances and changes to individual systems over time. It offers: visualization features, tagging and templating, and advanced security.

Visualization

The HTML5-based, bajaux user interface benefits the end-user as well as provides an improved developer experience. Building owners and facility managers who typically need to locate and visualize data to drive efficiencies can use the dashboard and web chart reporting capabilities. Developers can easily create dynamic, interactive applications and views using bajaux widgets, which display across media, such as Workbench or in a modern web browser (no browser plug-in required). Search functionality integrated in Workbench enables the quick location of data to drop into other views. Optimized workflows for common tasks require few clicks and provide intuitive interaction. Additionally, the look and feel of the interface incorporates a clean and crisp design, as shown here, with a subdued and focused use of color to emphasize important information.

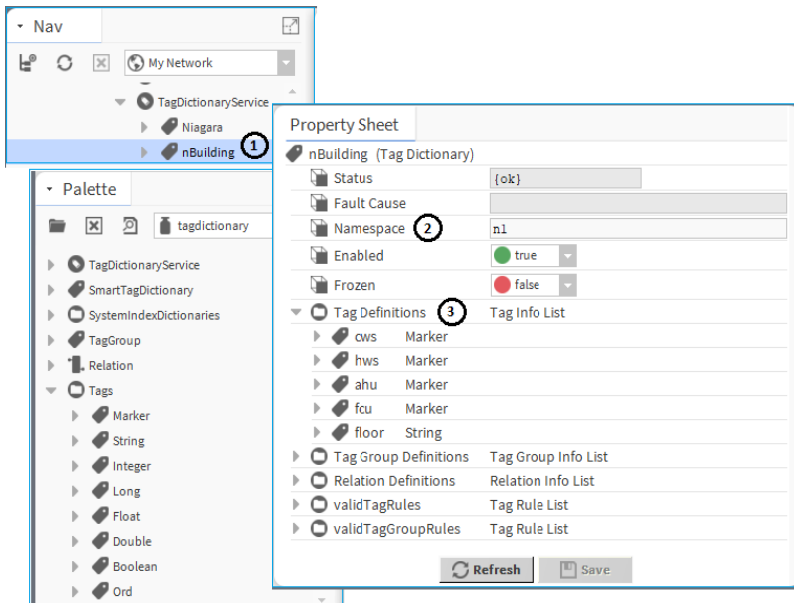
Figure 1. Workbench interface (left) compared with web browser view (right)



Tagging and templating

Systems integrators can use metadata tagging at the component level to enable multiple ways of finding data via search and navigation hierarchies, as well as ways of narrowing results via filtering. Similarly, creating templates using pre-tagged devices results in built-in reusability, which translates into shorter integration time. Using tag-based hierarchies, multiple navigation schemes can easily serve different user roles. There is no need to update Nav files every time new devices are added to a system.

Figure 2. Views that support tagging and templating



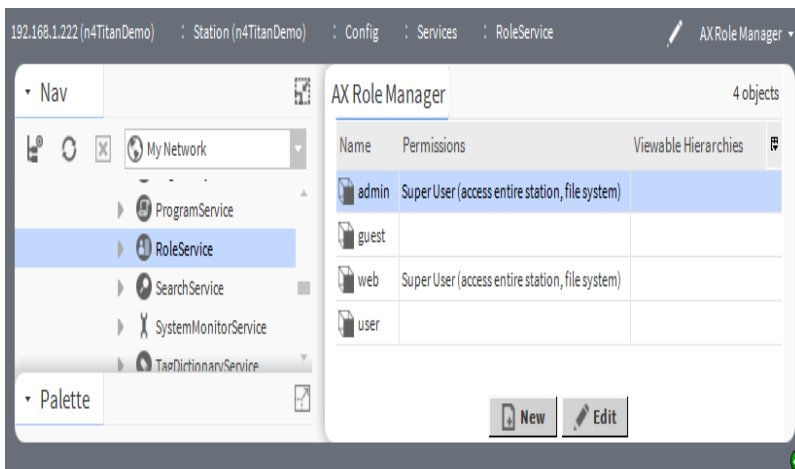
1. Shows a custom tag dictionary
2. Defines one or two characters used to group tags that share a semantic meaning for a given domain.
3. Shows the related tag definitions.

On the lower left is the tagdictionary **Palette**.

Advanced security

Developers and systems integrators can easily create secure systems using role-based access control and enhanced encryption features. Security enhancements include configurable authentication based on connection type, code signing, which verifies that modules have not been modified, and improved platform connection security. You do not have to be a cyber security expert to develop a secure system.

Figure 3. Role Manager view



Parent topic: [About the Framework](#)

Niagara is designed to run across a range of embedded systems, and to provide scalability to highly distributed systems.

About embedded systems capabilities

The framework is one of the only software stacks designed to run across the entire range of processors from small embedded devices to server class machines. It is targeted for embedded systems capable of running a Java VM. This excludes some very low-end devices that lack 32-bit processors or have only several megabytes of RAM, but opens up a wide range of processor platforms.

To speed time to market for partners designing smart devices, a reference design processor core has been developed. Known as the Niagara Processor Module (NPM), this platform can be licensed and makes it possible to quickly develop framework-based products.

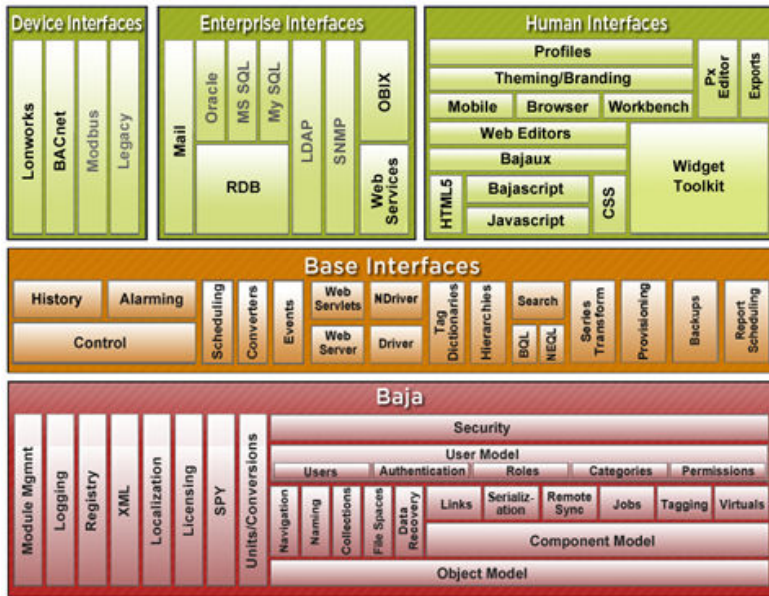
About distributed systems capabilities

The framework is designed to provide scalability to highly distributed systems composed of tens of thousands of nodes running the software. Systems of this size span a wide range of network topologies and usually communicate over unreliable Internet connections. The framework is designed to provide an infrastructure for managing systems of this scale.

Parent topic: [About the Framework](#)

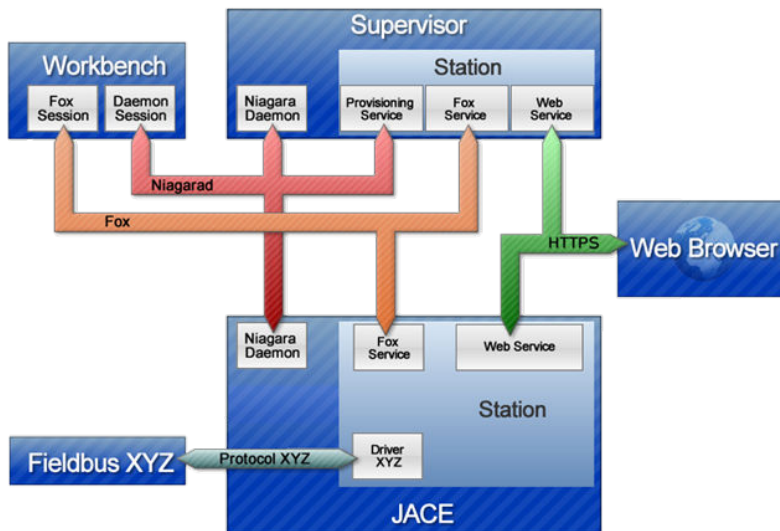
The framework's architecture is organized around a set of subsystems.

Figure 1. Niagara software subsystems



Processes serve specific functions and protocols connect the processes together.

Figure 2. Niagara software processes and protocols



- **About control systems integration**
Using the framework, control systems integration means connecting devices to a common communication medium, modeling those devices in software and programming applications to use the information in the devices.
- **About Java, virtual machines and Baja**
Much of the software is written in Java, which means that it is platform independent.
- **About APIs**
The API (Application Programming Interface) defines how software engineers access the capabilities of software like the framework. Workbench is a Niagara API. Using it you create and edit the control logic for your job site.

Parent topic: [About the Framework](#)

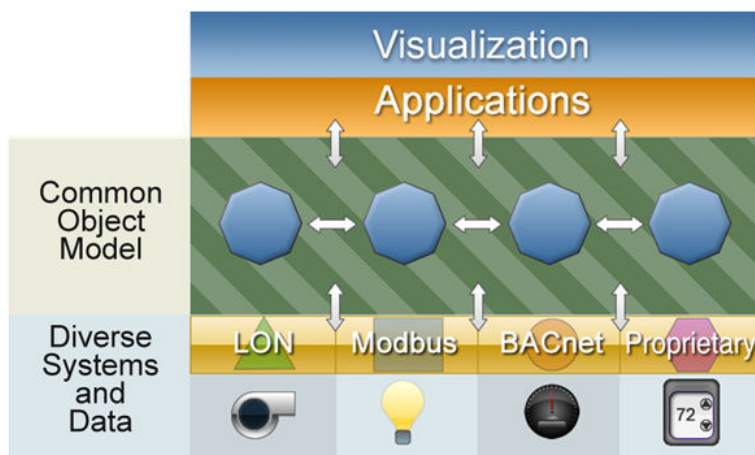
Using the framework, control systems integration means connecting devices to a common communication medium, modeling those devices in software and programming applications to use the information in the devices.

Before a device, such as a chiller, VAV box, or temperature sensor, can be used, its information must be pulled into the system database.

The framework then models devices and their data types in software through the common object model. This usually entails simplifying the device's data types to make them easier to manipulate and control through the software.

The system's common object model is then used to build applications, with the goal being to provide non-programmers a means to program the system easily without developing raw code. The framework's common object model is similar to a programming language in that there are a few key concepts that are used, but the real power is in the reusable libraries of applications and collections of objects that are available. Once you understand the key concepts and you can put them to work using the objects to build control system solutions quickly and efficiently.

Figure 1. Common object model



The framework's common object model allows the framework to:

- provide secure two-way communication between devices and the Internet
- send real-time device information across the Internet
- control devices in real-time across the Internet.

Parent topic: [Software architecture](#)

Much of the software is written in Java, which means that it is platform independent.

Prior to Java, most software was written and compiled for a particular machine or operating system. If that software needed to run on some other processor, the program had to be compiled again. Java, on the other hand, compiles once.

The software runs on embedded controllers using the QNX operating system and the IBM J9 Java Virtual Machine (JVM) and runs on Microsoft Windows desktop operating system platforms, as well as Linux and Solaris using the HotSpot JVM.

Virtual machine

It is possible to compile code once and run it on any platform due to a layer of software that exists between the machine and the software called the Java virtual machine (JVM). The framework uses the Java VM as a common runtime environment across various operating systems and hardware platforms. The core framework scales from small embedded controllers to high end servers. The framework runtime is targeted at Java 8 Standard Edition, Compact 3 profile for runtime components. The user interface toolkit and programming tools are targeted at Java 8 Standard Edition.

There are a number of different virtual machines for different platforms on which the NRE is running, but the NRE itself, and all of its modules, are the same regardless of platform. The VM is responsible for defining how the software works with a given set of hardware-how it talks to a LonWorks adapter, how it talks to the communications port, how it interacts with the operating system, among other tasks.

Baja

The Baja (Building Automation Java Architecture) core framework is designed to be published as an open standard. This standard is being developed through Sun's Java Community Process as JSR 60. This JSR is still an ongoing effort, but it is important to understand the distinction between Baja and the Niagara Framework®.

Fundamentally Baja is an open specification and the Niagara Framework is an implementation of that specification. As a specification, Baja is not a set of software, but rather purely a set of documentation.

The Baja specification will include:

- Standards for how Baja software modules are packaged
- XML schema for the component model;
- The component model and its APIs
- Historical database components and APIs
- Alarming components and APIs
- Control logic components and APIs
- Scheduling components and APIs
- BACnet driver components and APIs
- LonWorks driver components and APIs

Over time many more specifications for features will be added to Baja. But what is important to remember is that Baja is only a specification. The framework is an implementation of that specification. Furthermore you will find a vast number of features in the framework that are not

included under the Baja umbrella. In this respect the framework provides a superset of the Baja features.

Parent topic: [Software architecture](#)

The API (Application Programming Interface) defines how software engineers access the capabilities of software like the framework. Workbench is a Niagara API. Using it you create and edit the control logic for your job site.

Many features found in the framework are exposed through a set of Java APIs. In the Java world, APIs are grouped together into packages, which are scoped using DNS domain names. Software developed through the Java Community Process is usually scoped by packages starting with “java” or “javax.” It is important to understand the two types of APIs related to the framework:

- javax.baja

The APIs developed for Baja are grouped under javax.baja. These APIs are part of the open Baja specification and may be implemented by vendors other than Tridium. Using these APIs guarantees a measure of vendor neutrality and backward compatibility.

- com.tridium

Software, which is proprietary and outside of the Baja specification is grouped under the com.tridium packages. The com.tridium packages contain code specific to how the framework implements the Baja APIs. The com.tridium code may or may not be documented. If com.tridium APIs are publicly documented then the developers encourage other developers to use them, but do not guarantee backward compatibility. Undocumented com.tridium APIs should never be used by developers.

Note: Some APIs have been developed under javax.baja even though they are not currently part of the Baja specification. These APIs may eventually be published through Baja, but are currently in a development stage.

Parent topic: [Software architecture](#)

The framework does not assume one standard network protocol, distributed architecture, or fieldbus. The design goal is to integrate cleanly with all networks and protocols. The framework standardizes what is inside the box, not what the box talks to.

The software suite implements a highly efficient adaptation of the Java component software model and current Internet technologies to provide true interoperability across a wide range of automation products. The object model can be used to integrate physical devices, controllers, and primitive control applications including LonMark profiles, BACnet objects, and legacy control points. The architecture supports future enhancements by allowing legacy systems to be brought forward, where they can readily adopt new standards, solutions, and applications.

Enterprise-level software standards include Transmission Control Protocol/Internet Protocol (TCP/IP), eXtensible Markup Language (XML), Hyper Text Transfer Protocol Secure (HTTPS), Transport Layer Security (TLS), and others. These standards provide the foundation on which to build solutions that allow information to be shared between the control system and the enterprise information system.

Parent topic: [About the Framework](#)

This section describes some of the fundamental building blocks of the framework. It is important to understand these concepts and associated terminology in order to fully benefit from the use of the Workbench.

Concepts include the following:

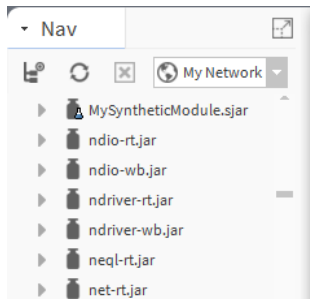
- **Modules** — the most basic unit of the software that comprises the framework.
- **Components** — the primary building blocks of the framework.
- **Presentation XML (Px)** — an XML file format that defines how the framework visualizes information (text, graphics, alarms, and so on) across diverse media, such as: Workbench, web browsers, and mobile devices.
- **Stations** — the main unit of server processing in the framework architecture. Defined by a single .bog file, a station application is launched as a single virtual machine (VM) on the host PC.
- **ORDs** — (Object Resolution Descriptor) is the framework's universal identification system and is used throughout the framework.
- **Views** — a visualization of a component, such as: Property Sheet view, Wire Sheet view, etc.
- **Lexicons** — The framework provides non-English language support by use of lexicons. Distributed as lexicon modules identified by two-digit Java locale codes, such as `fr` (French) as evidenced by this filename: `niagaraLexiconFr.jar`.
- **[About modules](#)**
Modules are the smallest units of software in the framework architecture.
- **[About stations](#)**
A station is the main unit of server processing in the framework architecture.
- **[About components](#)**
A component is a piece of self-describing software that can be assembled like building blocks to create new applications. Components are the primary building blocks that you use to engineer an application using Workbench. Component-oriented development provides many advantages for the application developer.
- **[About ORDs](#)**
The ORD (Object Resolution Descriptor) is the framework's universal identification system and is used throughout the software. The ORD unifies and standardizes access to all information. It is designed to combine different naming systems into a single string and has the advantage of being able to be parsed by a host of public APIs.
- **[About presentation](#)**
The framework provides a powerful presentation architecture based on XML and the Niagara component model.
- **[About views and HTML5 views](#)**
When you connect to your station via Workbench or via browser using the HTML5 browser-based implementation, there are many ways to visualize your system and its components.
- **[About lexicons](#)**
The framework provides non-English language support by use of lexicons. A two-digit Java locale code identifies each lexicon, such as `fr` (French) and `de` (German). All of the lexicons are distributed as modules (`niagaraLexiconXx-rt.jar`) included in the software installation.

Parent topic: [About the Framework](#)

Modules are the smallest units of software in the framework architecture.

Major releases of the software are distributed along with a set of release modules (.jar extension), but as new modules are made available for that release, they may be distributed as independent revisions within that release. Following, is a partial list of modules, as displayed in the Nav sidebar pane.

Figure 1. Module listing in the Nav tree sidebar



Note: Don't confuse modules with components. Components are used to build implementations, while modules make up the software itself.

Module characteristics

All modules are composed of a single Java Archive (.jar) file that complies with PKZIP compression. Modules contain an XML manifest; they state their dependencies on other modules and their versions.

About jar files

Module .jar files are the mechanism for distributing Java modules. A .jar file is basically a compressed package whose components can be viewed with PKZIP or other archive viewing tool. Do not attempt to unzip a .jar file. The extracted file will not work. Module .jar files are the add-ins that are plugged into the software to give additional functionality. Inside a JAR file are the compressed software; its documentation; in some cases, examples or pre-canned applications; and libraries.

There are different types of modules based on the applicable runtime profile (bajaux, doc, runtime):

- moduleName-rt.jar indicates a runtime module
- moduleName-wb.jar indicates the Workbench module
- moduleName-ux.jar indicates a bajaux web module
- moduleName-doc.jar indicates an online help module
- moduleName-se.jar indicates a Java SE compact-3 compliant module

Some modules, such as the lonworks module, are distributed as multiple JAR files because there are so many different LonWorks devices. The core protocol is packaged in a JAR file called lonWorks.jar. There are individual .jar files for each LON device manufacturer (for example, lon_mfgName.jar). Every .jar file has its own version number.

About module versions

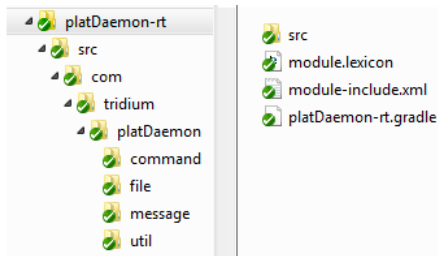
The framework uses the module version number to make sure that you have the latest available module. Versions are specified as a series of whole numbers separated by a period, for example 4.0.16. To understand which version of a module is more recent, simply observe the module version number. For example, 4.0.16 is higher (more recent) than 4.0.8 because $4.0.16 > 4.0.8$.

About directory structure

Every (.jar) module is managed in its own directory structure. The image below shows an example of a directory for the platDaemon-rt module.

All of the source code that is used to build the module's .jar file is located under the src directory. During the build process the libJar directory is used to build up the image, which is then zipped up to create the module's .jar file.

Figure 2. Directory structure used to build the module's jar



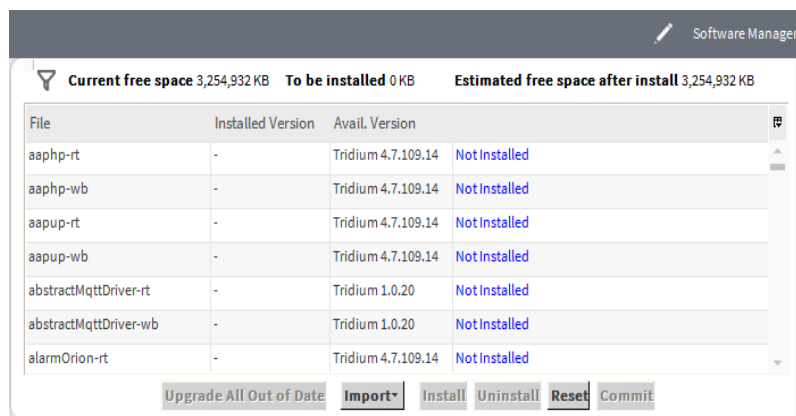
Benefits of modular software development

Modular software development provides several benefits, including the following:

- Improves tracking deployment and the assigning of version numbers

For example, if a new driver is available to be added to the framework, it can be packaged, delivered, and added as a single module or with multiple modules, using the platform **Software Manager** view. Refer to the *Niagara Platform Guide*.

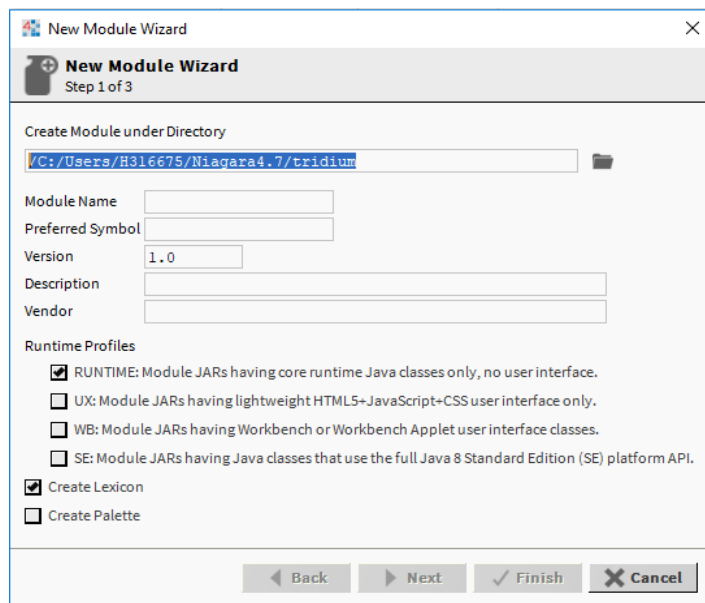
Figure 3. Software manager view



- Requires less space on the controller — the framework's modular development saves space in your controller by installing only necessary modules.

- Requires less space on the workstation — when you install the framework, you can choose the modules that you want to install with your application and omit any modules that you do not need. Later, you add modules using the platform **Software Manager** (as shown).
- Simplifies dependencies and secures files — a .jar's runtime profile describes its contents based on which systems are able to use them, or on which content is appropriate for its configuration. This simplifies handling of dependencies and enables files to be secured via digital signature.
- Creates new modules — software developers can use the **New Module** tool in Workbench to create new (.jar) modules and deploy them.

Figure 4. New module wizard



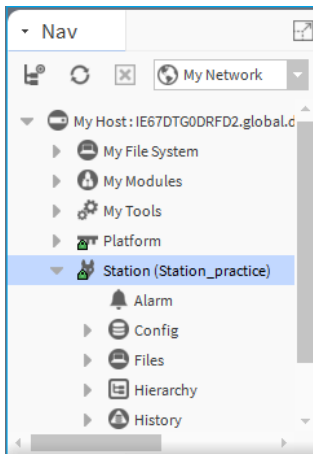
Parent topic: [Software building blocks](#)

A station is the main unit of server processing in the framework architecture.

- A station database is defined by a single .bog file, for example:
file:!stations/{name}/config.bog
- Stations are booted from their config.bog file into a single Virtual Machine (VM), or process, on the host machine.
- There is usually a one-to-one correspondence between stations and host machines (Supervisors or controllers). However it is possible to run two stations on the same machine if they are configured to use different IP ports.

A station runs the components of the framework and provides access for client browsers to view and control these components. The primary parts of a station include components and services. A station combines a database, a web server, and a control engine. The station either runs on a Web Supervisor PC or the controller.

Figure 1. Station in the Nav tree

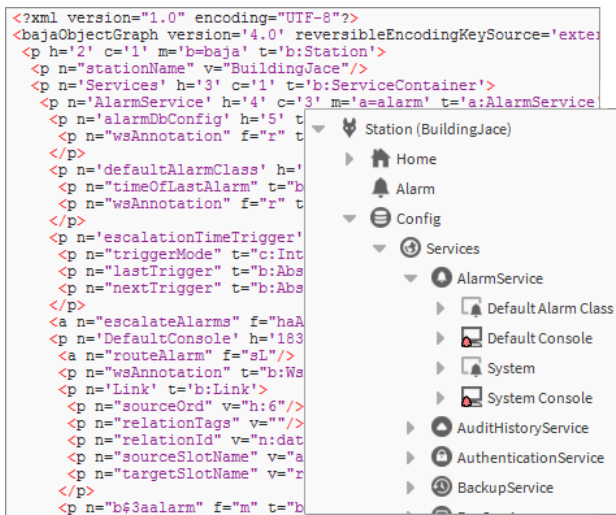


A system can be a single station or multiple stations depending on the size of the project and it is defined by a bog file.

About BOG files

A bog file (Baja object graph) contains components. It can be a complete database or any collection of components. A bog file is a special file that describes the components in a database. All views can be used on components in a bog file just as if they were in a station.

Figure 2. Sample bog file and Nav tree



Parent topic: [Software building blocks](#)

A component is a piece of self-describing software that can be assembled like building blocks to create new applications. Components are the primary building blocks that you use to engineer an application using Workbench. Component-oriented development provides many advantages for the application developer.

Components differ from modules in that components comprise an implementation of the framework, whereas modules comprise the software itself. A component-centric architecture provides the following:

- Data normalization

Components provide a model used to normalize the data and features of different types of protocols and networks so that they can be integrated seamlessly.

- Graphic development tools

Applications can be assembled with components using graphical tools in the Workbench. This allows new applications to be built without requiring a Java developer.

- Application visibility

Components provide unsurpassed visibility into applications. Since components are self-describing, it is very easy for tools to look at how an application is assembled, configured, and to determine what is occurring at any point in time. This provides great value in debugging and maintaining applications.

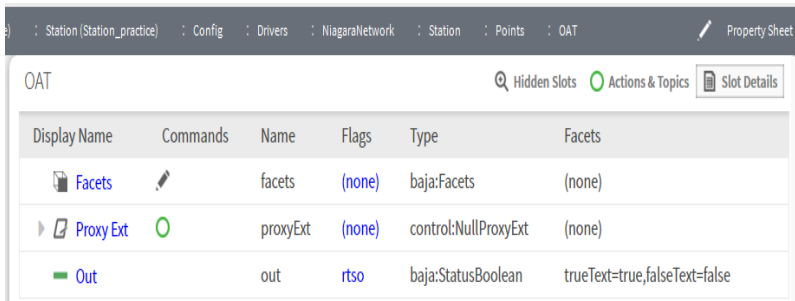
- Software reuse

Components enable software reuse. The framework supports custom development and extension. Its components are extensible and go beyond data and protocols to unify the entire development environment.

About slots

Framework components are defined as a collection of slots. You can see all the slots that make up a component by viewing its slot details on either the **Property Sheet** view or **AX Slot Sheet** view.

Figure 1. Slot details for a single component



Display Name	Commands	Name	Flags	Type	Facets
Facets		facets	(none)	baja:Facets	(none)
ProxyExt		proxyExt	(none)	control:NullProxyExt	(none)
Out		out	rtso	baja:StatusBoolean	trueText=true,falseText=false

- Slot type — there are three types of slots:

- Property — property slots represent a storage location of another object.
- Action — an action is a slot that specifies behavior that may be invoked either through a user command or by an event. Actions provide the capability to provide direction to components. They may be issued manually by the operator or automatically through links. Actions can be invoked in the **Property Sheet** view or by right-clicking on the component in the Nav tree.
- Topic — topics represent the subject of an event. Topics contain neither a storage location, nor a behavior. Rather a topic serves as a place holder for an event source.

- Slot name — every slot is identified by a slot name that is unique within its type. Slot names must contain ASCII letters or numbers.
- Slot definition — slots are either frozen or dynamic. A frozen slot is defined at compile time within a Type's Java class. That means that frozen slots are consistent across all instances of a specified Type – they don't change. Dynamic slots may be added, removed, renamed, and reordered during runtime – they can change. The power of the framework is in providing a consistent model for both frozen (compile time) slots and dynamic (runtime) slots.
- Flags — slots have flags that allow modification of an object's presentation or behavior. For example, "read-only", "operator allowed", and "hidden", are some of the slot flags that may be used to restrict the presentation or behavior of an object.
- Facets — facets contain metadata about an object. For example, "units of measurement" is a type of facet. Facets may be viewed in the slot sheet and edited from a component property sheet.

About master/slave components

Master components can be defined so that persistent properties are copied to slave components when they are changed. This allows you to change a property in one component that automatically updates the components that are linked to it as slaves. In this way a master component can update slave components in all the other stations in a system.

About point components

In any station, all real-time data are normalized within the station database as points, a special group of components. The following image shows several types of control points, as listed in the control module palette in Workbench.

Each type of point may be used for different purposes. When you engineer a job you may want to name a point, for example: a NumericWritable point named CondSetpoint. Points may be named and renamed but they retain their initial point type characteristics and their characteristic icon color.

Points serve as a type of shell, to which you may add point extensions. These extensions allow you to select only those functions that you need and thereby limit your point properties to just those that are necessary for your current application.

About component naming

In a station, components should be properly named using the following set of rules:

- Only alphanumeric (A-Z, a-z, 0-9) and underscore (_) characters are used. Spaces, hyphens, or other symbols characters (for example, %, &, ., #, and so on) are illegal in component names. using ASCII code in component names.
- The first character in the name must be a letter (not a numeral).
- The name must be unique for every component within the same parent component. Workbench automatically enforces this rule, via a popup error message.
- Naming is case-sensitive—for example, zone21 and Zone21 are unique names.

Note: Case differences among names affect name sorts in table-based views, which order by ASCII code sequence, that is capital letters (A-Z) first, lower case (a-z) following.

To convey multiple-word names without using spaces, naming conventions, such as CamelCase and/or underscores are often used. For example:

- Floor1 or Floor_1
- ReturnAirTemp or Return_Air_Temp
- Zone201_SAT or Zone_201_SAT

Workbench allows you to name components improperly, such as with spaces or other non-alphanumeric characters, without any warning.

Various drivers have learn features to automate the creation of points, some of which (by default) may also have improper names—reflective of the native name of the source object. For example, a BACnet proxy point might have the default name Zone 6 RH%, which matches the actual (native) BACnet object's name. Be aware that the actual component name has all illegal characters escaped using a \$ character, along with the ASCII code for that character, in hexadecimal. The proxy point mentioned above, for example, results in the name Zone\$206RH\$25, where the \$20 escapes the space and the \$25 escapes the %. You can see these escaped names in the slot sheet of the component's parent container. Or, with the component selected, look at its ord (shortcut **Ctrl + L**) to see its actual name. Other examples include the dash character "-" which is "\$2d" and anytime you begin a name with a number, the "\$3" is appended to the front of the name.

For the most part, this escaped name scheme is transparent to users. Whenever the name is displayed to the user, say in the Nav side bar, **Property Sheet**, **Wire Sheet**, or a **Point Manger**, the component's name is unescaped by replacing the code (say, \$20) with the actual ASCII character (say, a space). This way, the user sees Zone 6 RH% and so on. This is the component's display name.

In some cases, escaped names lead to confusion. You should avoid them if possible (rename them). For example, if you add history extensions to escaped-named points, you see those escape codes listed for source points when accessing the **History Ext Manager** (although associated histories use the display names). Or, if you are building Px pages and manually typing in ORDs in Px widgets, you probably know source points by display names only. If you manually type in an ORD without the actual (escaped) name, the widget binding fails with an error.

Note: If this sounds too complicated, remember that drag-and-drop operations resolve escaped names without problems—for example, drag any point onto a Px page to get its proper ORD.

About palettes

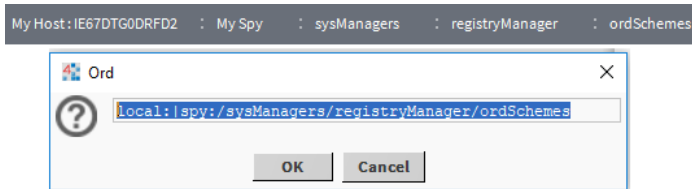
The palette provides a hierarchical view of available components. You may copy a component from the palette and paste it where you need it — on a **Wire Sheet**, **Property Sheet**, **Px View**, or in the Nav side bar pane. You can also create custom palettes that you associate with a module and use to hold a collection of frequently used components.

Parent topic: [Software building blocks](#)

The ORD (Object Resolution Descriptor) is the framework's universal identification system and is used throughout the software. The ORD unifies and standardizes access to all information. It is designed to combine different naming systems into a single string and has the advantage of being able to be parsed by a host of public APIs.

An ORD is comprised of one or more queries where each query has a scheme that identifies how to parse and resolve to an object. ORDs may be displayed visually, as with the Open Ord locator or they may be entered in a text field, as shown in the **Open ORD** window.

Figure 1. Open ORD and graphic locator system



ORDs can be relative or absolute. A relative ORD takes the format of `slot:...`, such as `slot:AHU1/Points/SpaceTemp`. The ORD is relative to the base ORD that contains `slot:AHU1`. An absolute ORD usually takes the general format of `host|session|space`, as illustrated below.

Figure 2. Absolute ORD typical structure

ORD:		
host	session	space
- ip:hostname - dialup	- fox:port - platform:	- station - file - history - view ...

- host — identifies a machine usually by an IP address such as `ip:hostname`. For example `fox:` indicates a fox session to the host.
- session — identifies a protocol being used to communicate with the host.
- space — identifies a particular type of object. Common spaces are `module:`, `file:`, `station:`, `view:`, `spy:`, and `history:`

The local VM is a special case identified by `local:`, which always resolves to `BLocalHost.INSTANCE`. The local host is both a host and a session (since no communication protocols are required for access).

Both a slot path and a handle scheme can name components within a ComponentSpace. So the ORD for a component usually involves both a space query and a path/handle.

ORD examples

- `ip:somehost|fox:|station:|slot:/MyService`
- `ip:somehost|fox:|station:|h:/42`

- `ip:somehost|fox:|file:/C:/dir/file.txt`
- `local:|file:!jre/lib/logging.properties`
- `local:|module://icons-ux/x16/cloud.png`
- `local:|spy:/`

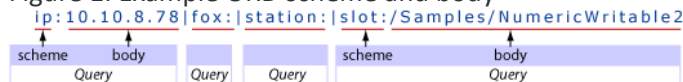
In the framework, you may view the complete list of installed ORD schemes at `spy:/sysManagers/registryManager/ordSchemes(local:|fox:|spy:/sysManagers/registryManager/ordSchemes)`.

- **About schemes**
An ORD is a list of one or more queries separated by the pipe symbol (|). Each query is an ASCII string formatted as `<scheme>:<body>`.
- **Types of schemes**
A scheme is a globally unique identifier which specifies how to find a piece of framework code with which to lookup an object.
- **Types of space**
Space defines a group of objects that share common strategies for loading, caching, lifecycle, naming, and navigation. Following, is a list of some of the different types of space:
- **Types of files**
In the file system, you may create and edit various types of files. Following, is a list of some of the different types of files that reside in the file space:
- **About file naming**
When working in a station, you often create different types of files, for example, a Px file if adding a new view, if creating a new Chart file, whenever exporting a view to pdf/txt/csv file, or if making a station backup .dist file. In addition, you often create new station file folders, and also copy graphic image files over to the station's file space.
- **Configuring the display of the Source Ord**
The Source Ord for a component may display the slot path or the handle Ord. The slot path associated with the component may change if you change the location of the component in the station. The handle Ord is a unique, short ID that does not change. This ID functions primarily in the background, however, you may configure the Source Ord to display it instead of the slot path to a component.

Parent topic: [Software building blocks](#)

An ORD is a list of one or more queries separated by the pipe symbol (|). Each query is an ASCII string formatted as `<scheme>:<body>`.

Figure 1. Example ORD scheme and body



- `<scheme>` name is a globally unique identifier which specifies how to find a piece of code to lookup an object from the body string.
- `<body>` string is formatted differently, according to the requirements of the scheme. The only rule is that it cannot contain a pipe symbol. Queries can be piped together to let each scheme focus on how to look up a specific type of object. In general, absolute ORDs are in the following format: `host | session | space`.

Some examples follow:

– `ip:somehost|fox:|file:/dir/somefile.txt`

In this example, the `ip` scheme identifies a host machine. The `fox` scheme specifies a session to that machine usually on a specific IP port number. Finally, the `file` scheme identifies an instance of a file within the `somehost` file system.

– `ip:somehost|fox:1912|station:|slot:/Graphics/Home`

In this example, the `ip` scheme identifies a host machine using an IP address. The `fox` scheme specifies a session to that machine usually on a specific IP port number. Finally, the `station` and `slot` schemes identify a specific component in the station database.

– `local:|module://icons/x16/cut.png`

This example illustrates a special case. The scheme `local`, which always resolves to `BLocalHost.INSTANCE` is both a host scheme and a session scheme. It represents objects found within the local VM.

Parent topic: [About ORDs](#)

A scheme is a globally unique identifier which specifies how to find a piece of framework code with which to lookup an object.

- `ip` identifies an `Ip Host` instance. ORDs starting with `ip` are always absolute and ignore any base that may be specified. The body of an `ip` query is a DNS hostname or an IP address of the format: `dd.dd.dd.dd`.
- `fox` establishes a `Fox` session. `Fox` is the primary protocol used by the framework for IP communication. A `fox` query is formatted as `fox:` or `fox:<port>`. If the port is unspecified, the framework assumes the default port, 1911.
- `file` identifies files on the file system. All file ORDs resolve to instances of `javax.baja.file.BIFile`. File queries always parse into a `FilePath`. File ORDs include the following examples:

– Authority Absolute: `//hostname/dir1/dir2`

– Local Absolute: `/dir1/dir2`

– Sys Absolute: `!defaults/system.properties`

Sys absolute paths indicate files rooted under the framework's installation directory identified via `Sys.getBajaHome()`.

– User Absolute `^config.bog` is rooted under the user home directory identified via `Sys.getUserHome()`. In the case of station VMs, user home is the directory of the station database.

User absolute paths

– Relative: `myfile.txt`

- Relative with Backup: ../myfile.txt
 - **module** accesses BFiles inside the module jar files. The module scheme uses the file: scheme's formatting where the authority name is the module name. Module queries can be relative. If the query is local absolute then it is assumed to be relative to the current module. Module queries always parse into a FilePath:
 - module://icons/x16/file.png
 - module://baja/javax/baja/sys/BObject.bajadoc
 - module:/doc/index.html
 - **station** resolves the BComponentSpace of a station database.
 - **slot** resolves a BValue within a BComplex by walking down a path of slot names. Slot queries always parse into a SlotPath.
 - **h** resolves a BComponent by its handle. Handles are unique String identifiers for BComponents within a BComponentSpace. Handles provide a way to persistently identify a component independent of any renames which modify a component's slot path.
 - **service** resolves a BComponent by its service type. The body of the query should be a type spec.
 - **spy** navigates spy pages. The javax.baja.spy APIs provide a framework for making diagnostics information easily available.
 - **bql** encapsulates a BQL query.
 - **nspace** provides a global resolution of objects.
 - Local station example- nspace:SupervisorName|slot:|/a/b/c
 - Remote station example- nspace:DeviceName|slot:/a/b/c. This ORD resolves on a Supervisor to a virtual,
 example- station:|slot:/Drivers/NiagaraNetwork/DeviceName/virtual|virtual:/a/b/c
- nspace provides a global resolution of objects.
- Local station example- nspace:SupervisorName|slot:|/a/b/c
 - Remote station example- nspace:ControllerName|slot:/a/b/c. This ORD resolves on a Supervisor to the framework virtual.
 example- station:|slot:/Drivers/NiagaraNetwork/ControllerName/virtual|virtual:/a/b/c
- **sys** queries the systemDb.

Note: It supports NEQL only.

- For example, to query the entire systemDb for all devices in the system, enter:
ip: <SupervisorIP> | foxs: | sys: | neql: n: device

where <SupervisorIP> is the IP address of the Supervisor PC.

- Another example: to issue a scoped query against the systemDb for ModbusTCP devices in subordinate station, enter: ip: <SupervisorIP> | foxs: | station: | slot: / Drivers / ModbusTCPNetwork | sys: | neql: n: device

where <SupervisorIP> is the IP address of the Supervisor PC.

- **single** is commonly piped at the end of a NEQL query (or certain BQL queries) to resolve the first entity in the result set. The following example runs a NEQL query to find all components tagged with “hs:ahu”, and then resolves the first one in the result set.

ip: <SupervisorIP> | foxs: | station: | slot: / | neql: hs: ahu | single

where <SupervisorIP> is the IP address of the Supervisor PC.

Parent topic: [About ORDs](#)

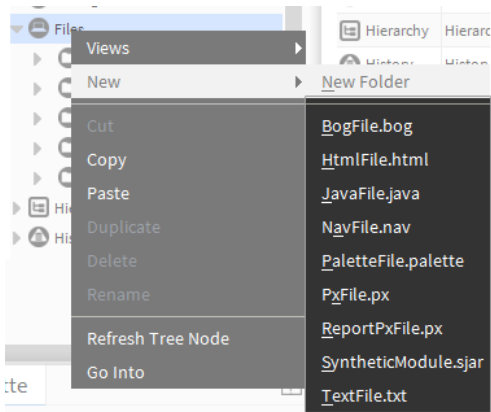
Space defines a group of objects that share common strategies for loading, caching, lifecycle, naming, and navigation. Following, is a list of some of the different types of space:

- component
- file
- hierarchy
- history
- module
- orion
- station
- view

Parent topic: [About ORDs](#)

In the file system, you may create and edit various types of files. Following, is a list of some of the different types of files that reside in the file space:

Figure 1. Types of files available from the New menu



- **BogFile.bog** is a database file.
- **HtmlFile.html** is edited in the **Text File Editor** view and viewed in the **Html View**.
- **JavaFile.java** is edited in the **Text File Editor** view.
- **NavFile.nav** is edited in the **Nav File Editor** and viewed in the Nav tree.
- **PaletteFile.palette** is a custom collection of components that you create and save for viewing in the **Palette** sidebar.
- **PxFile.px** is edited in the Px view and used to store graphic presentations that are available for viewing in the Px viewer and in a browser.
- **ReportPxFile.px** is a regular Px file that has been pre-loaded with a ReportPane widget. This reporting functionality helps you design, display, and deliver data to online views, web browser, and various other formats.
- **TextFile.txt** is edited and viewed in the **Text File Editor**.

Parent topic: [About ORDs](#)

When working in a station, you often create different types of files, for example, a Px file if adding a new view, if creating a new Chart file, whenever exporting a view to pdf/txt/csv file, or if making a station backup .dist file. In addition, you often create new station file folders, and also copy graphic image files over to the station's file space.

Regardless of file types, whenever saving files (or before copying files to the station), you restrict all characters in file and folder names to ones in the original set of ASCII characters in the BNF (Backus-Naur Form notation) for framework file paths, namely:

a-z | A-Z | 0-9 | specials

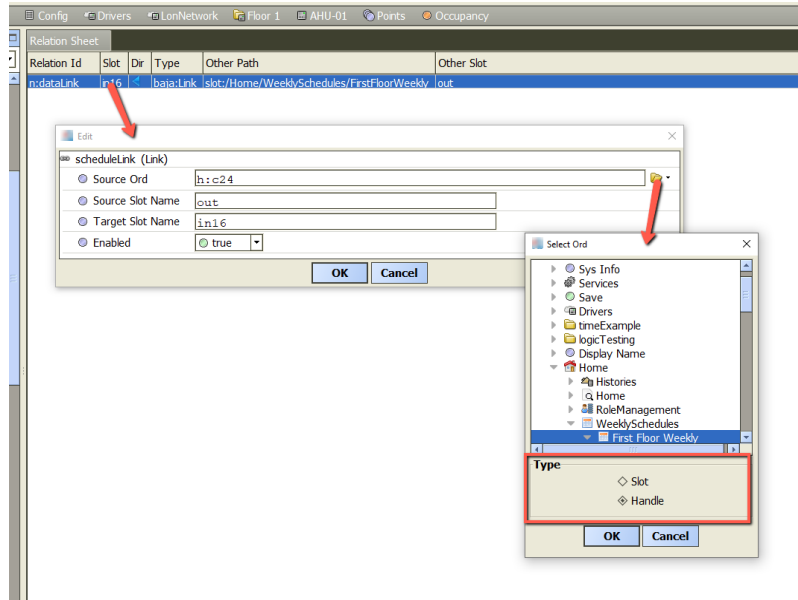
specials= space | . | : | - | _ | \$ | + | (|) | & | ' | ' | @ | [|]

Otherwise, use of other characters, for example, a tilde (~) in file or folder names may result in obscure problems, particularly on controller platforms (QNX-based hosts).

Parent topic: [About ORDs](#)

The Source Ord for a component may display the slot path or the handle Ord. The slot path associated with the component may change if you change the location of the component in the station. The handle Ord is a unique, short ID that does not change. This ID functions primarily in the background, however, you may configure the Source Ord to display it instead of the slot path to a component.

1. Open a **Relation Sheet** for the component.



In this **scheduleLink** example, the **Relation Sheet** shows the data link and displays the schedule component's slot path Ord under the Other Path column as `slot : /Home/WeeklySchedule/FirstFloorWeekly`. This path is easy to understand.

2. Double-click the Slot cell in the table row.
An **Edit** window opens that shows the Source Ord, in this case, it is `h:c24`.
3. Click the component chooser using the folder icon at the right hand side of the Source Ord property (BOrd fieldeditor).
The **Type** selection window toward the bottom displays either Slot or Handle.

In most cases, links and relations use the handle Ord because it is more fault tolerant of things like renaming the component, renaming a parent of the component, or moving the component. Whereas, if you use the slot path Ord as the reference, all of those sorts of changes break the link or relation.

Parent topic: [About ORDs](#)

The framework provides a powerful presentation architecture based on XML and the Niagara component model.

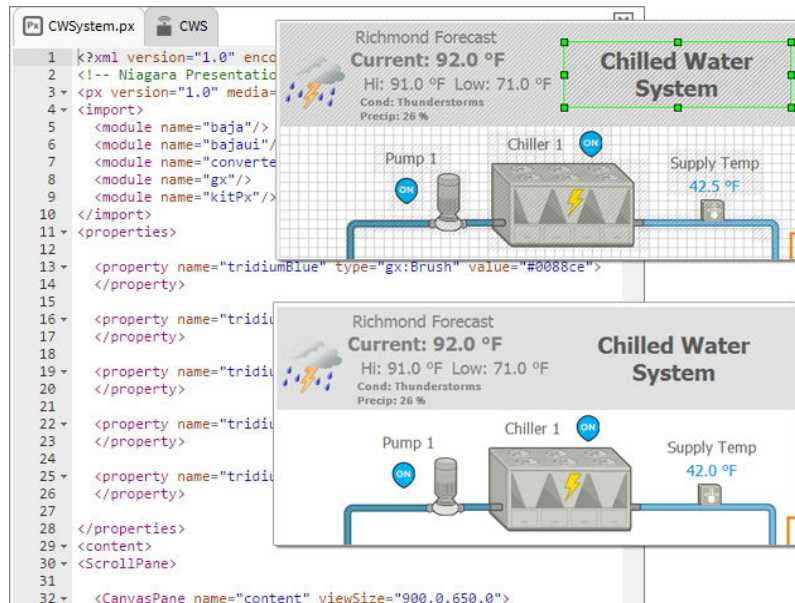
Presentation describes how the software framework provides visualization of information across different types of media. The terms information, visualization, and media may comprise the following:

- information

- real-time data
- historical data
- configuration data
- alarm data
- scheduling data
- graphical data
- textual data
- visualization
 - graphics (bitmap, JPG, PNG, SVG, vector)
 - videos
 - text documents
 - tables
 - charts
 - dashboards
 - input controls (actions, field editors, text fields, check boxes, trees)
- media
 - web browsers (HTML5, CSS3, JavaScript)
 - Workbench (framework stack)
 - mobile devices
 - csv
 - pdf
 - svg
 - xml
 - printed pages

An XML file format is used to present data. This file format is called “Px” for presentation XML. The term “Px” commonly describes the framework’s presentation architecture, that is how the framework visualizes information (text, graphics, alarms, and so on) across diverse media, such as: Workbench, web browsers, mobile devices, and so on. The following image shows an example of a Px file in the **Text Editor** (Px source file), the **Px Editor** view, and as displayed in the **Px Viewer**.

Figure 1. Px file in Text Editor (left), Px Editor (top right), and Px Viewer



- **About presentation modules**
The framework's presentation architecture is based on many modules and their associated public APIs.
- **About presentation design philosophy**
The presentation architecture is based on the following design principles. Understanding these principles will help you use the framework effectively.
- **About presentation media**
The framework supports four primary target media.

Parent topic: [Software building blocks](#)

The framework's presentation architecture is based on many modules and their associated public APIs.

- **baja** defines the core component model upon which framework subsystems are built. This document describes enhancements to the baja component model that are used by the presentation stack.
- **gx** provides an API for drawing 2D graphics in a device. The gx module deals with drawing primitives: color, strokes, gradients, vector geometries (line, rectangle, ellipse, paths), bitmap images, fonts, and transforms.
- **bajau** provides the widget toolkit. Widgets are the basis for graphical composition, layout, user input, and data binding. The bajau module builds upon gx to paint the widgets.
- **bajaux** provides dynamic HTML5 functionality. Bajaux widgets, such as the Dashboard widget, enable you to add data, modify the view, and save your customized version of the widget for subsequent viewing.
- **PDF** is a document implementation of the gx APIs.
- **HTML** is the framework's rendering and data binding engine.

Parent topic: [About presentation](#)

The presentation architecture is based on the following design principles. Understanding these principles will help you use the framework effectively.

- **Component model** — the core philosophy of every subsystem is to build atop the component model. The framework's presentation architecture is no exception. The design embraces a pure component model solution. The component model normalizes presentation using the document object model (DOM), which is always a BComponent tree.
- **Unified visualization** — presentations include a unified approach to representing graphics, text, and input controls all within a single component model. This allows all visualizations to share a common file format as well as a rendering, layout, and input API.
- **Unified media** — the design goal of Px is to build a single presentation that can be used across multiple media. For example, given a Px file, it can be automatically rendered in Workbench, as an HTML web page, or as a PDF file. All presentations are stored in the normalized component model as Px files.

Parent topic: [About presentation](#)

The framework supports four primary target media.

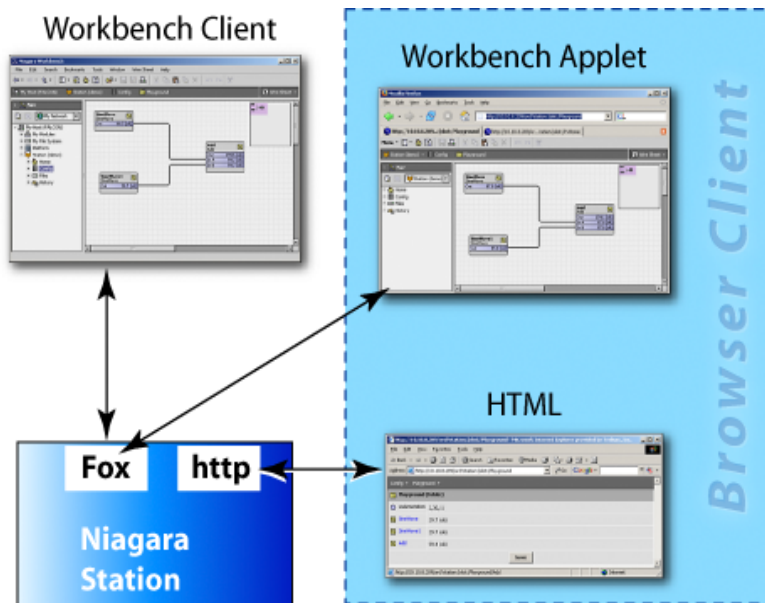
- **Workbench** — the stack must be available to take full advantage of the presentation architecture. The desktop version of Workbench provides a stand alone application that can render presentations with their full power. With the WbProfile feature, new custom applications can easily be built using Workbench and its presentation engine.
- **Web Launcher** — a Java-based web client application, which provides an applet-like Workbench environment running in an application window completely outside of a web browser.

When the Web Launcher application creates a Workbench view, that view is not contained within a web page. The view has no direct relationship to HTML elements in a web page, and the HTML does not know about the view. The view is not displayed as part of the layout of a page. This means that certain things that might have been possible with an applet in a browser do not work in Web Launcher. An applet contained within an HTML frame, for instance, cannot be supported in Web Launcher.

- **PDF** — Adobe's PDF (Portable Document Format) is the standard way to export a presentation for printing. PDF provides explicit control for how a presentation is rendered on paper in various sizes. It is also a convenient file format for access via HTTP or email. The presentation architecture includes an engine for generating PDF files from Px files.
- **HTML5** — provides enhanced capabilities for users and developers. A set of open web technologies (HTML5, CSS3, and JavaScript) provide a modern web interface using common standards. HTML5 views offer interactive functionality, makes it possible to edit properties and invoke commands right in the view. Other HTML5 functionality includes context sensitive menus and the ability to add data to views dynamically. For the designer/developer, consistent rendering across media means you can develop a view once and it renders in both Workbench and Hx interfaces. The bajaux HTML5 widgets included by default provide interactive charting and dashboarding functionality. The bajaux widgets also integrate into the environment. For example, commands defined for a WebWidget render as added icons in the Workbench tool bar or in a modern HTML5-capable web browser.

Refer to the *Niagara Graphics Guide* for information about presentation media technologies.

Figure 1. Presentation media options



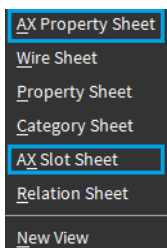
Parent topic: [About presentation](#)

When you connect to your station via Workbench or via browser using the HTML5 browser-based implementation, there are many ways to visualize your system and its components.

A view is a visualization of a component. One way to view a component is directly in the Nav tree sidebar. From the Nav tree you can right-click on an item and select one of its views to display in the view pane. For example, a component that appears in the Nav tree may have a **Wire Sheet** view, a **Property Sheet** view, and a **Px View**. Each displays in the view pane. Each component has a **default view** that appears whenever you activate a component (double-clicking, for example) without specifying a particular view.

In Workbench, two AX views are available under the **Views** selector menu and the right-click pop-up menu (as shown here): the **AX Property Sheet** view, and **AX Slot Sheet** view.

Figure 1. AX views under Views selector



Optionally, you can use the **Property Sheet** view, which combines the functionality of the two views. When working in this view, you click the **Slot Details** toggle command (upper right) to display and edit slot sheet details, as shown below. Enhanced functionality in this view invokes actions and edits properties.

Figure 2. Property Sheet view with enhanced functionality

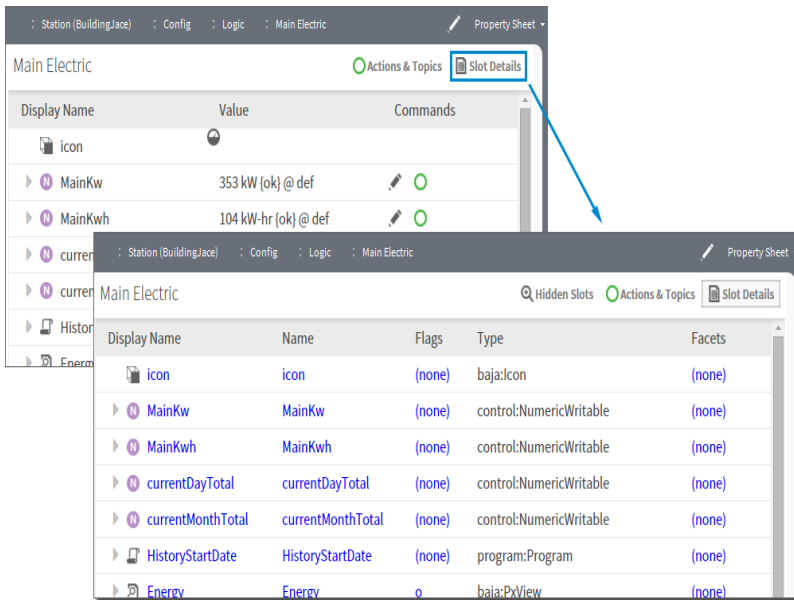
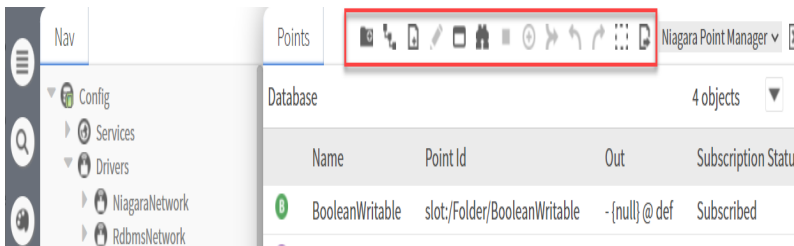


Figure 3. HTML5 view with enhanced toolbar



HTML5 browser-based views (as of Niagara 4.12) such as the Web **Wire Sheet** view or the HTML5 **Niagara Point Manager** view allow you to access your properties from mobile devices. Tailored to simplify your user experience, the enhanced toolbar makes it easy to edit properties and invoke commands using a handheld device. The HTML5 user interface provides context-sensitive menus and the ability to add data to views that renders instantly in Workbench.

Refer to the *Plugin Guides* section for a comprehensive list of views.

Parent topic: [Software building blocks](#)

The framework provides non-English language support by use of lexicons. A two-digit Java locale code identifies each lexicon, such as `fr` (French) and `de` (German). All of the lexicons are distributed as modules (`niagaraLexiconXx-rt.jar`) included in the software installation.

Niagara 4.0 requires that custom lexicon files be compiled as a module (`.jar`).

For detailed information on lexicons and using the **Lexicon Tool**, refer to the *Niagara Lexicon Guide*.

Parent topic: [Software building blocks](#)

A Baja Format (BFormat), is a class that returns object strings using a standardized formatting pattern language (script).

Script

A BFormat script consists of one or more calls chained together using the dot/period (.) operator. BFormat syntax requires that a percent (%) character begin and end each script. Like a wildcard or a variable, you embed BFormat script in static text strings. The system resolves the format (executes the calls contained in the embedded script) starting with the object that declares the format. Then the embedded calls (instructions) dynamically resolve to objects, and the system converts the final object into a string.

BFormats are very important for making templates (reusable portions of the data model tree that require minimal configuration), when creating Px views, and to enable localization (foreign language support).

Many properties that allow text entry support BFormat scripts. For example, a formula can refer to multiple points by using a Value Ord that contains BFormat script:

slot:/Drivers/NiagaraNetwork/%parent.name%/points/%name%

The system resolves the BFormat script in this example by substituting the name of the folder that contains the point for the %parent.name%, and the name of the point for %name%.

Call resolution sequence

BFormat scripts consist of one or more calls to methods that execute against system objects.

Within a BFormat, the system resolves calls in this order:

1. Special calls:

- time() returns the current time as a Niagara BAbsTime object
- user() returns the current user's name
- lexicon(module:key) get the specified lexicon text
- decodeFromString(module:type:escapedEncodedValue)
- substring(to) on a string
- substring(-fromEnd) on a string
- substring(from,to) on a string

2. Java method get<call>(Context)

3. Java method get<call>()

4. Java method <call>()

5. Java method get("<call>")

- [BFormat example: naming points, VAV scenario](#)
This example uses the BFormat %parent.parent% script to name points in alarm extensions.

- [BFormat example: naming histories](#)
This example uses a technique other than the `%parent.parent%` script to name histories. This method may be called the folder-level-independent method, as explained in this topic.
- [BFormat Px Widget examples](#)
BFormat scripts can be used with Px widgets as demonstrated by this example.
- [BFormat: WeatherService example](#)
This example uses the **WeatherService** to show another way to use BFormat script.
- [BFormat errors](#)
This topic covers a few example errors and considerations for dealing with errors.
- [BFormat default scripts](#)
The system populates the text properties of some components (copied from palettes or originated from manager views) with default BFormat scripts. Other text properties default to empty.

Parent topic: [About the Framework](#)

This example uses the BFormat `%parent.parent%` script to name points in alarm extensions.

This example involves a driver network of VAVs for 60 zones using 60 identical devices, each with identically-named proxy points, but under a uniquely-named device component. For simplicity, assume that the devices are named: VAV1, VAV2, ...to VAV60.

Several proxy points in each zone require an alarm extension. You could manually rename the points for all 60 devices (a task that could take hours), or use BFormats to configure all VAV point names in a single operation.

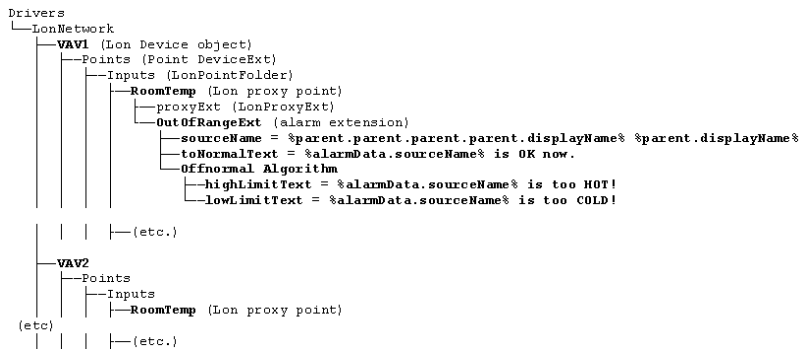
Manually renaming the points in each zone

This would involve entering unique names for all BFormat-capable properties under each alarm extension. In some cases, you may also have to modify related properties under the device's `offNormalAlgorithm` slot. Although not required, these changes would ensure that alarm records viewed in the alarm console provide unique source values (names) for each alarm. Without these unique names, only the station ords identify each VAV. For example, it can be difficult to isolate the zone and RoomTemp point for a specific alarm by looking for the point's ord.

Replicating properties using BFormats

This method will save you time. You begin by replacing the default values for several BFormat-capable (text) properties of one VAV's alarm extension such that generated alarms contain more useful source data, then you replicate the VAV application to the remaining 59 VAVs.

Figure 1. Example config structure and alarm extension property values using edited BFormat scripts



- **VAV1, VAV2, etc.** are Lon device objects.
- **Points** is the point device extension.
- **Inputs** is a Lon point folder.
- RoomTemp is a numeric Lon proxy point.
- proxyExt is a Lon proxy extension.
- OutOfRangeExt is a **RoomTemp** alarm extension.
- sourceName is a **RoomTemp** property. Instead of entering a value for this property on each of 60 property sheets, the following BFormat script resolves to a unique name for each VAV in the alarm console:
`%parent.parent.parent.parent.displayName% %parent.displayName%`

This name contains two variable scripts separated by a space. The variable on the left resolves to four parent nodes in the tree structure to arrive at the device name (VAV n). The variable on the right resolves to the proxy point name, in this example, RoomTemp. In the alarm console the alarm source displays as VAV n RoomTemp, where n is a number between 1 and 60.

- toNormalText is a **RoomTemp** property that sets up a message when a point in alarm returns to normal. Instead of entering a value for this property on each of 60 property sheets, the following BFormat script customizes the message for each VAV:

`%alarmData.sourceName% is OK now.`

This script resolves to the VAV name.

- OffnormalAlgorithm is the title of a group of properties on the property sheet.
- highLimitText is a RoomTemp property that sets up a message when the point returns a temperature above the defined limit:
`%alarmData.sourceName% is too HOT!`
- lowLimitText is a RoomTemp property that sets up a message when the point returns a temperature below the defined limit.

`%alarmData.sourceName% is too COLD!`

Here is the tricky part: All the alarm text properties are relative to the *alarm record* component generated by an alarm, and *not* to the alarm extension responsible for generating the alarm. Alarm text properties in the alarm extension `OffnormalAlgorithm` include:

- `toNormalText`
- `toOffNormalText`

If the extension is an `OutOfRangeExt`, alarm text properties include:

- `highLimitText`
- `lowLimitText`

These `OutOfRangeExt` properties override any entry in the `toOffNormalText` property of the alarm extension parent.

The location of these text properties in the alarm record means that you cannot use the `%parent.displayName%` script for the alarm text properties, at least not if you expect to get any useful results. But because each alarm extension's `sourceName` is now unique (using the technique above), you can reference it within alarm text type properties, along with any desired static text. Except here, the `sourceName` is an *alarmData* field, from the alarm record.

In the example , when `RoomTemp` in `VAV1` triggers a high limit alarm, the alarm data message text displays: `VAV1 RoomTemp is too HOT!`, and when it returns to normal the alarm data message text displays: `VAV1 RoomTemp is OK now..`

You could further modify the `OffnormalAlgorithm` high and low limit text properties to include the numerical (alarm) limit, using another *alarmData* field. For example, if `highLimitText` is set to this `BFormat` script: `%alarmData.source% is above %alarmData.highLimit% degrees!`, and the extension's `highLimit` is set to `74.5`, upon a high limit alarm the message text displays `VAV1 RoomTemp is above 74.5 degrees!`. This technique may be useful if routing the alarm, for example, to a cellphone, where a minimum amount of alarm data text, including only the timestamp and the message text, are needed.

Note: To see what *alarmData* fields are available for use in this manner, go to a station's alarm console and view the complete details (**Alarm Record** window) for any one alarm.

Parent topic: [Formats \(BFormat\)](#)

This example uses a technique other than the `%parent.parent%` script to name histories. This method may be called the folder-level-independent method, as explained in this topic.

History extension scenario

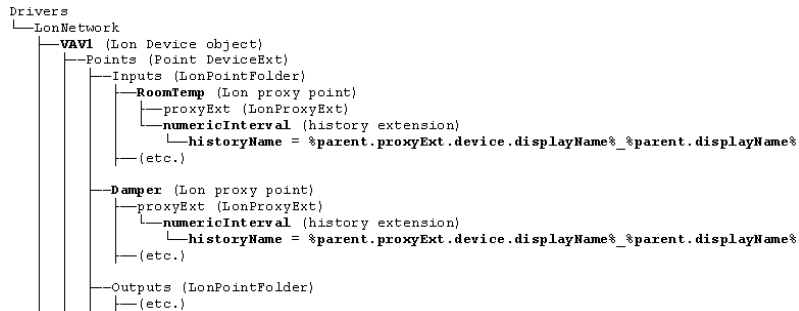
Consider a VAV network with replicated device applications. In each VAV zone, you need the histories of several proxy points, all identically named. For example:

- Each zone requires a numeric interval history on **RoomTemp**.
- Each zone requires a numeric interval history on **Damper**.

To manually configure these requirements you must either rename the parent point(s) or rename the history extensions' *historyName* to something unique, as duplicate history IDs are forbidden.

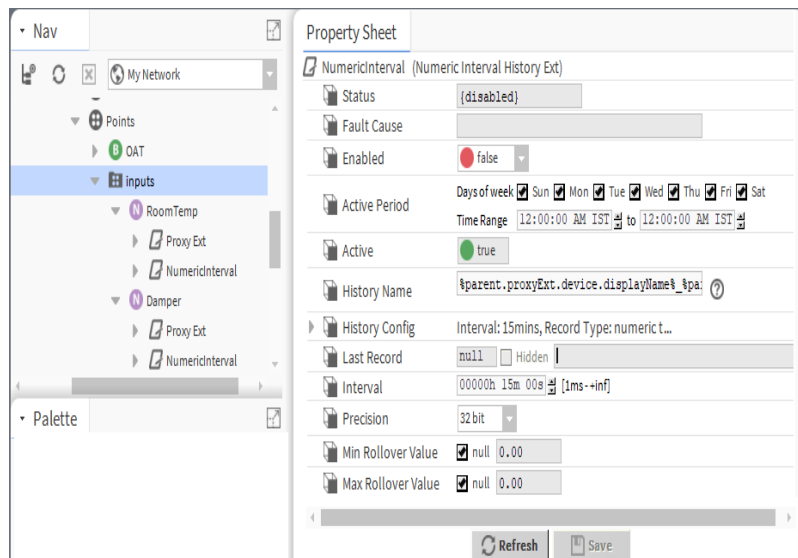
To automatically create unique histories names, before replicating this VAV application, you add a BFormat to the historyName in all history extensions, similar to editing the sourceName when naming points.

Figure 1. Example config structure and history extension property values using edited BFormat scripts



- **VAV1** is a Lon device object.
- **Points** is the point device extension.
- **Inputs** is a Lon point folder.
- **RoomTemp** and **Damper** are numeric Lon proxy input points each with a **proxyExt** and a **numericInterval** history extension as child nodes.
- historyName is a **numericInterval** property on each history extension (double-click the **numericInterval** node to view its property sheet).

Figure 2. Lon device points showing a NumericInterval Property Sheet



Instead of entering a value for this property on each of 60 property sheets, the following BFormat script resolves to a unique name for each

```
VAV:%parent.proxyExt.device.displayName%_%parent,displayName%
```

This name contains two variable scripts separated by an underscore.

- The variable script on the left of the underscore uses a special `getDevice()` method, where it resolves the proxy point's parent device name regardless of its folder depth under the **Points** extension. (In this example, proxy point **RoomTemp** is in an **Inputs** point subfolder, while the **Damper** proxy point is in the root of the **Points** extension).

Because the points are located in different folders, the `%parent.parent%` script would work for **RoomTemp**, but not for **Damper**. The folder-level-independent method, however, is more fault tolerant as a result of moving a proxy point, especially to change its hierarchy.

- The variable script to the right of the underscore resolves to the proxy point name (**RoomTemp** and **Damper**)

Given the tree structure of this network, the resulting histories appear as VAV1_RoomTemp, VAV1_Damper, and if replicated, VAV2_RoomTemp, VAV2_Damper, and so on.

Here is how this works: `%parent.proxyExt.device.displayName%`, the `parent`, steps up one level to the Lon proxy point (say, RoomTemp). The `proxyExt` is the slot name that walks back down the tree to a different child component, in this case to the LonProxyExt. The device calls the `getDevice()` method, and the `displayName` calls the `getDisplayname()` method.

This example assumes that no other components in the station also have a VAV component with a RoomTemp child, which also requires a history extension. Also, the example uses an underscore instead of a space even though spaces in object names are permitted (history being one type of object), they are escaped in the database using a "%20" string. This can be confusing in certain scenarios.

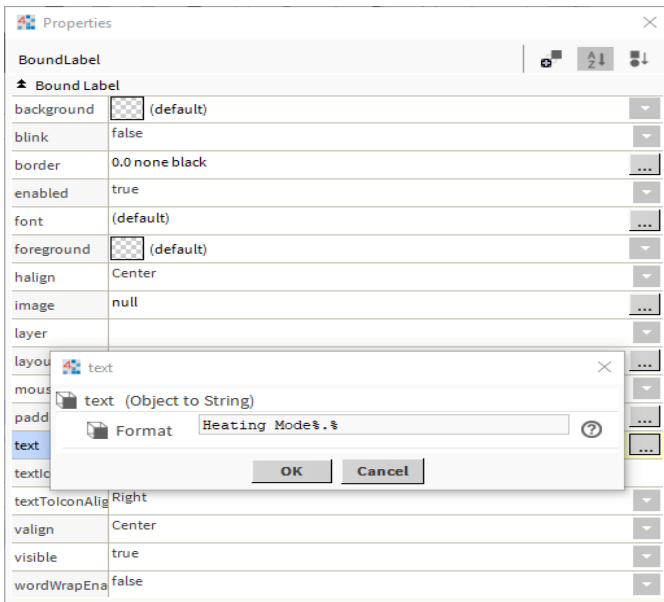
Parent topic: [Formats \(BFormat\)](#)

BFormat scripts can be used with Px widgets as demonstrated by this example.

Px widget scenarios

When you configure Px widget properties, especially for BoundLabel types with a binding to a component, double-clicking the Text property opens the **Text** window.

Figure 1. Example of adding static text in Text property of a BoundLabel



The `ObjectToString`, `Format` property determines the content of the displayed text. This property defaults to `%.%`, which displays the bound component's ord. In the example, a simple edit adds the words "Heating Mode:" in front of this default `BFormat` script.

For any point (or any component with an `Out` property), default text from the binding is identical to the out value displayed in the component's property sheet. The default text that is identical to the `Out` value includes facets, as well as the following information:

- If bound to a *writable* point, the **Text** Format contains three pieces of data, namely: `<value> <status> @priorityLevel>` where:
 - `<value>` is
 - `<status>` is {ok}, etc.
 - `@priorityLevel>` is

For example: On {ok} @16 (a `BooleanWritable`) or 20% {ok} @12 (a `NumericWritable`)

- If bound to a read-only point, the **Text** Format provides two pieces of data, that is: `<value> <status>`
For example:

Clean {ok} (a `BooleanPoint`) or 72.3 °F {ok} (a `NumericPoint`)

- If bound to a component that is not a point (there is no `Out` property), you must bind to a particular *slot* of that component, in order to display text other than its component type.

For example, if you drag a `DegreeDays` component to a Px page, the system displays the default text: Degree Days. However, if you change the binding's ord to `<objectName>/`

clgDegDays, the system calculates and displays the cooling degree-days value (and status), such as: 5.0 {ok}

- **Text scripts for points**

You can edit the **Text** Format property in any BoundLabel widget to include additional static text, and/or modify (or limit) the real-time data in the text.

- **More about text scripts**

Object-to-String scripting is quite flexible when working with BoundLabel widgets. You are limited only by your understanding of Baja (see online Bajadoc in the Help system).

Parent topic: [Formats \(BFormat\)](#)

You can edit the **Text** Format property in any BoundLabel widget to include additional static text, and/or modify (or limit) the real-time data in the text.

The following table provides a few example BFormatting scripts and results for writable points.

Text (BFormat) script	Description	Example 1	Example 2 (script and result)
%out.value%	Value only (with facets).	On	AHU is %out.value% AHU is On.
%out.status%	Status including priority level, if writable point.	{ok} @ 16	Status of AHU is %out.status%. Status of AHU is {ok} @ 16.
%activeLevel%	Number only (1-16, def) for priority level, writable points only.	16	AHU is %out.value% at level %activeLevel%. AHU is On at level 16.
%status.flagsToString%	String value(s) for status flags set, without braces. If non-ok.	ok	AHU status is %status.flagsToString%. AHU status is ok.

The **Example 1** column illustrates the resulting text if the Out script is On {ok} @ 16.

The **Example 2** column shows the script and the resulting display when static text is added to the script.

Parent topic: [BFormat Px Widget examples](#)

Object-to-String scripting is quite flexible when working with BoundLabel widgets. You are limited only by your understanding of Baja (see online Bajadoc in the Help system).

For the non-developer, these few simple rules may help:

- The BoundLabel widget must actually be bound to an object (using an ord). In other words, you cannot simply drag a BoundLabel from the **kitPx** palette to the Px page, edit the **Text** Format property, and get results. If needed, the BFormat script you use may be totally unrelated to the bound object. For example, you can bind to any object and enter a system-type call, as shown here:
%time().toDateString% to produce text like "01-Nov-08"

- Relative to the bound object, you can use the parent technique to “walk up” the component tree of the text for a slot (or name), for example:
`%parent.parent.name%` gives the name of the parent two levels up.

For example, a `BoundLabel` bound to a `DiscreteTotalizerExt` under a `BooleanPoint`, where you wish to display the (parent) point’s name and the number of times it has changed state since its last reset, could be achieved using this Text script:
`%parent.displayName% had %changeOfStateCount% COS since last reset.`

The result might be: `ChWPump2 had 14 COS since last reset..`

- In addition, relative to the bound object, you can also “walk down” the tree in a parallel path, using the slot name (versus name or displayName).

An example of this “walk down” method (via slot name) is in the history extension (historyName) example, along with the parent technique.

Parent topic: [BFormat Px Widget examples](#)

This example uses the **WeatherService** to show another way to use BFormat script.

An ORD as a reference

The **WeatherService** can provide many pieces of information, including current conditions and forecasts.

Figure 1. Example Weather Report property sheet

The screenshot shows a 'Weather Report' property sheet for 'Charlotte'. It includes a 'Status' field set to '{ok}', a 'Provider' dropdown set to 'NC', and a 'Current' weather display showing '59.0 °F, Mostly Cloudy {ok}'. Below this is a 'Today' section with a detailed forecast: '61.0 °F/43.0 °F, Increasing Clouds, 10.0 % {ok}'. This section also includes sub-fields for 'Status' ({ok}), 'Last Update' (23900109), 'Date' (12-Feb-2007 EST), 'Sunrise' (07:11:52 AM EST), 'Sunset' (06:01:47 PM EST), 'High' (61.0 °F {ok}), 'Low' (43.0 °F {ok}), 'Precip Chance' (10.0 % {ok}), and 'Weather Summary' (Increasing Clouds {ok}). At the bottom is a 7-day forecast table:

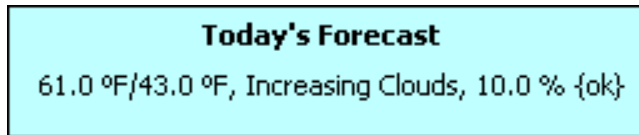
Day	Forecast
Tomorrow	52.0 °F/34.0 °F, Rain, 78.0 % {ok}
Wednesday	44.0 °F/24.0 °F, Partly Cloudy, 11.0 % {ok}
Thursday	44.0 °F/20.0 °F, Mostly Sunny, 14.0 % {ok}
Friday	42.0 °F/20.0 °F, Mostly Sunny, 11.0 % {ok}
Saturday	49.0 °F/28.0 °F, Partly Cloudy, 10.0 % {ok}
Sunday	49.0 °F/-, Partly Cloudy, - {ok}

To display this information on a graphic you create a bound label that references the **WeatherService** and the applicable property. For example, to display the forecast for today in Charlotte, NC, the referenced ORD would be `station:/Services/WeatherService/Charlotte/day0`.

Default BFormat script

Instead of entering this ord, you could expand the **WeatherService** in the Nav tree to find this slot, then drag it to the Px page, where the Make Widget wizard automatically resolves to this ord. The default BFormat script of `% . %` for Text returns the information shown as the second line in the following image.

Figure 2. Default BoundLabel Text to Weather Report's Today property



Modified BFormat script

To further control weather values, you can add additional BFormat scripts. For example, setting Text to the following:

High %High.value% °F / Low %Low.value% °F / Precip %PrecipChance
.value%%
returns the information shown in the second line of the following image.

Figure 3. Example of a modified BoundLabel Text to Weather Report's Today property



Note: The system uses the percentage symbol (%) To delimit scripts in format Text fields. To display this symbol as text, enter two of them (%%).

Parent topic: [Formats \(BFormat\)](#)

This topic covers a few example errors and considerations for dealing with errors.

Syntax errors

Not all attempts at customizing Format-type property values may be successful. If a syntax error causes the script to fail, an ERROR or `err: <item>`, where *<item>* is the script value appears in the produced text. For example:

- If you forget a % on BoundLabel Text entry, say: Fan is %out.value, the system displays: ERROR Fan is %out.value.
- Or, a script call to a misnamed slot might fail with a displayed error similar to: ChwPump2 had %err:control:DiscreteTotalizerExt:changeOfStates% COS since last reset.

You should test all modifications to BFormat scripts, to make sure you get the intended results.

Security breaches

To prevent a BFormat scripts from calling an object and, in the process, unintentionally changing the state of the object, the framework maintains a BFormat blacklist of prohibited script call methods. By default, the methods listed here generate an error when any attempt is made to execute them via a BFormat:

- Any method that returns void. No exclusions are allowed.
- All synchronous action calls (that is, doAction()) calls that return a BValue . Exclusions are allowed.
- Any call to the submit (Context) method on any subclass of BJob.
- Any call to the clear () method on **BDaySchedule** and **BEnumSetSehcdule**.

Parent topic: [Formats \(BFormat\)](#)

The system populates the text properties of some components (copied from palettes or originated from manager views) with default BFormat scripts. Other text properties default to empty.

This table lists examples of the components with default scripts.

Table 1. Default BFormat scripts for a few properties

Component	Property	Default Script	Notes
Alarm extension for points, e.g. OutOfRangeExt, etc.	sourceName	%parent.displayName%	Suitable as is in many cases, such as where all parent points are uniquely named.
History extension for points, e.g. NumericInterval, etc.	historyName	%parent.name%	
Any network component's AlarmSourceInfo slot.	sourceName	%parent.displayName%	Often both properties work with these default values.
Any device-level component's AlarmSourceInfo slot.	sourceName	%parent.parent.displayName% %parent.displayName%	
EmailRecipient	subject	%alarmData.sourceName%	The system provides much additional alarm data in the slot that contains the body of the email.

For a property value you can enter multiple BFormat scripts along with static text, as demonstrated by the defaults for a device's AlarmSourceInfo (sourceName) property in the table. A static space character separates the %parent.parent.displayName% from %parent.displayName%. The subject property of the EmailRecipient contains static text (Niagara Alarm From) ahead of the BFormat script (%alarmData.sourceName%).

BFormat scripts can save time by enabling the replication of applications, where you can achieve the desired result with minimal custom edits to property values. You may still custom format specific text as needed.

Parent topic: [Formats \(BFormat\)](#)

Workbench is the name for the framework's graphical user interface. There are several ways to start Workbench. Depending on your configuration and licensed features additional information may be required.

You are working on a PC.

1. To open Workbench click start and navigate to the Niagara folder.
The Start menu displays the Workbench options.
2. Click on one of the following:
 - To open from an Administrator: Command Prompt window, type `-- start_wb` or `-s` in the command prompt.

The splash screen opens.

- To open with a background command-line console and launch Workbench, click (Console), type `wb` at the console command prompt, and press **Enter**.

Workbench opens with a console, which stays open even if you close the program.

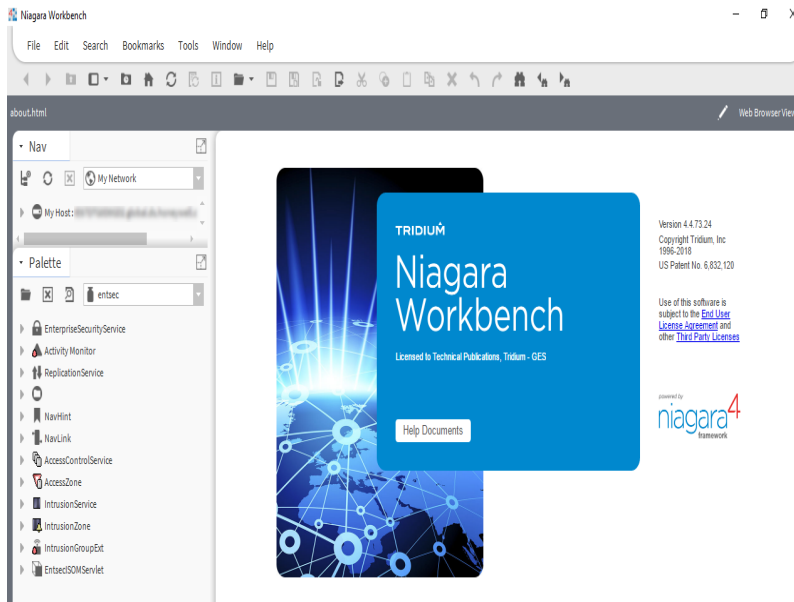
- To open Workbench and a console, click Workbench (Console).

This opens a command window that immediately launches the program. Closing this console closes the corresponding instance of Workbench.

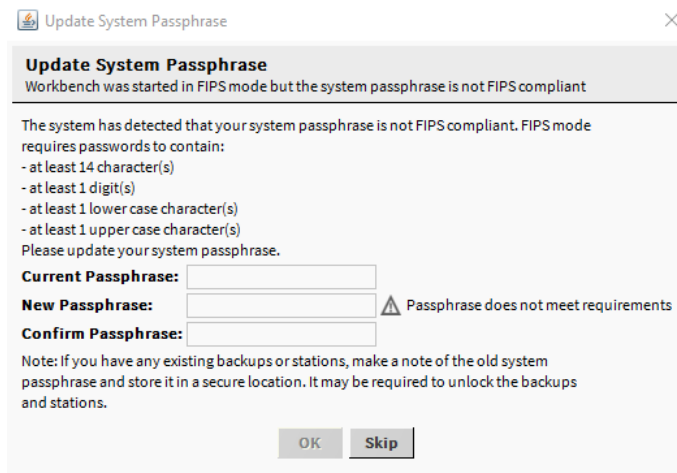
- Click Workbench alone or double-click its icon on your desktop.

The splash screen opens.

Tip: Opening Workbench with a background console ensures that you (and Technical Support) have ready access to important diagnostic information in the event that something goes wrong in the Supervisor platform or station. If you wait until you are already experiencing symptoms before you open the console, you may miss the opportunity to gather important data.



If you skipped the passphrase when you installed the software or if your installation requires FIPS, and the passphrase you created does not meet the FIPS requirements Workbench prompts you to define the passphrase now.



3. Enter the current passphrase (defaults to the platform password for new installations), and create a new passphrase or skip it again and click **OK**.
If the platform requires FIPS, skipping this configuration window disables FIPS.
4. If you skipped updating the FIPS passphrase, click **File > Non-FIPS Restart**.
Workbench starts again and prompts you to confirm that you choose to work without FIPS.
5. Later, to re-enable FIPS, click **File > FIPS Restart**.
Workbench restarts.

- **Tour of the Workbench GUI**
When you start Workbench, The home window opens.

- [About window controls](#)
The interface provides typical Windows-type controls plus other features unique to the framework.
- [About the side bar panes](#)
Side bar panes are normally visible only if a side bar is open. All side bars display in the side bar pane and have some common features, such as the side bar title bar. When you close all side bars, the side bar pane collapses and the view pane and console pane (if open) expand to fill the window.
- [Group Nav Tree Host](#)
- [About popup menus](#)
Context-specific popup menus control system options. Right-clicking on a component or a view opens these popup menus. The contents of each menu varies depending on the context.
- [Table controls and options](#)
Many Workbench views present information in a table. All tables share similar features and controls.
- [Types of edit commands](#)
In addition to view-specific editing tools, commands for editing components are available in many of the views.
- [About keyboard shortcuts](#)
Workbench provides a number of keyboard shortcuts for common actions. These are performed by holding down the combination of keys listed. They include:
- [Using the help system](#)
The help system opens in a sidebar or its own window.
- [Managing bookmarks](#)
Bookmarks customize the way you use the interface. You may create as many bookmarks as needed.
- [Opening and closing a side bar](#)
- [Opening a palette](#)
- [Adding a standard component to the Nav tree, a Property Sheet or Wire Sheet](#)
The standard components include the control point types, two folders and a text block. You can add one of these to the Nav tree or to a **Property Sheet** or **Wire Sheet**. To add a slot to a container refer to “Adding a slot.”
- [Reorganizing components in a station](#)
You can drag a component to a displayed **Property Sheet** by using a right-click drag, which prompts you to choose an action from a menu or by using a normal, left-click and drag.
- [Simplifying the Nav tree](#)
The Nav tree hierarchy can become confusing to navigate especially when the tree includes many levels of parent-child component relationships. The **Go Into** command can restrict the tree to show only the nodes you are currently working on. Then, when you finish your work in one area of the tree, you can return all nodes to the standard, unrestricted view.
- [Opening a new tab](#)
You can use tabs to help organize and selectively display information in the view pane. Workbench allows you to create multiple tabbed views.
- [Creating additional windows](#)
You can customize each individual window, as needed, to access different information, allowing you to see multiple concurrent views.
- [Working with rows in a table](#)
Editing more than one table row at a time is sometimes called “batch editing,” or “batch processing.”
- [workbench-WebChart](#)
This view plots historical data, live historical data, and live data, as well as schedules. It is the default history view for history records in Workbench and in Hx, and a secondary view on schedules and Enum, Numeric, and Boolean points. Legacy charts, those created in earlier releases, are available as secondary **History Chart** views on history records.

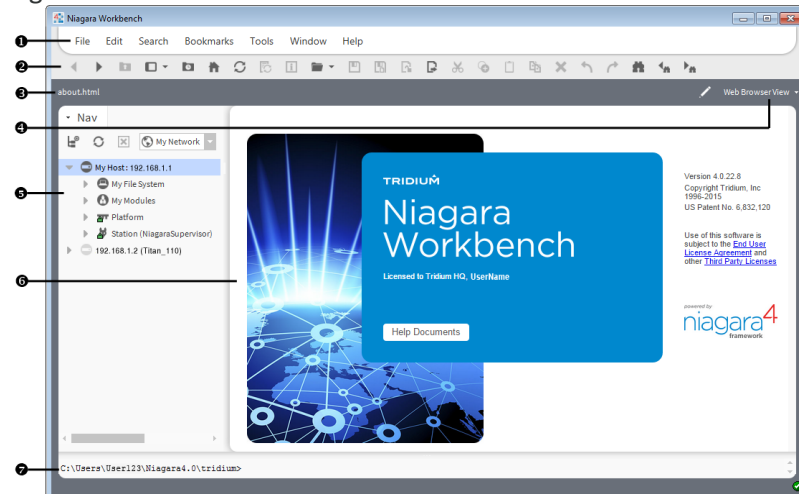
- **History Chart view**

The History Chart view uses one or more of the controls and options described in this topic.

When you start Workbench, The home window opens.

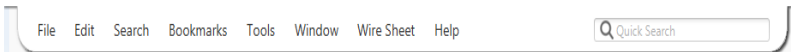
When you start Workbench, The home window opens.

Figure 1. Home view



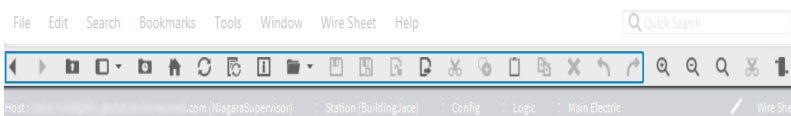
The home view is divided into seven areas:

- 1 Menu bar — contains available program menus.
Figure 2. Menu bar



Many of the menus are context-sensitive and only appear when certain views are active. In a station connection, the **Quick Search** field displays on the right-side of the menu bar when the SearchService is installed on the connected station.

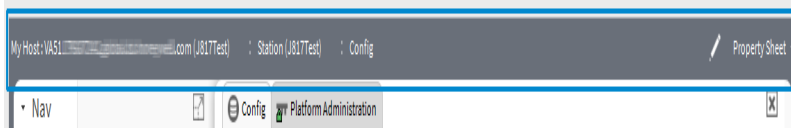
- 2 Tool bar — contains icons for typical interaction with the interface plus icons specific to the view currently in use. Hovering the mouse pointer over an icon invokes a tool tip. The tool bar is the row of icons, just below the menu bar, that provides icons for actions affecting the objects that appear in the view pane. Usually, tool bar icons provide single-click access to many of the most commonly used features of the Workbench.
Figure 3. Tool bar icons



The primary icons (highlighted in the above image) are always visible, Additional sets of icons are added to the toolbar when you select certain views. For example, when the **Wire Sheet** view is active, the **Delete Links** icon and **Zoom** icons are available.

When an icon is dimmed, it is unavailable. Hovering the mouse pointer over a toolbar icon invokes a window description known as a “tool tip”.

- 3 Path bar (locator bar) — located just below the toolbar, this area contains the path or Ord for the current view.
Figure 4. Path bar

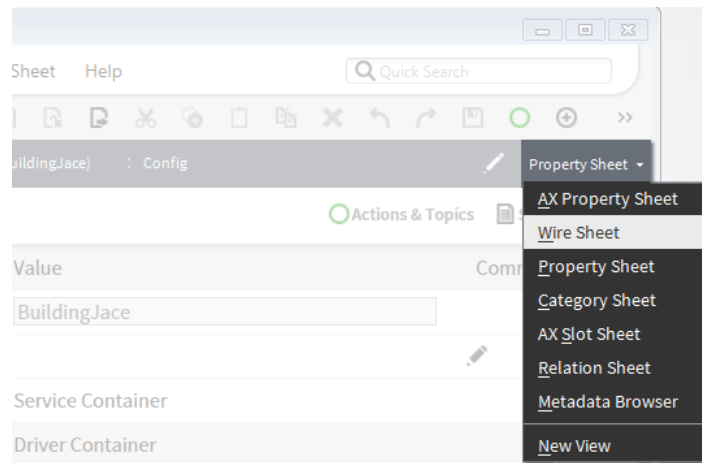


The left side of the path bar shows your current location (Ord or web address). The **View** selector appears on the right side.

The purpose of the path bar is to provide a graphical navigation field for selecting, displaying and entering destination references. The path bar serves several functions:

- It updates automatically each time you select a new view - so that it shows you the Ord of each view.
- The system displays an Ord in a graphical row of icons, so that when you hover the mouse pointer over an icon (or click on any icon) along that Ord, you can access any child node from the Ord's graphical drop-down list.
- The path bar functions much the same as a browser address field, permitting you to enter an Ord or a URL. Click the  (Edit Path) icon to enter a different Ord or a web address (internal or external).

- 4 View selector — a context sensitive menu with options that allow you to quickly display different views of the information that is currently in the view pane. This selector appears on the right side of the locator bar, just below the tool bar.
Figure 5. View selector



The options in the view selector differ, depending on the current view pane contents. For example, the view selector options that are available when you are viewing the platform in the view pane are different from the options that are available when you are displaying the **Driver Manager** view.

- 5 Side bar pane — left-side area displays one or more side bars that you may select from the **Windows** menu. For example you might have the following open at the same time: Nav tree, Search side bar, and a module palette.
- 6 View pane — This pane, located on the right-side of the window, displays the currently selected view for the active tab. It is the largest display area below the locator bar. Features of the view pane include multiple tabbed views and a thumbnail view.
- To change the selected view do any of the following:
- Double-click on an item in the Nav tree.
 - Select a view or action from a Nav tree palette menu.

- Select an option from a menu or submenu.
- Select an option from the locator bar.

The thumbnail view, when active, appears in the top right corner of the wire sheet provides orientation.

- 7** Console — bottom area provides access to a command line prompt without leaving the Workbench environment. To hide or show the console, select **Window > Hide Console** or **Window > Console** from the menu bar.

Figure 6. Example of a console

```
C:\Users\... \Niagara4.0\tridium>help

Niagara Shell Help (Version 4.0.22.8)
cd      View and change current directory
debug   Turn debug tracing on and off
print   Print to the output stream
reset   Reset the environment to its default state
set      View and modify environment variables
which   Resolve a filename in path

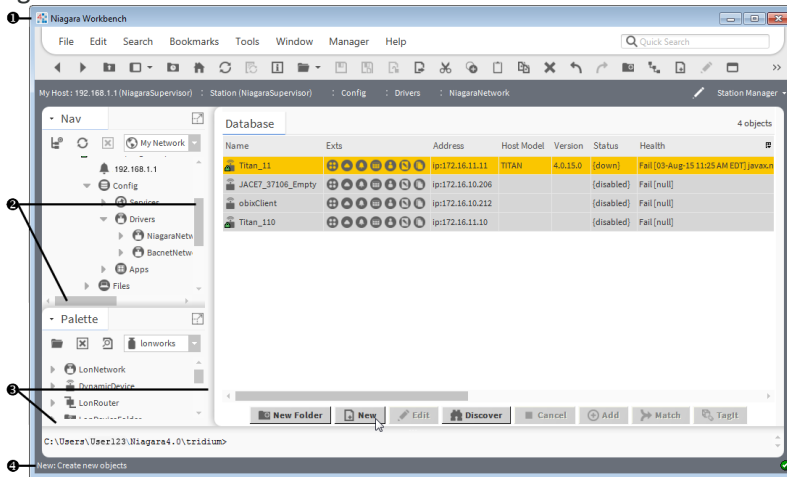
C:\Users\... \Niagara4.0\tridium>
```

The console has scroll bars on the right side and the window size may be adjusted by dragging the top border bar. From the console you may type in commands directly, including the help command for additional help.

Parent topic: [Getting to know Workbench](#)

The interface provides typical Windows-type controls plus other features unique to the framework.

Figure 1. Window controls



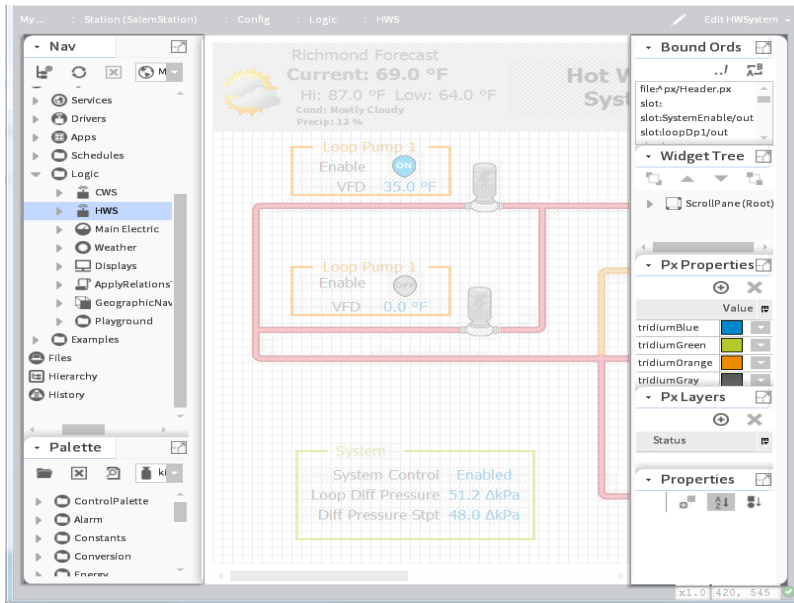
- 1** Title bar has standard Windows title bar features, including window name and icons to minimize, maximize, and close. Double-click the title bar to toggle between maximized and a sizable window. You may create additional windows after starting Workbench. All have these basic features.
- 2** Scroll bars appear in window and window pane areas (side bar, view, or console) when some content portions are not visible. They are along the right and/or bottom portions of a window or pane. You may drag a scroll slider (highlighted area) to scroll quickly or click an ending scroll arrow to move in incrementally.
- 3** Border controls resize the entire window. Drag the inside border between the side bar and view areas, or (if shown) the console area to change their relative sizes.

- 4 Status bar, at the bottom left of the window, displays tool tips for icons in the toolbar and for buttons in views. When working in views other details are displayed in the status line as well.

Parent topic: [Getting to know Workbench](#)

Side bar panes are normally visible only if a side bar is open. All side bars display in the side bar pane and have some common features, such as the side bar title bar. When you close all side bars, the side bar pane collapses and the view pane and console pane (if open) expand to fill the window.

Figure 1. Side bar panes



There are two types of side bar panes are:

- The primary side bar pane is the area on the left, below the locator bar. It displays the Nav tree and **Palette** window.
- The **Px Editor** side bar pane appears on the right side of screen and is available when the **Px Editor** is active.

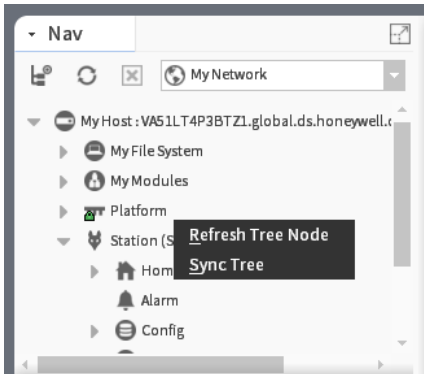
You use a menu to perform all operations that are available from inside the side bars (such as cutting or pasting). You use the icons in the title bar to open, close, and expand/restore the sidebars. To resize side bar height and width, click and drag on the borders.

Parent topic: [Getting to know Workbench](#)

Context-specific popup menus control system options. Right-clicking on a component or a view opens these popup menus. The contents of each menu varies depending on the context.

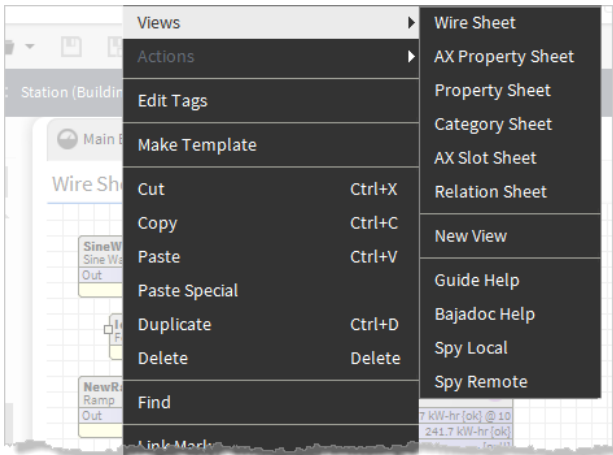
While most menu options are self-explanatory (Cut, Copy, Paste, etc.), all menu options are documented in the reference section at the end of this guide.

Figure 1. Nav side bar popup menu



Right-clicking in the Nav tree side bar without selecting a component displays a very short (two-item) popup menu. Selecting a component in the Nav tree side bar displays a much longer popup menu.

Figure 2. Wire sheet popup menu

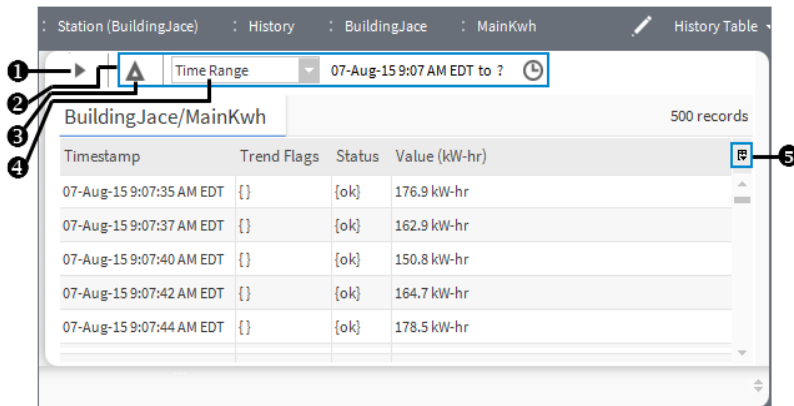


Right-clicking on the **Wire Sheet** displays a menu with different options.

Parent topic: [Getting to know Workbench](#)

Many Workbench views present information in a table. All tables share similar features and controls.

Figure 1. Table controls and options



1 Live Updates — The “play” icon, available for the **History Table** view, starts Live Updates (On Demand) updating. The icon changes to a “pause” icon while Live Updates is active.

2 Data parameters — These controls include **Delta** (for history logging) and **Time Range** settings.

3 Delta — Useful for history logging, displays value changes (delta) in your table.

4 Time range — The drop down option list has a variety of predefined time range options, including an option that allows you to restrict your data presentation to a particular date and time range that you specify.

Title bar — Displays the name of the data collection on the left side of the title bar and in some tables (collection table, history table, alarm extension manager, and others) displays the total number of records in the table on the right side of the title bar.

Column headings — Each column of data has a title that indicates the data type.

Column boundaries — Each column has a movable column boundary that can be used to re-size the column using the mouse control. Stretch or shrink column width by dragging the column boundary, as desired. Use the Reset column widths menu item to reset all column widths to their default size.

5 Table Options — Located in the upper right corner of the table, the drop down list provides one or more of the following controls and options:

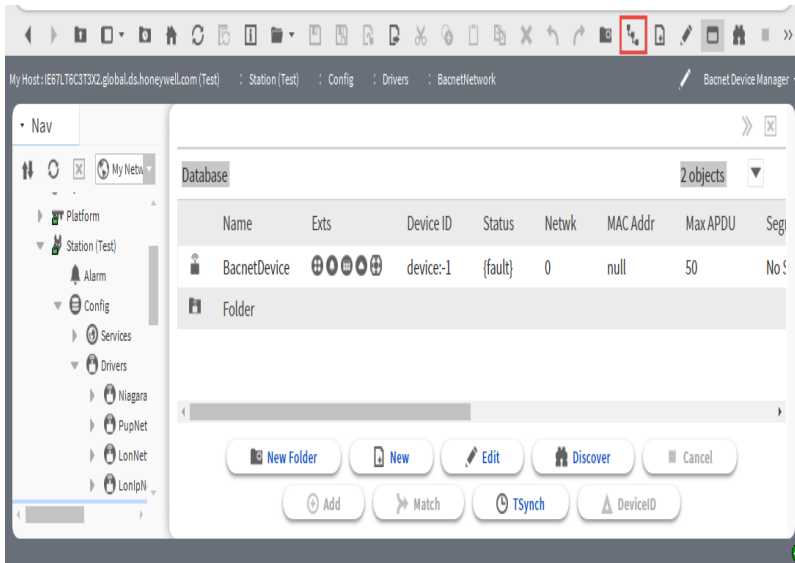
- **Reset column widths** — sets all columns in the table to their default widths. This is useful if you manually changed widths of columns, and now some contents are hidden (even after scrolling).
- **Export** — opens the **Export** window where you can choose to export the table to PDF, text, HTML, or CSV (comma separated variable).
- **Context-sensitive menu items** — additional context-sensitive menu items appear depending on the component that you are viewing.

Device Manager tables

The **Device Manager** of a device folder is unaware of devices in other device folders (or in the root of the network component). However, from the root **Device Manager** view (of the network), you can flatten the folder structure to view all devices by clicking the **All Descendants** tool on the toolbar.

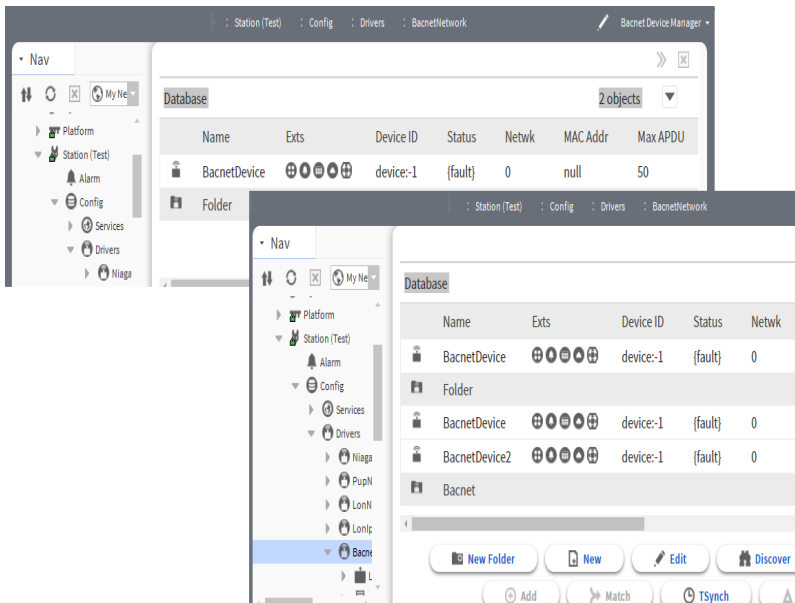
Note: In a large network, many subscribed device folders and devices, using the all descendants feature may decrease system performance.

Figure 2. All Descendants command



The result is a view that includes the devices managed by multiple device folders.

Figure 3. Device Manager showing multiple folders



If you are using device folders, and you click on a table column to restart devices, the view loses any device-to-device folder organization. However, you can always see the contents of device folders clearly in the Nav tree, and again when returning to the **Device Manager** view from another view.

Parent topic: [Getting to know Workbench](#)

In addition to view-specific editing tools, commands for editing components are available in many of the views.

Following, is a list of the standard commands that are available in Workbench views for editing components.

Command/ Action	Description
Drag	Works within a single application. However, it is possible to drag files from Windows Explorer a view. Dragging a component or file using the left mouse button performs a copy operation. Dragging a component or file using the right mouse button always prompts you for a Copy , Move , or Cancel menu selection.
Cut	Deletes the selected object and send it to the clipboard.
Copy	Sends the selected object to the clipboard without deleting it
Paste	Copies the current contents of the clipboard to the destination as a set of new dynamic properties.
Duplicate	Creates a copy of the current selection in the same container as the selection.
Delete	Removes a selected item from its parent container.
Undo	Reverses the previous command. Undo is only available for certain commands, such as, Paste, Cut, Delete, and the Link action
Redo	Restores a command-action after the Undo command has removed it.
Rename	Changes the name of a component.

Parent topic: [Getting to know Workbench](#)

Workbench provides a number of keyboard shortcuts for common actions. These are performed by holding down the combination of keys listed. They include:

- Help On View (F1)
- Console (F3)
- Hide Console (F4)
- Find (F5)
- SearchReplace (F6)
- Goto File (F7)
- SearchConsoleNext (F8)
- Save amp; Compile (F9)
- Back (Alt + Left)

- Forward (Alt + Right)
- Up Level (Alt + Up)
- Home (Alt + Home)
- Recent Ords (Alt + Space)
- Add Slot (Ctrl + A)
- Copy (Ctrl + C)
- Duplicate (Ctrl + D)
- Find Next (Ctrl + F)
- Goto Line (Ctrl + G)
- Open Ord (Ctrl + L)
- New Window (Ctrl + N)
- Open File (Ctrl + O)
- Print (Ctrl + P)
- Rename Slot (Ctrl + R)
- Save (Ctrl + S)
- New Tab (Ctrl + T)
- Paste (Ctrl + V)
- Word Wrap (Ctrl + W)
- Cut (Ctrl + X)
- Undo (Ctrl + Z)
- Find Bajadoc (Ctrl + F1)
- Active Plugin (Ctrl + F4)
- Find Files (Ctrl + F5)
- Replace Files (Ctrl + F6)
- Compile (Ctrl + F9)
- Next Tab (Ctrl + PageUp)
- Previous Tab (Ctrl + PageDown)

- SearchConsolePrev (Shift + F8)
- Find Prev (Ctrl + Shift + F)
- Redo (Ctrl + Alt + Z)

Parent topic: [Getting to know Workbench](#)

The help system opens in a sidebar or its own window.

You are using Workbench.

1. To open the help system do one of the following from the menu bar:

- **Select Window > Side Bars and click Help.**
- **Select Help and click Help Contents.**

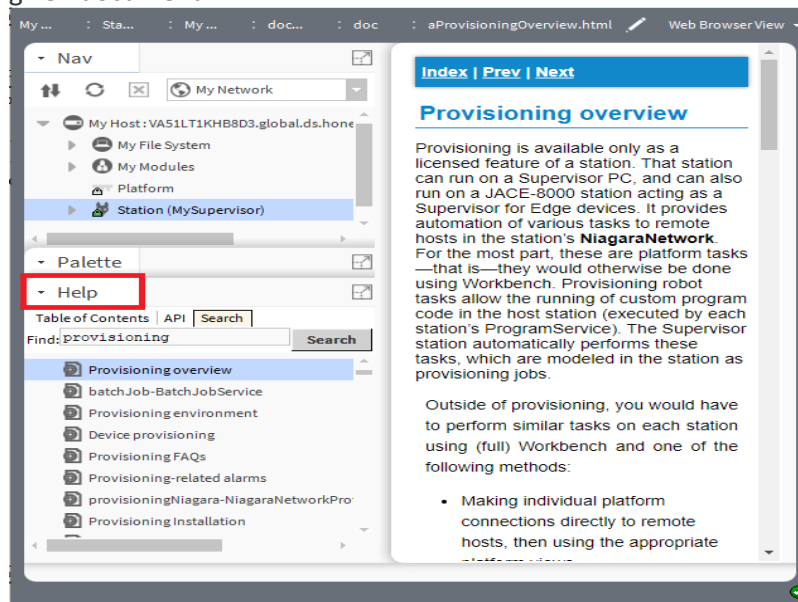
If you used the Window menu the **Help** sidebar opens in the left pane. If you used the Help menu the **Help** window opens.

2. Click the **Search** tab.

The Find: property opens.

3. Enter one or more words on which to search and click **Search**.

The system creates a list of documents that contain one or more occurrences of the words you have entered. The results show those document's containing any of the words, not all of the words. The ranking order is based on how many times any of the words appear in a given document.



4. Double-click on a topic to display the topic in the view pane.

Parent topic: [Getting to know Workbench](#)

Bookmarks customize the way you use the interface. You may create as many bookmarks as needed.

1. Do one of the following:

- **From the menu bar, select Bookmarks: Manage Bookmarks.**
- **From the popup menu in the Palette side bar, select Manage Bookmarks.**

The **Manage Bookmarks** window opens.

2. Click on one of the following buttons:
 - To create a new folder to organize your bookmarks, click **New Folder**.
 - To create a new bookmark, click **New Bookmark**.
 - To copy a bookmark to another location, click **Copy to**.
 - To move a bookmark to another location, click **Move to**.
 - To delete a bookmark, click **Remove**.
 - To edit a selected bookmark, click **Edit**.
 - To relocate a bookmark in the list of bookmarks, click **Move Up** or **Move Down**.

The framework fulfills your request.

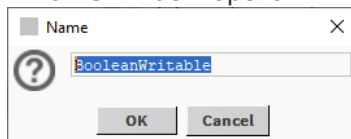
Parent topic: [Getting to know Workbench](#)

The standard components include the control point types, two folders and a text block. You can add one of these to the Nav tree or to a **Property Sheet** or **Wire Sheet**. To add a slot to a container refer to “Adding a slot.”

You are working in Workbench and are connected to a station.

1. Right-click in the container to which to add the new component and click **New**.
The container can be another component in the Nav tree, a **Wire Sheet** view or the **Property Sheet**.
The New command opens a sub-menu.
2. Select the desired type of component:
 - **Folder**
 - **IconFolder**
 - **TextBlock**
 - **BooleanWritable**
 - **NumericWritable**
 - **EnumWritable**
 - **StringWritable**

A **Name** window opens.



3. Type a name for the new component in the text field (or use the default name) and click **OK**.

The new component appears in the in the Nav tree, on the **Wire Sheet** view or on the **Property Sheet**.

4. To delete a any component, right-click the desired component and click **Delete**.

Note: If you delete a component that contains other components, all components in that container component are deleted as well.

Parent topic: [Getting to know Workbench](#)

You can drag a component to a displayed **Property Sheet** by using a right-click drag, which prompts you to choose an action from a menu or by using a normal, left-click and drag.

You are working in Workbench and are connected to a station.

1. Navigate to a target **Property Sheet**.
The **Property Sheet** you choose must be appropriate for the action you intend. The framework displays an illegal symbol if the action is inappropriate.
2. Do one of the following:
 - To use the right-click menu, right-click the component and, depending on what you need, click an action.
 - **Copy**
 - **Copy** makes a duplicate of the component to paste onto the **Property Sheet**.
 - **Paste**
 - **Duplicate**
 - **Delete**
 - **Rename**
 - **Reorder**
 - **Refresh Tree Node**

When you release the right mouse button, a popup menu appears.

- **Left-click and drag the component to the Property Sheet.**
A popup window may open.
3. If a popup window asks for a name, enter the name or accept the default name and click **OK**.
The component appears in the **Property Sheet**.

Parent topic: [Getting to know Workbench](#)

The Nav tree hierarchy can become confusing to navigate especially when the tree includes many levels of parent-child component relationships. The **Go Into** command can restrict the tree to show only the nodes you are currently working on. Then, when you finish your work in one area of the tree, you can return all nodes to the standard, unrestricted view.

You are working in Workbench and are connected to a station.

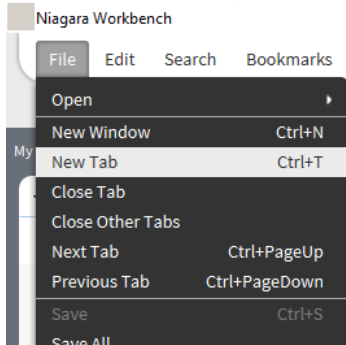
1. To view the Nav tree sidebar, click **Window**, select **Side Bars** and click **Nav**.
2. Click to select the highest node in your hierarchy of interest.
3. Right-click the node and click **Go Into** in the popup menu.
The node you selected becomes the root of the tree and the framework adds the selected node to the **Bookmarks**.
4. To change to a different bookmarked node, click the drop-down list at the top of the Nav sidebar and select the node.
5. To return to the full Nav tree, click the Close Tree icon (.

Parent topic: [Getting to know Workbench](#)

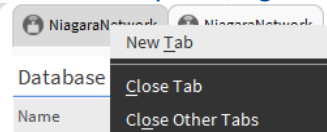
You can use tabs to help organize and selectively display information in the view pane. Workbench allows you to create multiple tabbed views.

1. Open a new tab in the view pane by any of the following methods:

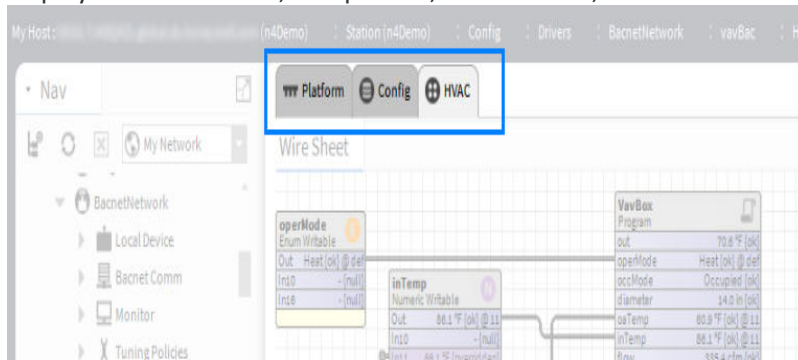
- Use the keystroke combination **Ctrl: Click + t**.
- From the menu bar, select **File > New Tab**.



- If tabs already exist, right-click on a tab and select **New Tab** from the popup menu.



A new tab (identical to the previous one) opens in the view pane. You can use each tab to display a different view, component, or even host, all within the same window.



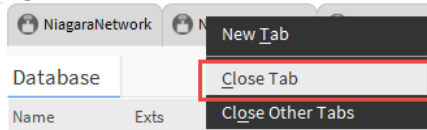
When you select a tab (make it active), the locator bar shows the current path and view. Also, the menu and tool bars update to show appropriate options for the current view. You can move to the next tab by choosing (from menu bar) **File > Next Tab**.

From active tab view, you can copy items, select another tab, and paste them into that view. You can also drag items into an active view from the (Nav or Palette) side bar.

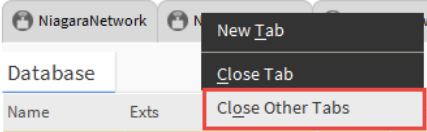
New tab opens.

2. To close the active tab in the view pane, use any of the following methods:
 - Click  (Close icon) located in the upper-right corner of the view, just below the View drop-down list.

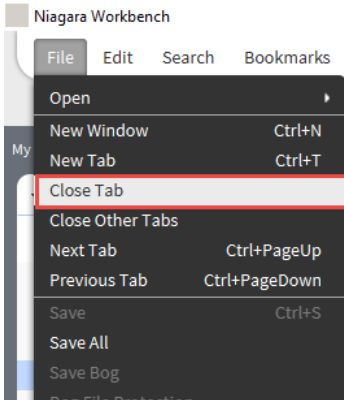
- Right-click a tab and choose **Close Tab** to close that tab.



- Right-click a tab and choose **Close Other Tabs** to close all tabs except that tab.



- From the menu bar, choose **File > Close Tab**.

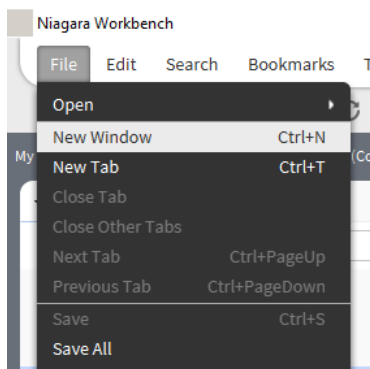


Parent topic: [Getting to know Workbench](#)

You can customize each individual window, as needed, to access different information, allowing you to see multiple concurrent views.

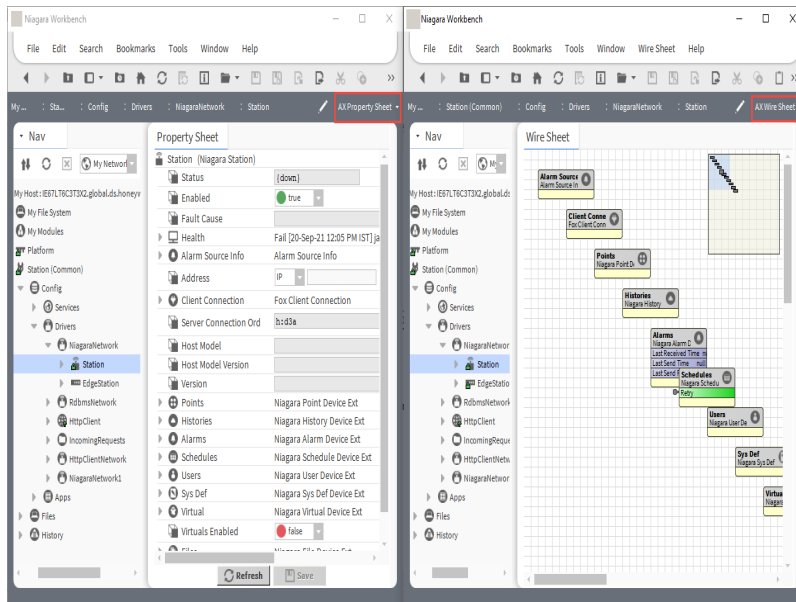
Workbench and a station should be running.

1. To add another window, click **File > New Window** or press Ctrl + N.



This creates a duplicate window.

2. In the additional window change the view by right clicking any node in Nav tree and select a different view from **Views**.



In this example, one window shows a **Property Sheet** and the other shows the **Wire Sheet** that goes with the **Property Sheet** of a station.

3. Repeat this procedure to add additional windows.
You can copy and paste items from one window into another.

Parent topic: [Getting to know Workbench](#)

Editing more than one table row at a time is sometimes called “batch editing,” or “batch processing.”

1. Select the rows in the table to process using the **Ctrl** or **Shift** keys while you click the desired rows.
2. To display the popup menu of available controls and actions, right-click on the selected rows.
3. Select the control or action.
4. If the system opens a window, make your selection and click **OK**.

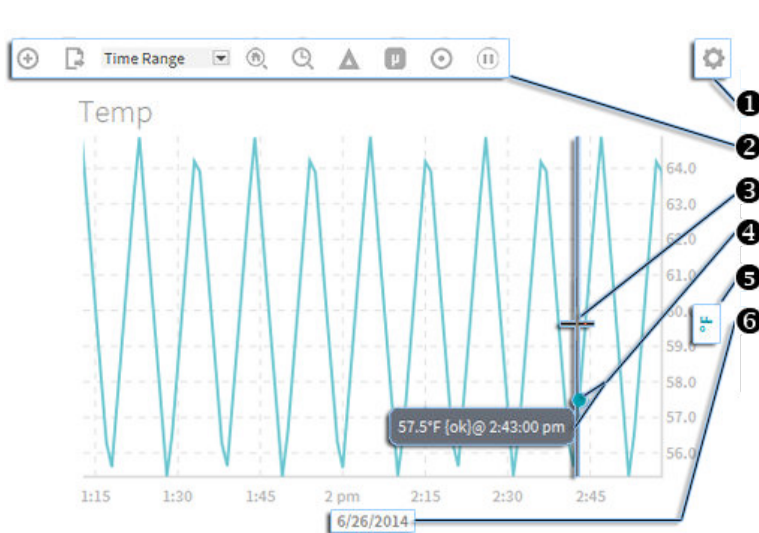
Note: Some actions, such as moving rows, or editing certain types of fields, are not appropriate for batch editing. In these cases – even though you can select multiple rows in the table, the action will be performed on only one (usually the top, or highest) selected record in the table.

Parent topic: [Getting to know Workbench](#)

This view plots historical data, live historical data, and live data, as well as schedules. It is the default history view for history records in Workbench and in Hx, and a secondary view on schedules and Enum, Numeric, and Boolean points. Legacy charts, those created in earlier releases, are available as secondary **History Chart** views on history records.

In Niagara 4 the **Chart** view is the default view for histories. While the **History Chart** view is a secondary view for legacy charts created an earlier release. Although the two views have a different look and feel, both offer many of the same controls and options.

Figure 1. Chart view description



- ① Settings icon — click to access chart **Settings** window
- ② Command bar — click icons to launch chart commands
- ③ Cursor position indicator
- ④ Data Value popup — displays when cursor is on a point
- ⑤ Y-Axis label — default orientation of Y-axis for primary data
- ⑥ X-Axis label — default orientation of X-axis. Once you have defined a specific Time Range for the chart, you can click this label to reopen the **Time Range** window to modify the range.

Export icon in Workbench view:

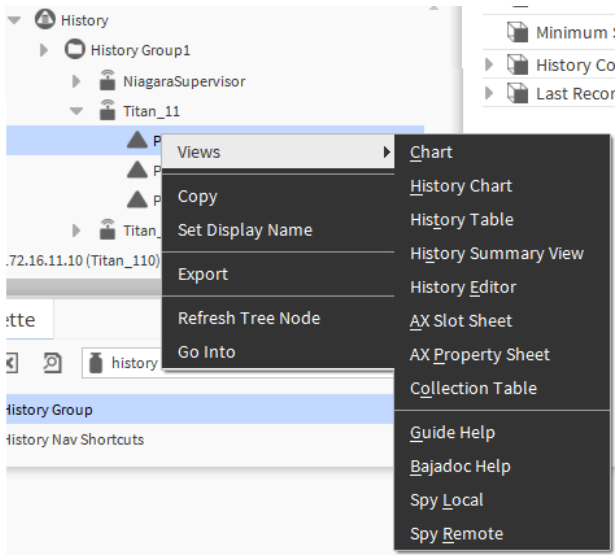


Export icon in browser view:



You can view histories in different ways in Workbench.

Figure 2. History views available from popup menu



The screen capture shows a menu of views that are available using either the Workbench view selector or a view popup menu.

Chart types

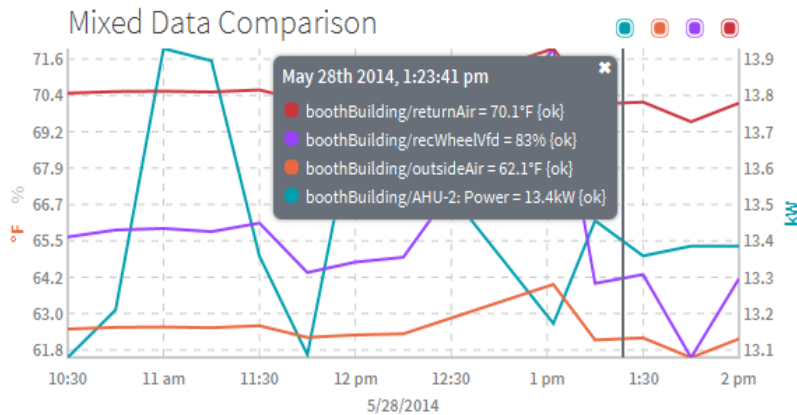
Although you can configure Chart Type via the **Settings > Series** window, the default chart type is determined by the type of data being presented. For example:

Component type	Default chart type
Numeric histories and points	Render as lines with interpolation and display as a line chart.
Numeric schedules	Render as discrete lines with no interpolation and display as a line chart.
Boolean and Enum points	Render as shaded areas referred to as swim lanes and displayed as a shaded chart. The ordinal of the component determines the opacity of the swim-lane fill.
Boolean and Enum schedules	Render as shaded areas referred to as swim lanes and displayed as a shaded chart. The ordinal of the component determines the opacity of the swim-lane fill.

Different types of data (Numeric and Boolean or Enum) can be combined on the same chart. To allow you to more clearly view the lines representing the numeric data, the swim lanes representing Boolean and Enum data display with dimmed opacity. Also, you can modify the default chart type of one or more components in a chart. For example, you can set a Boolean writable point to display bars while the data for another component plots a line.

The interactive Chart view allows you to modify the chart while it is rendering. For example, while viewing, you can add one or more points, history records, schedules and even containers of data. When adding data to a chart, the Y-axis automatically adjusts the units and can accommodate different units of measure by displaying multiple Y-axes.

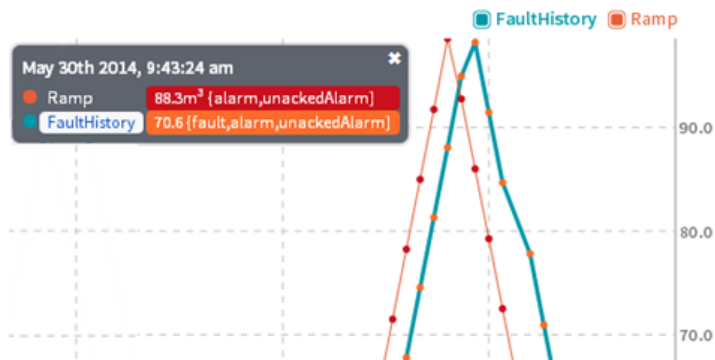
Figure 3. Multiple Y-axes accommodate data with different units of measure



On a chart containing data with three or more different units of measure, such as that shown above, the display still shows dual Y-axes. To switch the units displayed on the secondary Y-axis, click on the dimmed axis label. For example, on the left-side Y-axis, the dimmed % symbol indicates an alternate Y-axis with percent as the unit of measure. Clicking that % symbol switches the Y-axis units from displaying degrees to displaying percentage.

You can hide or show specific data and even completely remove data from a chart. Chart settings can customize the appearance of a chart via selectable data colors, chart type per component, axis orientation, and data source zooming, as well as turning the chart grid on or off, changing the background color, configuring data pop-ups and status colors.

Figure 4. Line chart displaying status colors



Web charts use standard Niagara status colors to indicate current status. The chart above invoked the Status Coloring command, a red dot indicating Alarm status to mark each plot in the **Ramp** line while an orange dot indicating Fault status marks each plot in the **FaultHistory** line. Status colors shown in the **Fixed Data** window confirm the status of charted data.

Shade and Bar charts display status colors. When enabled, and if there is a non-ok status, a color band at the top of the shaded area or bar indicates the status.

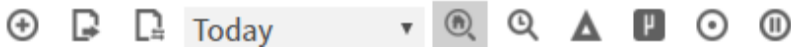
- [Chart commands](#)
Options in the **Chart view Command Bar** allow you to fine tune data presentation.
- [Chart settings](#)
Options in the **Chart view Settings** window allow you to make data presentation changes

that are of a persistent nature, meaning the changes are retained when the chart is exported or saved.

Parent topic: [Getting to know Workbench](#)

Options in the **Chart** view **Command Bar** allow you to fine tune data presentation.

Figure 1. Command Bar

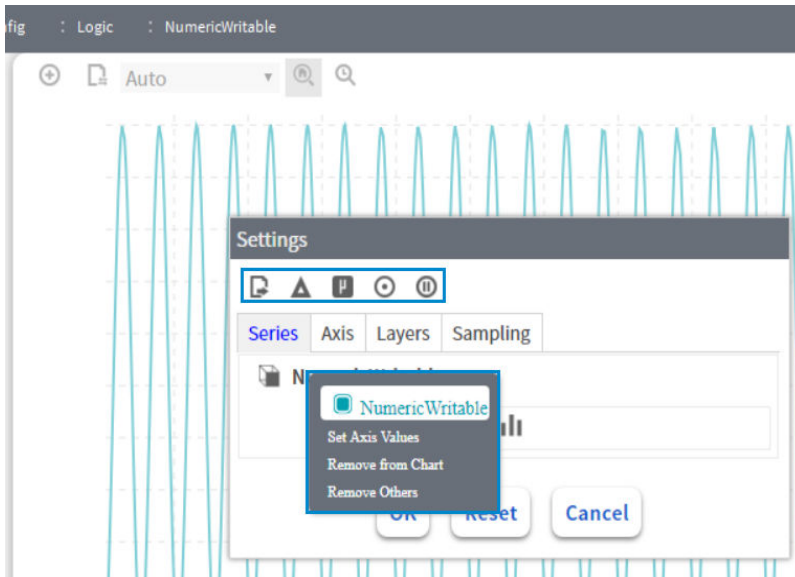


Note: Most options in the **Command Bar** provide fine tuning for viewing purposes only. Changes made with those options are of a temporary nature and are not included when the chart is saved or exported. For example, if you turn on time zoom and Delta using buttons in the command bar and then export the chart, the chart file displays with the original default settings for those options. Exceptions to this are changes made with the Time Range, Sampling, and Status Coloring options, which are included on export or save.

To export the chart and retain all of the changes that you have made, you need to do the following:

1. Export the modified chart to a Station File.
2. Create a **Px** view for this chart and load the exported Station File to this **Px** view. The chart will display with the modifications included.

Figure 2. Narrow chart width changes the chart display (Niagara 4.6 and later)



Any time the chart width is less than 800 pixels the following changes in the chart occur. This prevents the chart from appearing overcrowded which helps maintain legibility. Once the chart window is resized to greater than 800 pixels, the changes revert.

- Chart title and data series legend become hidden.

- Several of the commands icons move from the chart **Command Bar** into the **Settings** window.
- In the **Settings** window, a right-click menu is available on data series in the **Series** window. The right-click menu allows you to hide or show specific data or even completely remove data from a chart.

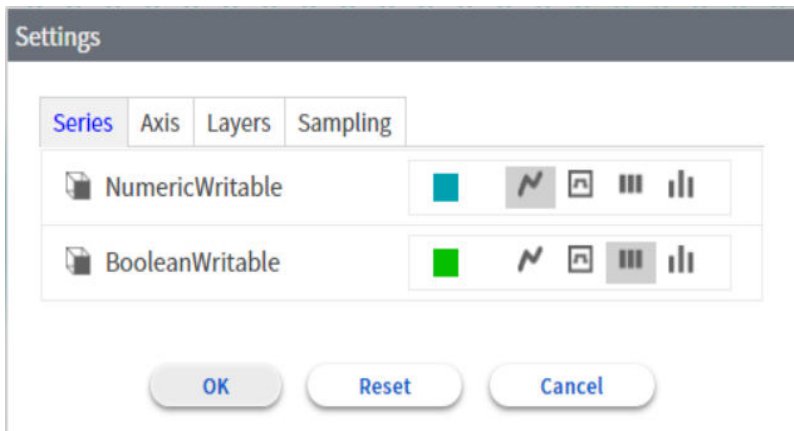
Command Bar	Default	Description
 Add Series	opens separate Nav tree	Adds components to the chart. Select one or more components via the File Chooser. Use Ctrl + Click to select multiple individual components or select a folder that contains multiple components.
 Save Chart	opens save window	Available only when you open an existing .chart file and make changes. Saves the file (chart or csv format) to the station File space (Files/charts/chartName.chart or Files/csv/chartName.csv).
 Export Current View or Object	opens exporting window	Available in a new chart and when you open an existing chart file. • Select Exporter opens a window to choose the exported file type. Options are Chart (default), CSV, and, in a browser connection, Print is also available. • Ord Type selects the type of ORD: Absolute or Relative (default). Note: In Niagara, there is added support for relative ORDs to better accommodate Px page reuse. • Base Ord specifies a base ORD from which to relate all of the ORDs in the series for that chart. This option applies only to chart exports with the Relative Ord Type. • Select Destination provides two options: Download or Station File. Download exports to your Windows file system. Station File exports the file to the station File space (Files/charts/chartName.chart) or (Files/csv/chartName.csv) • • File Name • View On Export
 Time Range Range	defaults to Auto (default)	Opens a window where you can enter custom Start and End times for the range. Leave the End time property blank for live data to continue plotting on the chart. Options are: • Auto (default) • Time Range • Today • Last 24 Hours • Yesterday

Command Bar	Default	Description
		• Week To Date • Last Week • Last 7 Days • Month To Date • Last Month • Year To Date • Last Year
 Toggle Home Zoom	• On (default) • Off	Turns On/Off Home Zoom. On — zooms to the X-axis of the primary data set. Note: If the primary data set is numeric, it zooms on the Y-axis. Off — reverts to Home Zoom.
 Toggle Time Zoom	• On • Off (default)	Turns On/Off Time Zoom. On — zooms X-axis to the time period specified by the Time Range drop-down list. Off — reverts to Home Zoom.
 Toggle Delta Command	• On • Off (default)	Turns On/Off Delta. On — plots the rate of change between points. Off — resumes plotting data points.
 Toggle Sampling Command	• On • Off (default)	Turns On/Off Sampling. On — sampling is enabled Off — turns off sampling and disables auto-sampling behavior.
 Toggle Status Coloring	• On • Off (default)	Turns On/Off data Status Coloring. On — displays data points with status colors in a line chart and in shade or bar chart displays a status color band at the top of each bar. Off — hides status coloring, data points/color bands.
 Toggle Pause	• On • Off (default)	Turns On/Off pause in live data plotting. On — pauses live data plotting. No longer in live mode when paused Off — resumes live data plotting
 Stop	• On • Off (default)	Visible only during data loading. Turns data chunking On/Off. On — stops the data chunking process, halts data coming from the server. While stopped, the button displays a red border. Off — reloads all of the data.

Parent topic: [workbench-WebChart](#)

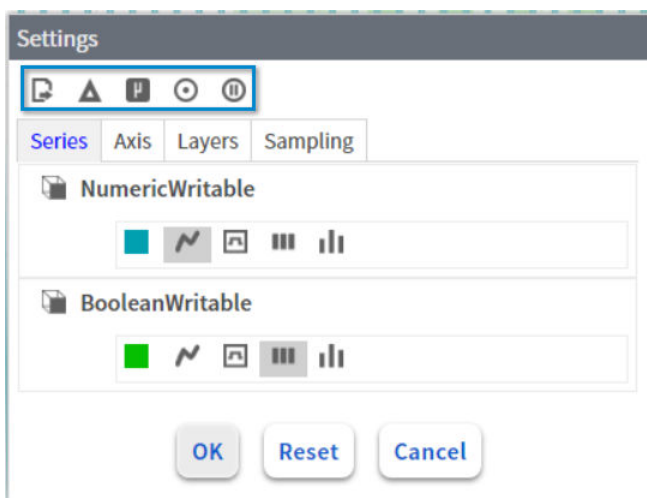
Options in the **Chart** view **Settings** window allow you to make data presentation changes that are of a persistent nature, meaning the changes are retained when the chart is exported or saved.

Figure 1. Settings Window in Workbench



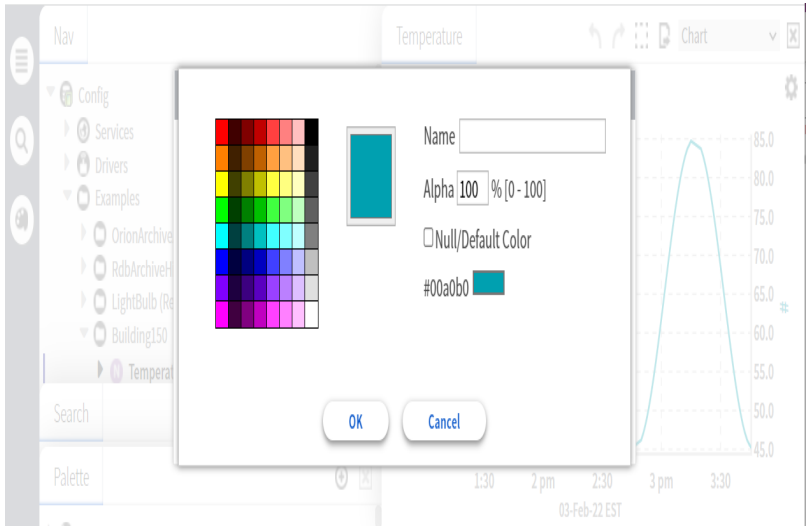
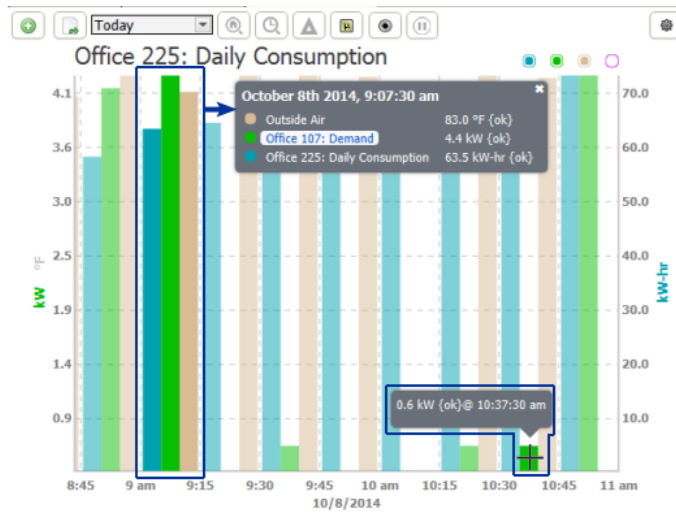
Note: In the Niagara 4.8 Workbench and later, if the chart width is less than 800 pixels several of the chart commands icons are moved into the **Settings** window above the tabs. When the chart is resized wider than 800 pixels, those icons revert back to the Commands Bar in the chart.

Figure 2. Commands icons in the Settings window



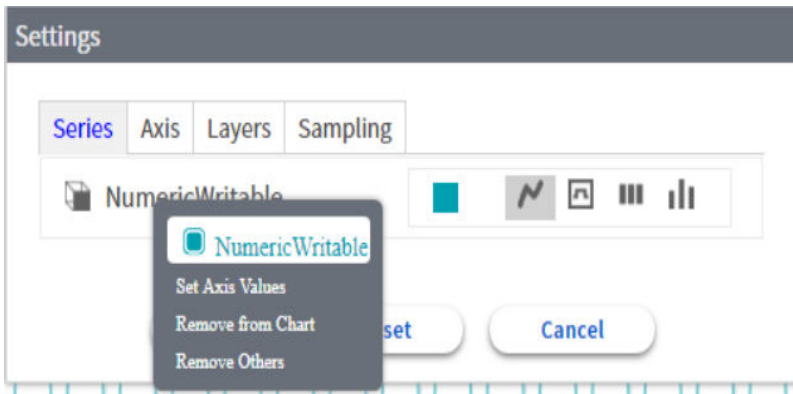
Series tab

Settings	Options	Description
Color	Color block assigned to each component	Change default data color by clicking color block and selecting different color via Color Picker. As of Niagara 4.12, the Color Picker is also available when you are connected to your station via the browser using the HTML5 browser-based

Settings	Options	Description
		implementation of the WebChart view. 
Chart type	• Line, • Discrete line, • Shade, • Bar	Line — plots a smooth line with interpolation. The default chart type for Numeric points and histories. Discrete line — plots a “stepped” line without interpolation. Shade — plots shaded areas, known as “swim lanes,” representing state change. The default chart type for Boolean and Enum points. Bar — plots vertical bars. Samples data into common intervals based on available width. When you have more than one component in a chart using bar chart type, they become a Bar Group, where the individual bars are adjacent (no space between). As shown below, clicking on a Bar Group selects the entire group and the values for all components in the group are shown in the Fixed Data Popup. While the mouseover Data Value Popup, shows the value of a single component. 

A right-click menu is available on data series in the **Series** tab. The right-click menu allows you to hide or show specific data or even completely remove data from a chart.

Figure 3. Right-click menu options



Axis tab

Settings	Options	Description
Y-Axis Orientation	- left - right (default)	Aligns Y-axis of primary data to either the left or right side of the chart.
Data Zoom Scope	- primary (default) - all	Sets the Data Zoom Scope to primary or all. Primary — zooms to the X-axis of the primary data set only. If the primary data set is numeric, it zooms on the Y-axis. All — changes the X-axis to accommodate all available data, including live data as it is recorded.
Show Grid	- true (default) - false	Turns on/off the chart grid. true — the grid displays in chart behind data. false — the grid does not display.
Background Color	- On - Off (default)	Turns on/off the background area color for the current theme. On — the background area color displays in chart behind data. Off — the background area color does not display.
Chart Cursor	- Crosshair (default) - None	Sets the appearance of mouse pointer while positioned over a chart. Crosshair — the mouse pointer appears as a crosshair. None — turns off the mouse pointer visibility (while positioned over a chart), hiding it completely.
Facets Limit Mode	- Off (default) - Inclusive - Locked	Configures whether the WebChart uses a point's facets for Min and Max. Off (default) — the WebChart ignores a point's facets for Min and Max. Inclusive — the WebChart includes a point's facets for Min and Max. Locked — forces the WebChart to use a point's facets for Min and Max.

Settings	Options	Description
		Note: In each of these settings chartMin and chartMax facet keys can be used as a higher priority substitute to “min” and “max”. Even if the Facet Limit Mode is “Off” it can be overridden for specific series if a facet key of chartLimitMode is supplied with the corresponding values of “Inclusive” or “Locked”. Note: Previously, if you were not using a chart file to load a WebChart, there was no way to preset any options. In latest version of Niagara, there is a Default Options WebProperty on a Px page which you can modify to preset WebChart default options. By default, modifications are saved to file: ^charts/defaultOptions.chart. Even when not on a px page, non-chart files will load their options from this file if it exists and the user has permissions to view it. This includes the ability to change all options, so even the default time range can be preset.
Show Start Trend Gaps	• Yes (default) • No	Configures the behavior when drawing the chart line, providing a visual indication (a line gap) of an interruption in data collection. For example, a station restart or that history collection was disabled and re-enabled. Yes — if there is a start trend flag on a record the chart does not connect the dot for that record to the previous record, resulting in a gap in the line No — allows the dots to be connected, eliminating any such gaps.
Show Data Gaps	• Yes • No (default)	Configures the behavior when drawing the chart line, it providing a visual indication (a line gap) for records that have either the hidden flag set or invalid values (+inf, -inf, NaN). Yes — if a record has a hidden flag set or invalid values (+inf, -inf, NaN) the record's dots are not connected to adjacent records. No — if a record has a hidden flag set or invalid values (+inf, -inf, NaN) the record's dots are connected to adjacent records.

Layers tab

Settings	Options	Description
Data Popup	• On (default) Displays • OffPauses	Enables/disables the Fixed Data popup. On — clicking on chart data displays the recorded date and time, as well as the name, value and status for each component in the chart at the point where you click. The persistent data popup remains visible until you close it. Off — suspends display of fixed data popup.
Data Mouseover	• On (default) • Off	Enables/disables the mouseover Data Value popup. On — mouse position on chart data displays the recorded component value, status, and the time for that mouse position. Off — suspends display of mouseover data value popup.

Settings	Options	Description
Status Coloring	- On - Off (default)	Turns On/Off data status coloring. On — displays data points with status colors in a line chart and in a bar chart displays a status color band at the top of each bar. Off — hides status color data points/color bands in the chart.

Sampling tab

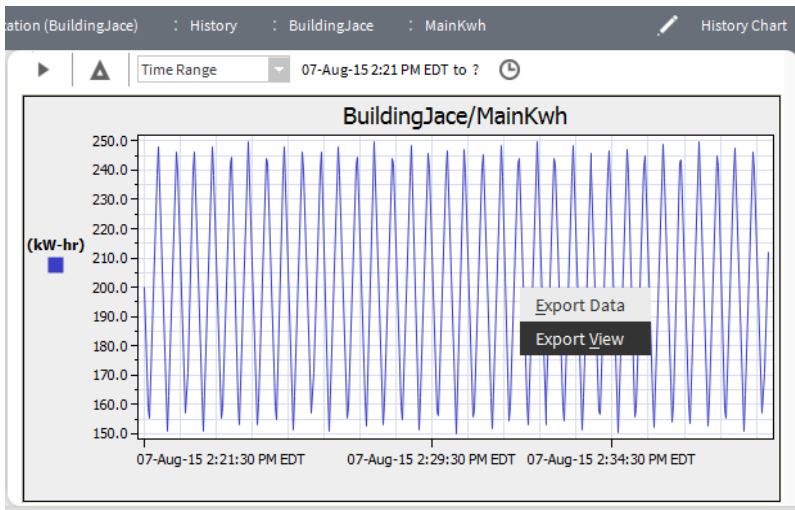
Settings	Options	Description
Auto Sampling	- true (default) - false	Enables/disables automatic sampling optimizations. true — automatically begins sampling if the focused data set exceeds 2500. false — automatically stops sampling if the focused data set is below 2500.
Sampling Type	- Average (default) - Min - Max - Sum	Sets the Sampling type. Average - samples average values for the selected rollup period. Min - samples minimum values for the selected rollup period. Max - samples maximum values for the selected rollup period. Sum - samples the total of the values in the selected rollup period.
Desired Period	- Best Fit (default) 1 Minute 15 Minutes 30 Minutes - Hour - Day - Week - Month - Year - Custom	Configurable setting allows you to choose the desired sampling interval. By default, set to Best Fit which finds the best sampling period that fits the page that is one the standard collection intervals which are: Year, Month, Day, Hour, 30 minutes, 15 minutes, 1 minute, and other smaller common intervals.
Sample Size	2500 (default)	Specifies the number of points in the data set to sample. Range is 1–50000. Note: The default auto sampling size is configurable in the system.properties file.
Sampling	- true - false (default)	Enables/disables sampling for any size data set. true — turns on sampling false — turns off sampling Note: For performance reasons, sampling cannot be turned off once the focused data set exceeds 50,000. This threshold is configurable in the system.properties file.

Settings	Options	Description
Data Points	Read only	Displays the maximum number of points in the data set that are available to sample.
Sampling Period	Read only	Visible only once sampling has begun, displays the calculated average of the amount of time between each of the points that have been sampled.

Parent topic: [workbench-WebChart](#)

The History Chart view uses one or more of the controls and options described in this topic.

Figure 1. History Chart view controls and options



To view a history chart, expand **Station > History**, right-click the chart name and click **Views > History Chart**.

- The Live Updates icon (▶) starts on-demand history plotting. The icon changes to a pause icon while live history plotting is active.
- The Delta icon (Δ) configures the chart to display value changes (delta).
- The Time Range drop-down list Time range option list has a variety of predefined time range options, including an option to restrict your data presentation to a particular date and time range that you specify.
- The from and to dates display to the right of the drop-down list.
- The clock icon (🕒) opens an **Edit Time Range** window for editing the End Time.
- The chart title displays the name of the chart and identifies the point. You edit this name in the **Chart Builder** view.
- The Y axis (vertical axis) displays units.

- The X axis (horizontal axis) displays time
- Charted values display in the body of the chart You can edit the color and type of line in the **Chart Builder** view.
- When you hover the mouse pointer over a point in the history chart, a tool tip displays the date, time, and value of that location in the chart. The values are defined by chart axes and not the values of the actual data points.

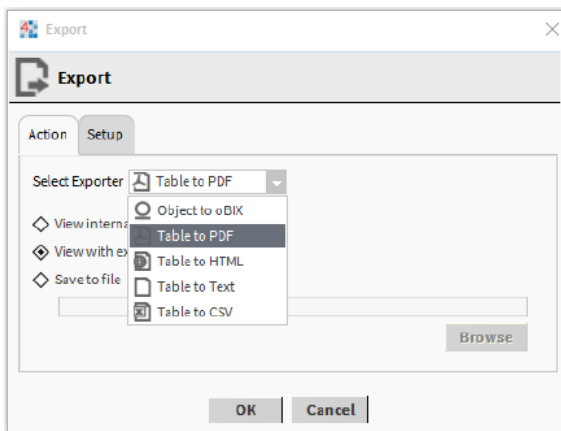
Export options

Right-clicking on the History Chart invokes a short menu with two options: **Export Data** and **Export View**. Both open an **Export** window. The difference between the windows is in the Select Exporter drop-down list.

Clicking **Export Data** opens the **Export** window with table export options.

Action tab

Figure 2. Export Data options

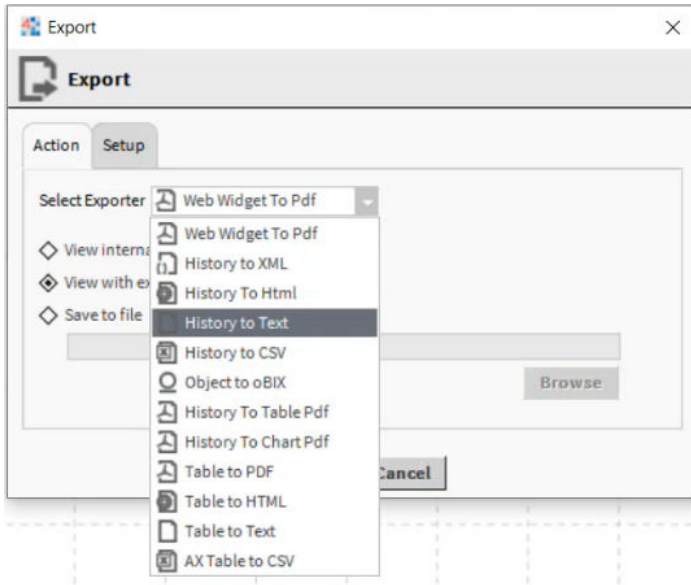


Export Data exports the table to an oBIX, PDF, HTML, Text, or CSV (comma separated variable) file.

Note: This export function works only with charts configured with a single history. Multiple histories do not export to table-type data in a usable format.

Clicking **Export View** opens the **Export** window with chart export options.

Figure 3. Export view options



Export View exports the chart view to an oBIX, PDF, Text, HTML, or CSV (comma separated variable) file.

Setup tab

As of Niagara 4.14, on the **Setup** tab, you can specify the export Time Range and Delta mode to use for the export of history data to the following outputs:

- History to Table Pdf
- History to Chart Pdf
- History to CSV
- History to Html
- History to Text

When you export the data while viewing a history, the Time Range and Delta export settings are pre-filled to match the settings configured in the history view (if applicable). You have the option to change the settings before continuing with the export.

Figure 4. Example of “History to Text” setup in Workbench view

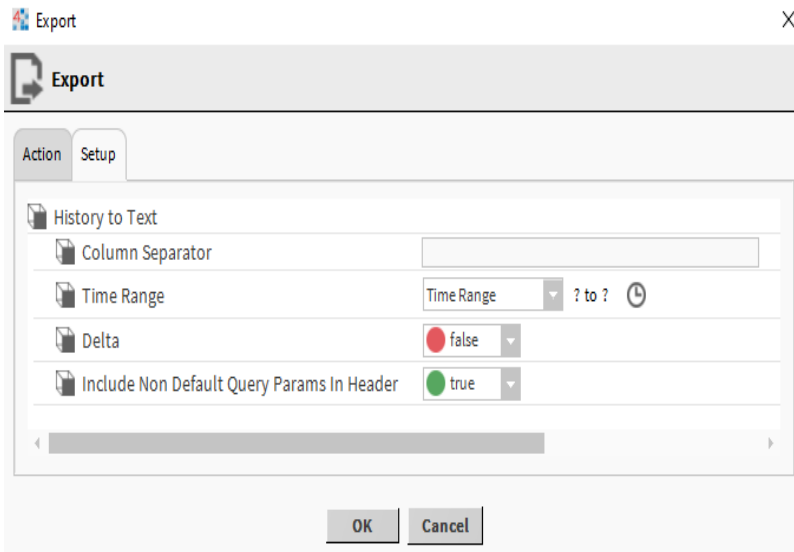
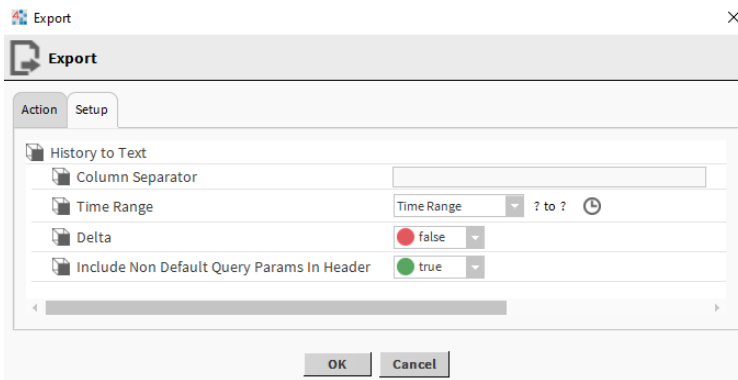


Figure 5. Example of “History to Text” setup in browser view



The Include Non Default Query Params In Header property (defaults to true) applies only to the following exports:

- History to CSV
- History to Html
- History to Text

It decides whether the exported file should include any non-default time range or delta indications in the header of the exported data.

Even with Include Non Default Query Params In Header set to true, if the Time Range is set to default, it will not be included, and if Delta is set to false, it will not be included in the header. For the History to Html export, the indication is included in the title instead of the header. You can hover over the browser tab to see it in the title.

Note: The Include Non Default Query Params In Header property does not apply to the PDF exports since it automatically decides when to include the extra information in the header and

when not. When Delta is set to true and the header is expected, you will see "Delta Values Displayed" in the exported data header/title.

Parent topic: [Getting to know Workbench](#)

Workbench provides a complete user interface to the Niagara Framework. It can directly access a station and perform any action that the system supports. You can view and control any component using any of the supported views.

Common Workbench tasks include the following:

- Creating a new station
- Editing components
- Managing alarms
- Creating a Live History Chart
- Working with modules
- [Creating a new station](#)
Use the **New Station** tool from the **Tools** menu to create a new controller or Supervisor station. The new station is automatically configured with appropriate services.
- [Creating a new Linux Supervisor station](#)
Niagara provides a Linux Supervisor station template. This procedure describes the steps to create a new Linux Supervisor station.
- [Creating a station template](#)
With a station template that contains everything needed for the initial starting point of a new station, you can configure multiple stations in a single step. This procedure is one the Systems Integrator might perform on a fully configured basic station for purposes of reuse and standardization.
- [Managing alarms](#)
You can configure how alarms appear in the **Alarm Console**, acknowledge alarms and manage the database of alarms. This topic summarizes in steps the basic ways to manage alarms.
- [Backing up a station using the BackupService](#)
You may use this procedure to manually back up a single Supervisor or controller station. The *Niagara Provisioning Guide* explains how to set up a job prototype to back up all stations on a network.
- [Restoring a controller station from a distribution file using Workbench](#)
A software upgrade may require that you restore the controller station from a backup .dist file. This procedure works for a controller station only. You cannot restore a Supervisor station from a backup .dist file.
- [Signing an unsigned module](#)
Using only signed modules provides some assurance that the code came from a trusted source, reduces the risk of installing malicious code, and complies with module signing requirements. For example, in Niagara if you attempt to install an unsigned module it will cause a signature verification warning.
- [Converting NDIO modules to NRIO](#)
Use the **NDIO to NRIO Conversion Tool** to easily convert NDIO proxy extensions to NRIO proxy extensions. This is necessary when replacing a legacy JACE with the JACE-8000 or JACE-9000, which do not support NDIO.
- [Local Workbench logs](#)
Use the **Logger Configuration** tool to modify log settings for local Workbench logs. For example, you can add and remove log categories. To collect more or less data, you may change the severity level for an existing log category.
- [Viewing logged data](#)
You can view logged data for stations and for Workbench.

- [Viewing the station log](#)

You can review the station log to check the log information for the connected station and to view the recently recorded station. This procedure documents how to view the station log in **Application Director** and **LogHistoryService**.

Use the **New Station** tool from the **Tools** menu to create a new controller or Supervisor station. The new station is automatically configured with appropriate services.

You are connected to the platform that will host the station.

If the platform that you are running is licensed for FIPS, the **New Station** tool creates a FIPS-compliant station.

1. In Workbench, select **Tools > New Station**. The **New Station Wizard** opens.

Name	Vendor	Version	Description
NewControllerStation.ntpl	Tridium	1.5	
NewJACEProvisioningStation.ntpl	Tridium	1.3	
NewSupervisorStationLinux.ntpl	Tridium	1.7	
NewSupervisorStationWindows.ntpl	Tridium	1.7	

2. Enter the Station Name.
The Station Name property is case-sensitive and must begin with a letter. Best practice is to keep station names short and use a station display name if a longer name with spaces or other characters is required.
To create a directory of the same name, the system automatically appends the Station Name text you enter to the read-only Station Directory property.

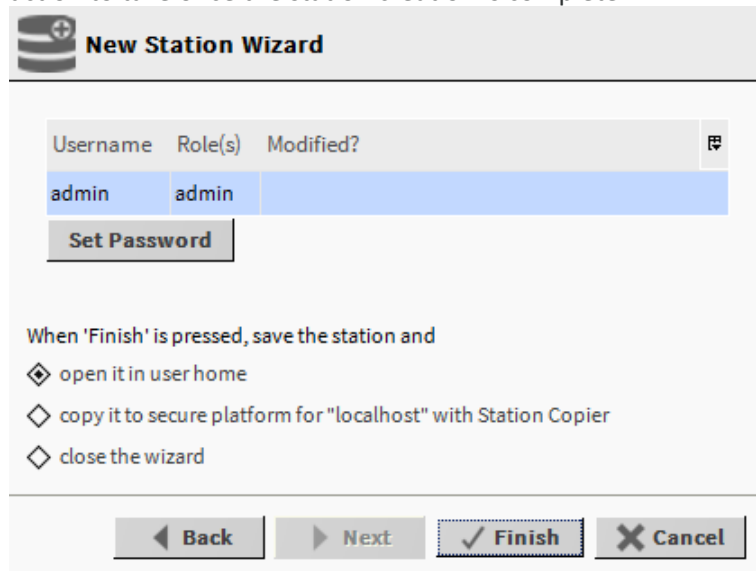
If you enter a duplicate station name, the system prompts you to delete the existing station, as shown below.

Either delete the existing station or enter a different station name.

3. Select a Station Template type and click **Next**.

The **Station Templates** table contains the default new station templates provided in Workbench as well as any user-defined templates.

A second window opens to set the admin user password and prompts you choose an action to take once the station creation is complete.



Username	Role(s)	Modified?
admin	admin	

Set Password

When 'Finish' is pressed, save the station and

- ☒ open it in user home
- ☐ copy it to secure platform for "localhost" with Station Copier
- ☐ close the wizard

Back **Next** **Finish** **Cancel**

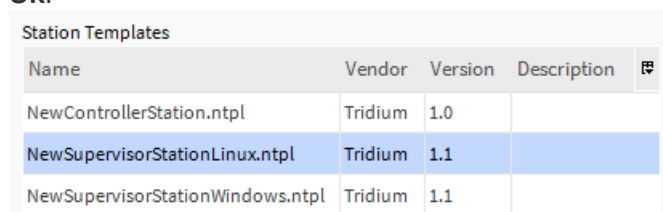
- To enter the station's password, click **Set Password**. The **Set Password** window opens.
- Enter a password for the station and click **OK**. Your password must contain at least 10 characters, one digit, one lowercase character and one uppercase character. If the new station template selected earlier contains any exposed properties, such as the ports, etc., this window presents those properties for you to configure.
- Select an option for the preferred action-on-completion and click **Finish**. The **New Station Wizard** closes. If the default option, open it in user home is selected, a **Property Sheet** view of the new station config.bog file opens.

Parent topic: [Key tasks](#)

Niagara provides a Linux Supervisor station template. This procedure describes the steps to create a new Linux Supervisor station.

The latest version of Niagara is installed.

- In Workbench, click **Tools > New Station**.
- In the **New Station** wizard, click the **NewSupervisorStationLinux.ntpl** template and click **OK**.



Name	Vendor	Version	Description
NewControllerStation.ntpl	Tridium	1.0	
NewSupervisorStationLinux.ntpl	Tridium	1.1	
NewSupervisorStationWindows.ntpl	Tridium	1.1	

The wizard walks you through the process of creating the station.

- In the new station's **Services** node, expand **WebServices** and confirm configuration of the following port settings:
 - HTTP = 8080
 - HTTPS = 8443
- In your network's firewall, set up rules to do the following:

- redirect HTTP port 80 (external) to port 8080 (internal)
- redirect HTTPS port 443 (external) to port 8443 (internal)

This makes it appear to a remote client that the web server is running on port 80, but the framework redirects all requests sent to port 80 to port 8080 where the web server is listening for them. Similarly, the framework redirects requests sent to port 443 to port 8443.

Parent topic: [Key tasks](#)

With a station template that contains everything needed for the initial starting point of a new station, you can configure multiple stations in a single step. This procedure is one the Systems Integrator might perform on a fully configured basic station for purposes of reuse and standardization.

An existing, fully-configured station suitable for use in a generic new station template exists.

1. To open the **Template View**, right-click on the station's **Config** node in the Nav tree, and select **Make Station Template**.

The **Template View** opens.

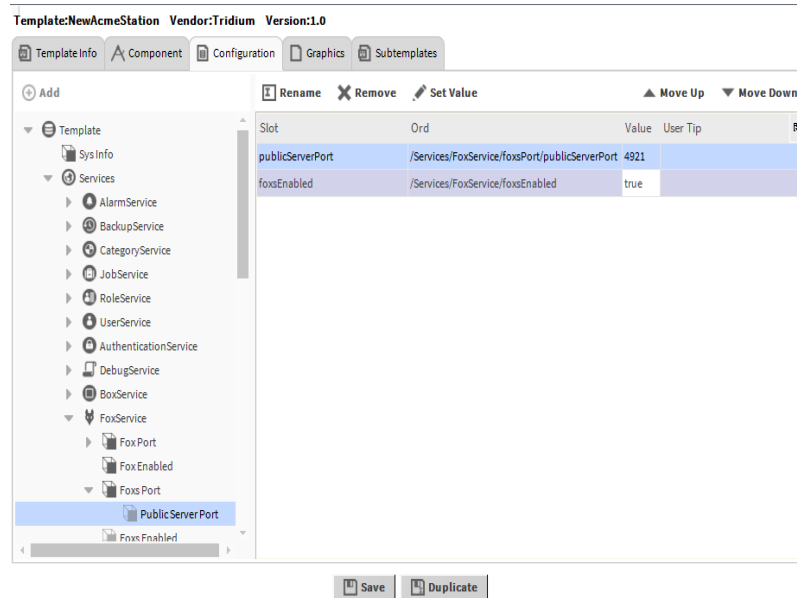
The screenshot shows the 'Template View' window for a new station template. The title bar reads 'Template:NewAcmeStation Vendor:Tridium Version:1.0'. Below the title bar are five tabs: 'Template Info' (selected), 'Component', 'Configuration', 'Graphics', and 'Subtemplates'. The 'Template Info' tab contains the following fields:

- Filename:** NewAcmeStation
- Title:** NewAcmeStation
- Vendor:** Tridium
- Version:** 1.0
- State:** Draft (dropdown menu)
- Description:** Acme new station template (text area with content: 'Use this template to create generic Acme station')
- Info:** (empty text area)
- Icon:** NO ICON SELECTED (with a folder icon button)

At the bottom of the window are two buttons: 'Save' and 'Duplicate'.

2. Click each of the tabs to configure the station template as needed.

For example, to set up the template to configure the Foxs Port:, click the **Configuration** tab.



a. In the left pane of the **Configuration** tab, expand **Services > FoxService > Foxs Port**.

b. Double-click Public Server Port.

The system adds it to the right pane.

c. Change the default value as needed and click **Set Value**.

The value entered into the station template for the exposed property becomes the default value upon creating a new station. Any exposed component properties defined in the template appear as configurable parameters in the **New Station Wizard** when this template is used.

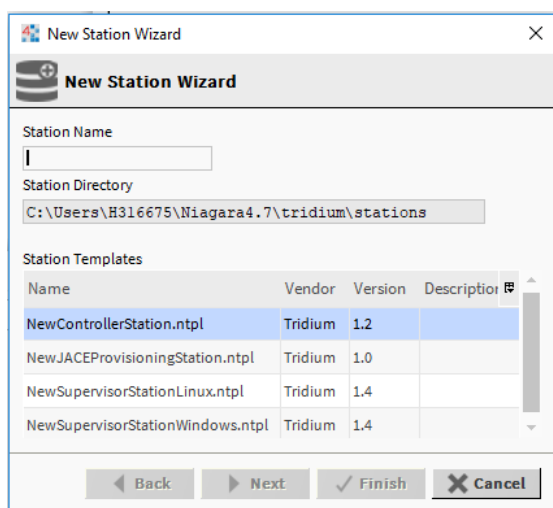
d. Click **Rename** and change the name to Foxs Port.

3. On the **Subtemplates** tab, review and/or modify any deployed templates contained by the station, which are now sub-templates of the station template you are creating.

4. When finished modifying the template, click **Save**.

The new station template is saved in the ~stationTemplates sub-directory of your Workbench User Home.

Also, the station template is immediately available for inclusion in the Workbench **New Station Wizard**, where your template appears as a selectable option in the **Station Templates** table as shown here:



Note: It is not necessary to restart Workbench in order for the user-defined station template to appear in the **New Station** wizard. The file simply needs to be saved to the `~stationTemplates` directory.

More details on making a template are available in the *Niagara Templates Guide*.

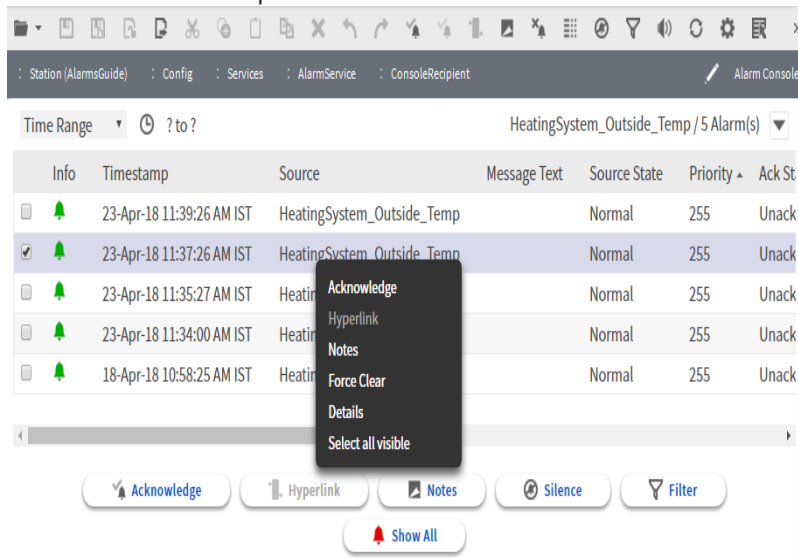
Parent topic: [Key tasks](#)

You can configure how alarms appear in the **Alarm Console**, acknowledge alarms and manage the database of alarms. This topic summarizes in steps the basic ways to manage alarms.

You are working in Workbench and are connected to a station.

1. To set up an alarm recipient, which is required to view the **Alarm Console**, open the alarm palette, expand **Recipients** and add a **ConsoleRecipient** to your station.
2. To display the **Alarm Console**, right-click the **ConsoleRecipient** and click **Views > Alarm Console**.

The **Alarm Console** opens.



Each record that appears in the alarm console table represents one alarm source and one or more alarms from that source.

- To silence an alarm, select it and click the **Silence** button at the bottom of the view.
- To sort the alarms in order of any column, click the column title (once for ascending, twice for descending).
- To view details about the selected alarm, double-click the alarm row in the table, select the alarm and click **Hyperlink** or right-click and select **Details** from the menu.
- To acknowledge one or more of the alarms, select one or more and do one of the following:

You may select multiple alarms using the **Shift** or **Ctrl** keys.

- To acknowledge the latest (most recent) alarm(s), right-click the selected alarm(s) and click **Alarm Acknowledge Most Recent** or click **Alarm > Acknowledge** from the menu.
- To acknowledge all alarms that are reported from a single source, click **Alarm > Acknowledge All** or click the **Acknowledge All** button at the bottom of the console.

The system acknowledges the latest or all alarms from each selected alarm source.

- To manage the alarm database, double-click the **Alarm** node under **Station** in the Nav tree.

The **Database Maintenance** view opens.

More information about managing alarms is in the *Niagara Alarms Guide*

Parent topic: [Key tasks](#)

You may use this procedure to manually back up a single Supervisor or controller station. The *Niagara Provisioning Guide* explains how to set up a job prototype to back up all stations on a network.

You are using Workbench running on a PC and are connected to the remote station.

- Expand **Config > Services** and double-click the **BackupService**.
The **Backup Manager** view opens.
- To back up the station, select the station and click **Backup**.

A **File Chooser** window opens with a default location and name for the backup. The default ~backups folder is in the user home: C:\Users\

- <user name> is the name that identifies your PC
- <framework version> identifies the current version of the Niagara Framework.

3. To change the backup location, use the icons at the top of this window or accept the default and click **Save**.

The **BackupService** creates a .dist (distribution) file of the station in the folder you selected.

Parent topic: [Key tasks](#)

A software upgrade may require that you restore the controller station from a backup .dist file. This procedure works for a controller station only. You cannot restore a Supervisor station from a backup .dist file.

A recent distribution (.dist) of the station exists.

1. Open Workbench in the newly-installed build.
2. In the Nav tree, right-click on the **Platform** node and choose **Views > Distribution File Installer**.
The Distribution File Installer defaults to the C:/Niagara/<framework version>/cleanDist folder (where <framework version> represents the version of software).
3. Use the buttons at the bottom of the view to initiate a restoration of your backed-up controller station to the newly installed station directory.
If you just upgraded the controller software and chose to install the platform daemon during the software installation process, your existing station may not start automatically with a new platform daemon.
4. Maintaining your platform connection, change views to the Application Director view and select the newly restored station.
5. To configure auto-start, enable the Auto-Start property.
6. To start the station, click the **Start** button.
The station is now ready. A Supervisor station should communicate successfully with any subordinate stations, which should operate normally.
7. To log in to the station, double-click the station entry.

Parent topic: [Key tasks](#)

Using only signed modules provides some assurance that the code came from a trusted source, reduces the risk of installing malicious code, and complies with module signing requirements. For example, in Niagara if you attempt to install an unsigned module it will cause a signature verification warning.

- You are running latest version of Niagara Workbench.
- You have already created a code signing certificate. For details, see *Niagara Station Security Guide*).

1. In Workbench, click **Tools > Jar Signer Tool** to open the **Jar Signer** window.
2. Beside the field labeled Select a jar to sign, click the folder icon to the right.

3. In the file chooser window navigate to the unsigned jar file and click to select it, and click **OK**.
4. In the Alias field, click and select your code signing certificate.
5. Enter the Certificate Password.
6. Enter a TSA URL (or use the default URL) and click **OK**.
The **Save signed jar as** window opens prompting you to select a destination location for the signed jar.

Note: You can save the file anywhere, but if you intend to use it in the current installation, it has to eventually go into the !modules folder.

7. Navigate to the desired location and click **Save**.

The signed jar is saved to the specified location. A confirmation window indicates signing is complete.

Parent topic: [Key tasks](#)

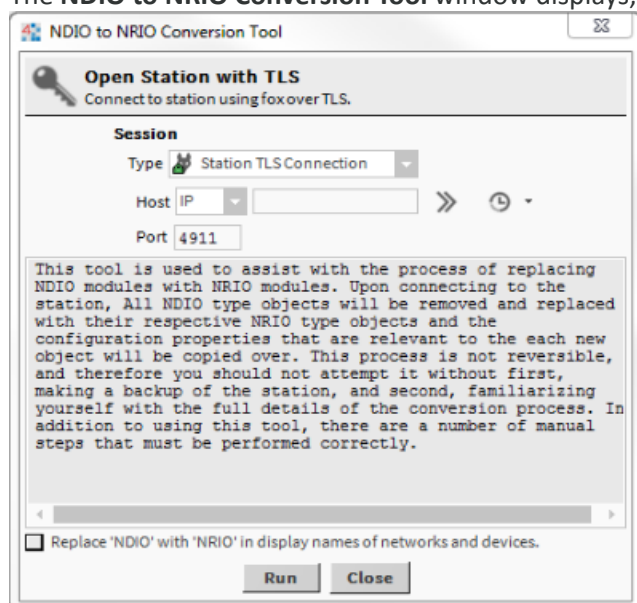
Use the **NDIO to NRIO Conversion Tool** to easily convert NDIO proxy extensions to NRIO proxy extensions. This is necessary when replacing a legacy JACE with the JACE-8000 or JACE-9000, which do not support NDIO.

- The nrioConversion-wb module is installed on machine running Workbench.
- Updated version of nrio-rt module installed on remote platform
- Existing station is running on Niagara 4 (regardless of whether or not the host supports NDIO). The station has an NDIO network with points.
- Credentials for a Super User on the target station.

CAUTION: This process is not reversible, therefore you should not attempt it without first making a backup of the station and familiarizing yourself with the full details of the conversion process.

1. Back up the controller station that contains the NDIO module(s) to be converted.
2. Confirm that both the NRIO and NDIO modules are installed on the controller.
3. Click **Tools > NDIO to NRIO Conversion Tool**.

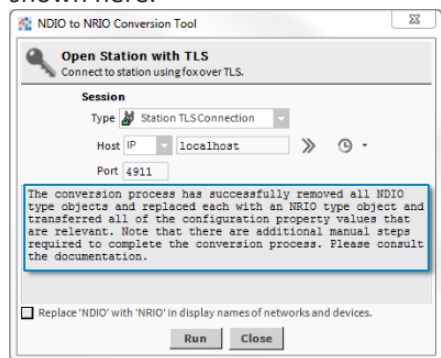
The **NDIO to NRIO Conversion Tool** window displays, as shown here.



4. Enter the following connection values:
 - Host: <IP address>
 - Change the connection type to TLS, if applicable.
 - If desired, click the checkbox labeled: **Replace 'NDIO' with 'NRIO' in display names of Ndio networks and devices.** This affects only display names, not slot names.
5. Click **Run**.
 - If already connected to the station you are not prompted for credentials.
 - If not connected, an **Authentication** window displays. Enter your credentials and click **OK** to continue.

Note: The conversion requires you to provide Super User credentials for the remote station. If you login as a non-Super User, you will receive an error message. In this case, you must logout and log back in as a Super User.

On completion, the window displays notification that the conversion was successful, as shown here.



6. Click **Close**.
 7. Check the newly created NRIO Network(s) for proper conversion.
-

Note: If an NDIO device has more points than can be accommodated by an NRIO-16 module (UI>8, AO>4, DO>4), it will be migrated to an NRIO-34 module. Otherwise, it will be migrated to an NRIO-16 module.

8. Open the **NRIO Device Manager** for one of the new NRIO Networks. Use the **Discover** and **Match** functions to match the converted NRIO devices to the connected ones. Repeat this step for each network.

The conversion process is now complete.

Parent topic: [Key tasks](#)

Use the **Logger Configuration** tool to modify log settings for local Workbench logs. For example, you can add and remove log categories. To collect more or less data, you may change the severity level for an existing log category.

Note: The procedures to configure station logs are identical except that, rather than access the **Logger Configuration** view using **Tools > Logger Configuration**, you access this view via the **DebugService** in the connected station.

- [Adding a new log category](#)
- [Removing an existing log category](#)
- [Changing the severity level of an existing log category](#)

Parent topic: [Key tasks](#)

You can view logged data for stations and for Workbench.

Viewing logged data for the station

To view logged data for the station perform either of the following:

- For a station connection in Workbench, open the platform **Application Director** view.
- For a browser connection to the station, view stdout on the **Spy** menu.

Viewing logged data for Workbench

To view logged data for Workbench:

- Open the console that started with Workbench, the logs for Workbench are visible in the Niagara console window.

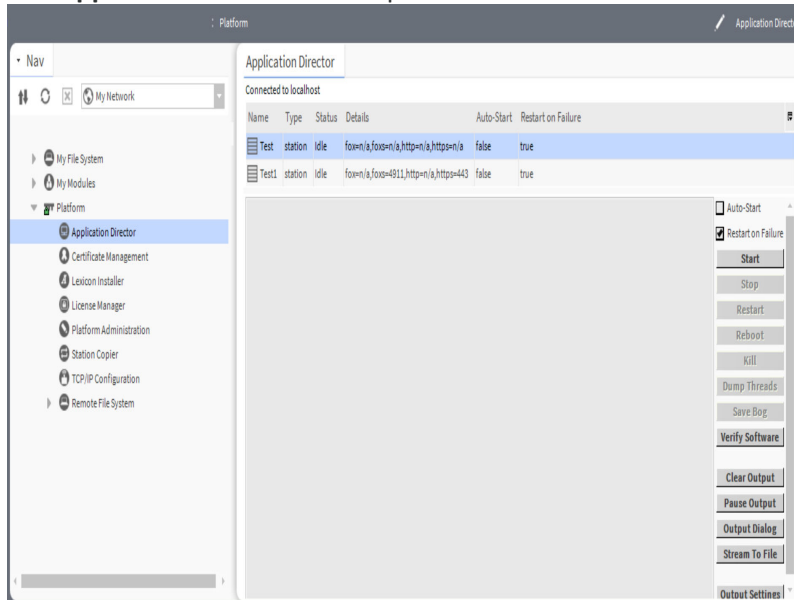
Parent topic: [Key tasks](#)

You can review the station log to check the log information for the connected station and to view the recently recorded station. This procedure documents how to view the station log in **Application Director** and **LogHistoryService**.

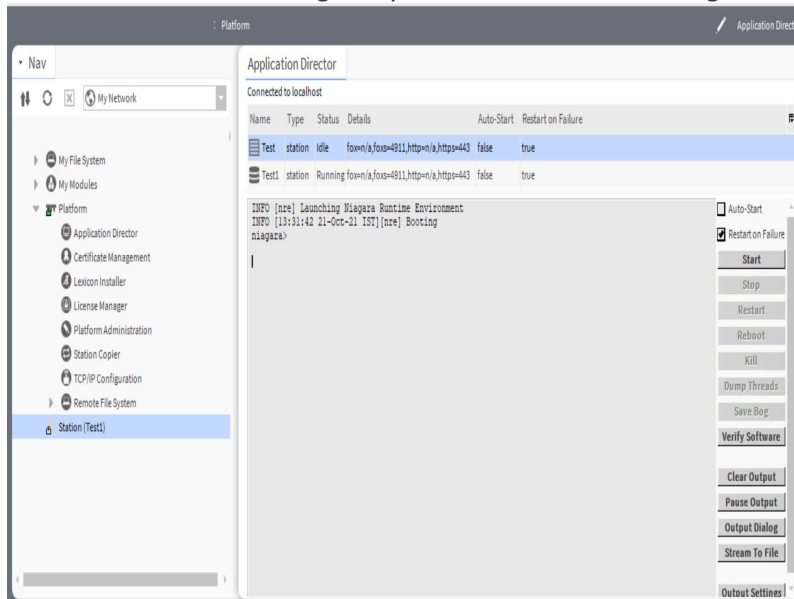
You are connected to a platform on a Workbench.

1. You can view the station log in the below choices.
 - [Application Director](#)
 - [LogHistoryService](#)
2. To view the station log, expand **Platform** and double-click **Application Director** in the Nav tree.

The **Application Director** view opens with a list of stations.

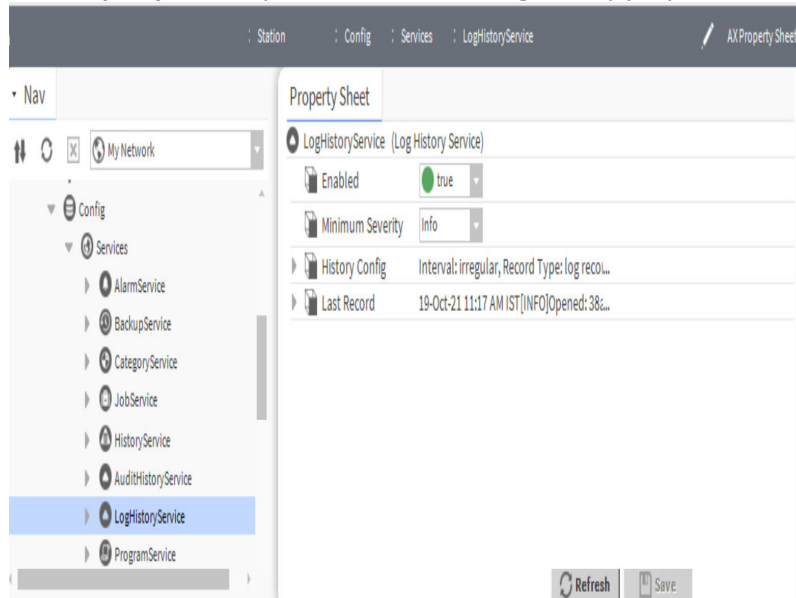


3. Select the desired station and click **Start** from the right side of the station table. The station started running and you can view the station logs in the standard output.

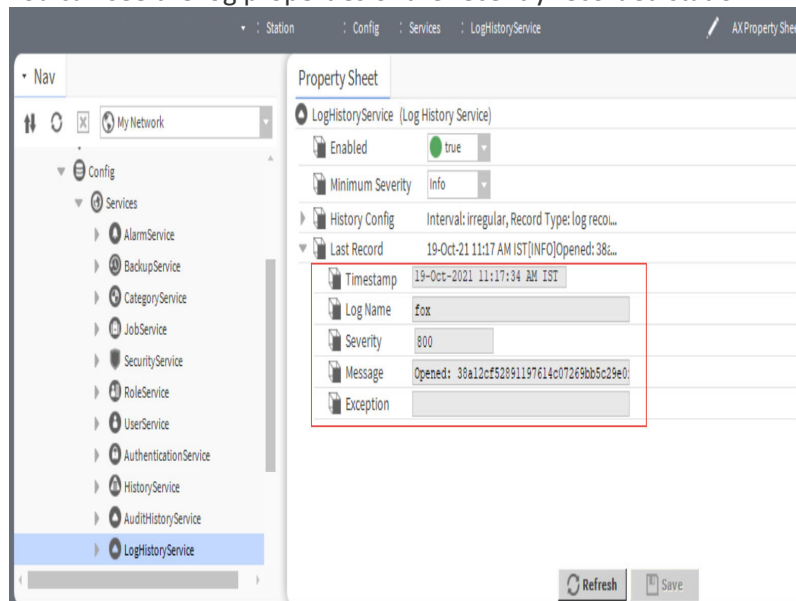


4. To view the station log in **LogHistoryService**, expand **Config > Services** and double-click **LogHistoryService** in the Nav tree.

The **Property sheet** opens with different log history properties.



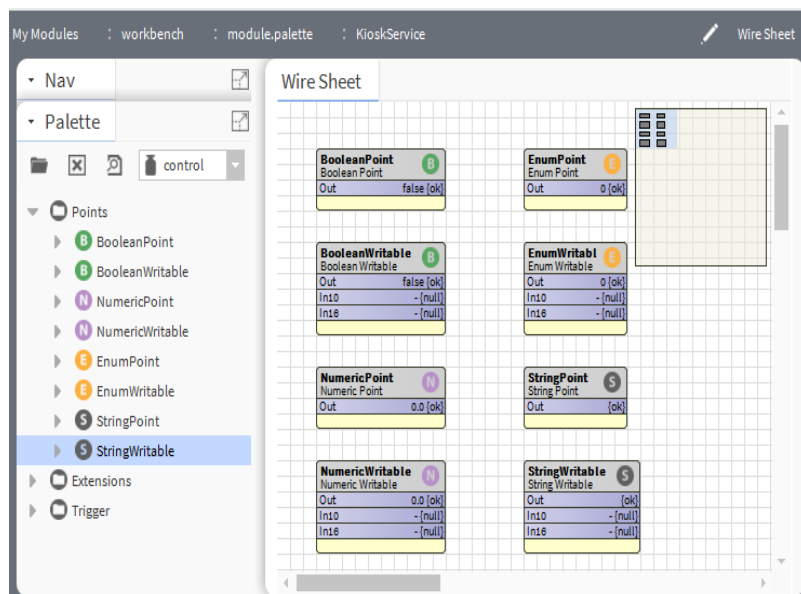
5. To view the station records, expand the **Last Record** in the **Property Sheet**. You can see the log properties of the recently recorded station.



Parent topic: [Key tasks](#)

A company's data control model is based on experiential data collected from analog and other devices. Companies exploit this model to control building systems. Control points are the foundation for all points in the station, including all proxy points.

Figure 1. Simple control points



The framework supports eight simple control point components. Each reflects a combination of a data (value) category and a point type. The **Points** folder of the control palette provides a container for these types or categories of points.

Versions	Boolean	Numeric	Enum	String
Read-only	BooleanPoint	NumericPoint	EnumPoint	StringPoint
Writable	BooleanWritable	NumericWritable	EnumWritable	StringWritable

Categories

The four categories apply to simple point components as well as to other components (for example, weekly schedules). These categories are:

- Boolean represents a binary value with only two states, such as off and on.
- Numeric represents an analog value, such as a temperature, level, rate or similar floating point number, or a varying count (integer). The system uses double-precision (64 bit) values.
- Enum represents an enumerated state (more than two), such as a multi-speed fan with the states of: off, slow, and fast. Enums are often called multi-states or discretes. States typically derive from established integer value/state name pairs.

- String represents one or more ASCII characters (alpha-numeric), often with some literal meaning.

Versions

Each of the four point categories provides two point versions:

- A read-only version, which represents a data item that provides information and cannot be changed. Unlike the writable point version, there are no input type properties for read-only points. These four types are: BooleanPoint, NumericPoint, EnumPoint, and StringPoint.

Note: As copied directly from the control palette, there is no application for read-only points. However, proxy points based upon read-only points, which are identical except for a non-null proxy extension and manner of creation, are both common and useful. For more details, see the *Niagara Drivers Guide*.

- A writable version, which represents a data item that can be changed, as well as read (usually by the station). These types are: BooleanWritable, NumericWritable, EnumWritable, and StringWritable.

An array of 16 *InN* inputs, each with a different priority level, is available to write a writable point's value. By default, the point's value can also be set with an operator-issued action (right-click command), available at priority levels 8 (override) and 1 (emergency).

Other point components

Other point components are found in the kitControl palette. Briefly, these components include:

- Extensions, which expand a given point's functionality. As needed, you can add one or more extensions to a point, each as a child of that point. Extensions add functionality in a modular fashion.
- Time triggers provide periodic actions. These objects do not represent data, but, instead, they regularly fire a topic.
- Other control objects are found in various folders of the kitControl palette. They provide station control logic based on data obtained from points. Example objects include numeric math objects, Boolean logic objects, and a PID loop, among others. Refer to the *Niagara KitControl Guide*.
- [Changing the name of a point](#)
- [About each point's Out property](#)
Each point has input properties and a single output property (Out)
- [About point facets](#)
Primarily, facets determine how the point's value displays in the station. Examples include engineering units and decimal precision for numeric types, and descriptive value (state) text for boolean and enum types.
- [About point versions](#)
Each of the four point categories provides two point versions: read-only and writable.
- [About point extensions](#)
As needed, you can add one or more extensions to a point, each as a child of the point. Extensions add functionality to a point (or extend it) in a modular fashion.

- [About control triggers](#)

There are two time triggers in the control palette: **Interval** and **Daily**. These objects do not represent data, but instead regularly fire a topic.

- [About point status](#)

Along with a point's value, its status is available at point Out. Status flags indicate status, which may get set singly or in combinations.

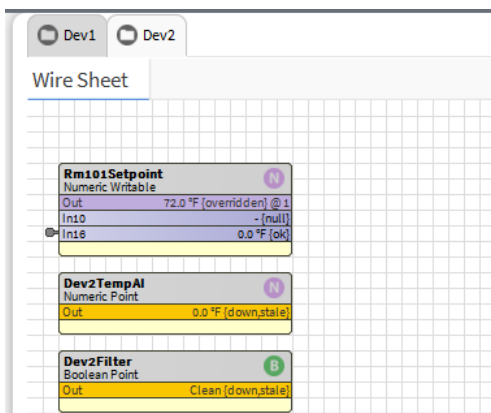
- [About composites](#)

Currently, a composite is something that Workbench allows you do to virtually any component in a station, notably control points and objects, and even folders that contain control logic. When you make a composite, you expose slots of child components in the glyph (object shape) of that parent (composited) component. This can simplify linking and promote reuse of control logic.

Each point has input properties and a single output property (Out)

A point's Out property provides real-time information.

Figure 1. Point Out provides real-time information



At a minimum, the Out property provides:

- A current value that conforms to one of four possible data categories.
- Facets, which define how the value displays. This information includes the value's number of decimal places, engineering units, or text descriptors for Boolean/enum states.
- The current status of the data item, meaning the health and validity of the value. Status is specified by a combination of status flags, such as `fault`, `overridden`, `alarm`, and so on. If no status flag is set, status is considered normal and appears with the default status of `{ok}`.

Status flags are set in a number of ways.

- The currently active priority level (from a 16-level priority scheme) for writable control points only.

The writable point's priority appears at the end of the Out value. By default it is formatted as `@ n`, where `n` is a number from 1 to 16. If the fallback value is in effect, the value is formatted as `@ def {ok} @ 1`.

Parent topic: [About control points](#)

Primarily, facets determine how the point's value displays in the station. Examples include engineering units and decimal precision for numeric types, and descriptive value (state) text for boolean and enum types.

With the exception of proxy points (with possible defined device facets), point facets do not affect how the point's value is processed. Refer to the *Niagara Drivers Guide* for related details.

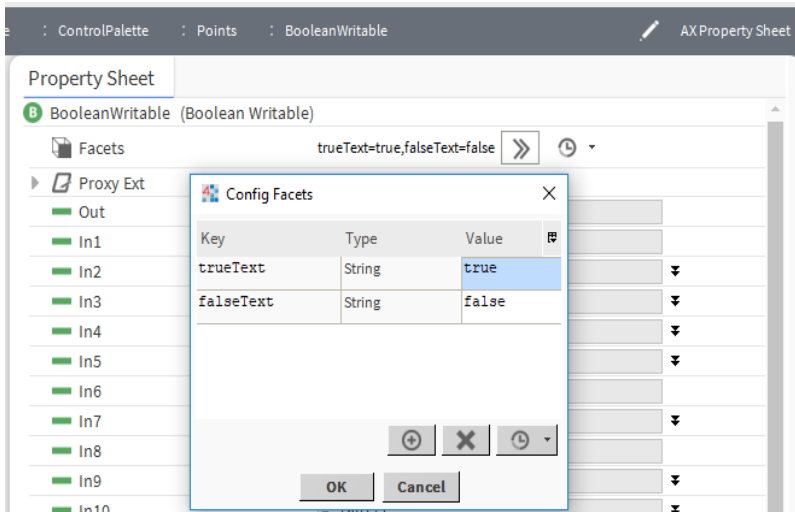
Note: Besides control points, various other components have facets too. For example, many **kitControl** and schedule components have facets. Details about point facets apply to these components too, unless especially noted.

- [Accessing and editing facets](#)
Facets is a frozen slot in all control points and objects in the **kitControl** module. You can modify the facet values in a point's property sheet, using the **Config Facets** window.
- [Facets importance for enum points](#)
Facets for enum-type components (EnumPoint, EnumWritable, EnumSchedule, etc.) define the operating range of the component, meaning its different possible enumerated states. Each state is defined by a pairing of an integer value-to-text, also known as ordinal-tag. Each ordinal must be a unique integer, and each tag must be unique text. By default, the point's value displays using tag text.
- [Effect of facets on point actions](#)
For some points with actions, facets also affect availability in the point's action (command) menu.
- [Maximum override duration facet](#)
Available for some time (but undocumented before), is the ability to limit the maximum override duration of an action invoked on a control point. By default, a manual (level 8) override of a writable point is unlimited in duration, thus the default Permanent label in the action menu.

Parent topic: [About control points](#)

Facets is a frozen slot in all control points and objects in the **kitControl** module. You can modify the facet values in a point's property sheet, using the **Config Facets** window.

Figure 1. Point facets and edit window



In this case a BooleanWritable's default `trueText` facet is `true`—to modify you simply click to select, then type over with whatever text is needed. For example, change `true` to `On` and `false` to `Off`. When done click **OK** and then **Save** to make the actual point change.

Optionally, in addition to modifying existing facets, you can add or remove facet values in a point. On many points you may only modify or provide new values for default facets. In the case of writable points, you can add a facet to limit override duration.

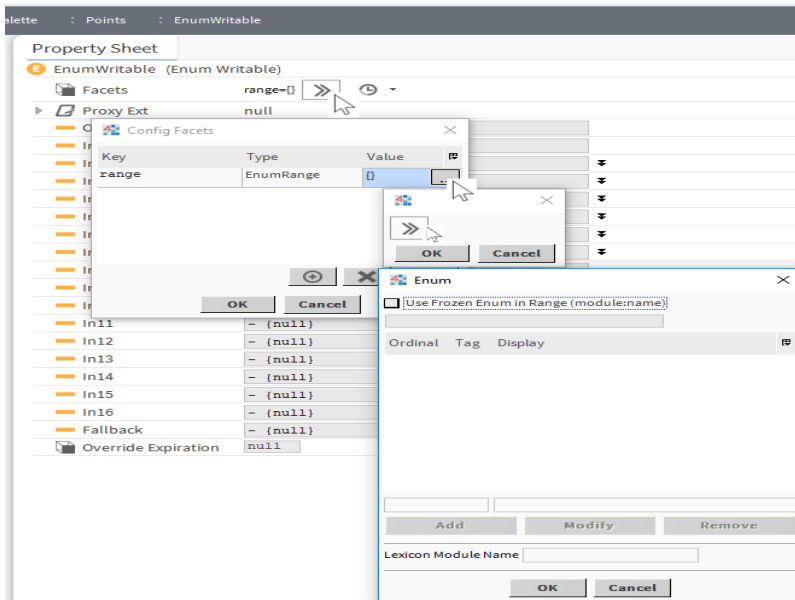
Note: For string-type points (StringPoint, StringWritable), facets typically have little practical application. By default, the **Facets** slot is empty for string-type points.

Parent topic: [About point facets](#)

Facets for enum-type components (EnumPoint, EnumWritable, EnumSchedule, etc.) define the operating range of the component, meaning its different possible enumerated states. Each state is defined by a pairing of an integer value-to-text, also known as ordinal-tag. Each ordinal must be a unique integer, and each tag must be unique text. By default, the point's value displays using tag text.

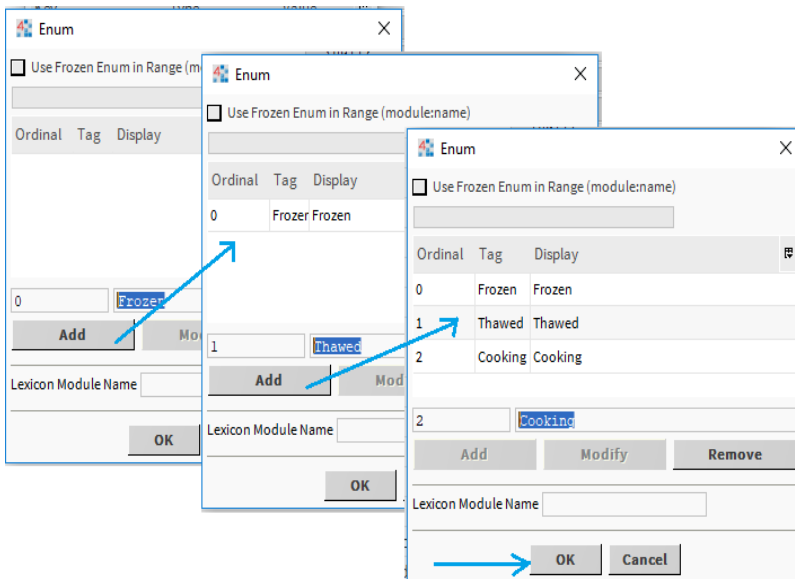
If you add an enum point from the control palette, its **Facets** slot has a blank range entry. Until you edit this facet and supply the ordinal-tag values, it can display only integer values. As shown, a special **Enum** window appears when you edit range facets.

Figure 1. Producing Enum window for enum point "range" facets



In the **Enum** window when adding an entry, the **Add** button becomes available after you enter an integer value in the left side Ordinal box as shown below. Type in the associated text in the right side Tag box, then click **Add** to add to the facet's range. Click **OK** when done, also **OK** on any remaining windows.

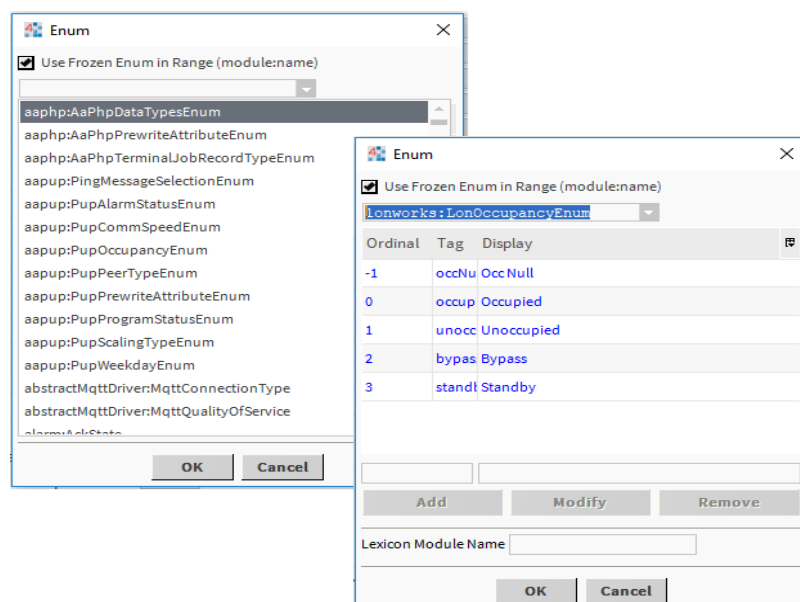
Figure 2. Type unique integer value in Ordinal box and associated text in Tag box



If using lexicons, in the Lexicon Module Name field you can enter the module name of a configured lexicon (for example, `control` or `kitControl`), if Tag strings match lexicon keys in that lexicon file. In this case, enumerations will display the lexicon strings (values) for those ordinals instead of the tag text.

When defining range enumerations, instead of defining a custom one with your supplied ordinals and tags text, you can also select from well known frozen enumerations, as defined in various installed modules. A checkbox enables this and provides a drop-down list for you to select by module and enumeration type.

Figure 3. Frozen enum selection in Enum window



Depending on the driver/network type, the **Point Manager** under a device may automate this facets range configuration when you add enum-type proxy points. For example, under a Lonworks device, if you add a EnumPoint for a Lonworks NVO that uses an enumerated SNVT, that point's facets will automatically be configured with the correct range values.

Note: If an enum-type point receives an input value not included in its defined facets range, it displays the ordinal integer value for that input. This varies from the multistate objects used in r2 Niagara, which would display Error for any value not defined in its stateText entries.

Parent topic: [About point facets](#)

For some points with actions, facets also affect availability in the point's action (command) menu.

- EnumWritable

Upon an override or emergency action, a secondary drop-down selection lists the possible enum values (in its range), using display tag text. This list appears ordered top-to-bottom by the tag associated with each ordinal, lowest-to-highest.

- NumericWritable

Upon an override or emergency action, an entry window permits a value only between the facets min and max values, inclusive. By default, these facets values for numeric-type points are min= -inf and max= +inf (no effective range checking for an action).

For example, you could use this facet's feature with a NumericWritable that sets a temperature control setpoint, by setting its facets min= 65 and max=85. After saving this change, any override or emergency action issued to that NumericWritable would need to fall within this range. Otherwise, a user would see a message showing the acceptable range, and be prompted to try again.

Note: Facets min and max values do not affect any received input values or proxied data, only what can be issued via an action.

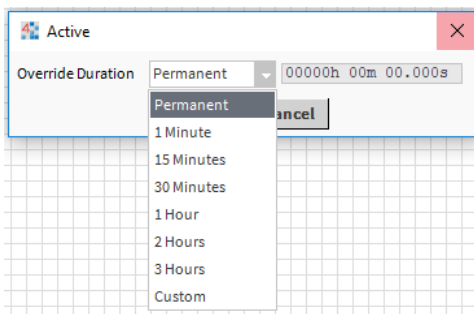
- Maximum override duration (for any writable point type)

Using facets you can also limit the maximum override duration of manual (level 8) override action invoked on writable control points. By default, the manual override of a writable point has no duration limits.

Parent topic: [About point facets](#)

Available for some time (but undocumented before), is the ability to limit the maximum override duration of an action invoked on a control point. By default, a manual (level 8) override of a writable point is unlimited in duration, thus the default Permanent label in the action menu.

Figure 1. Default override action menu for writable control point

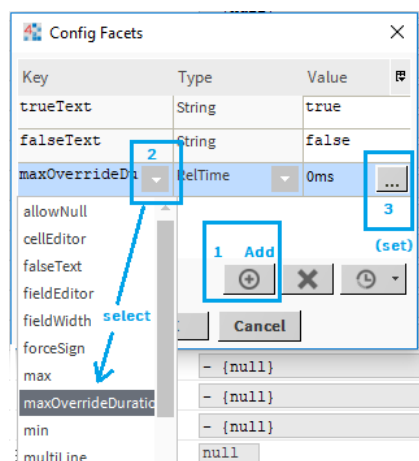


If needed, change this by adding a `maxOverrideDuration` facet (choosing type `baJa:RelTime`), with specified duration time, to either or both:

- Config, Sys Info property
- Any writable control point

Note: Override limits affects operator overrides (level 8) only, as emergency level overrides (level 1) are always unlimited in duration. In other words, an emergency level override lasts until an emergency level "auto".

Figure 2. Config Facets editor when picking `maxOverrideDuration`

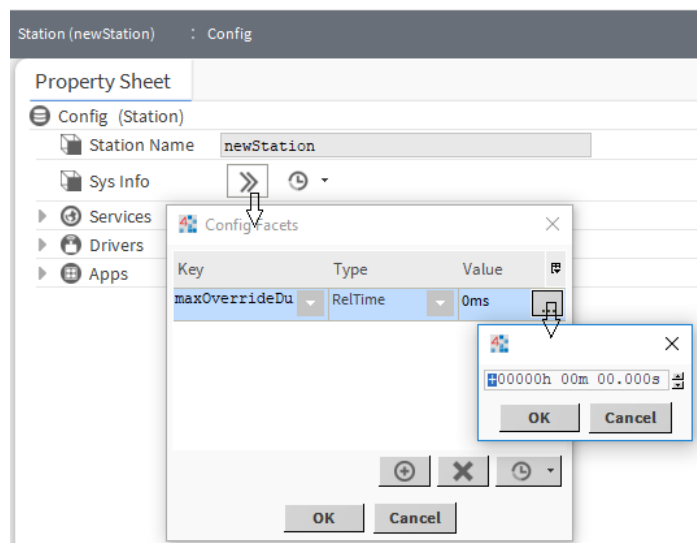


When a writable point is limited by a maxOverrideDuration facet, its action menu adjusts to show the allowable range.

Config, Sys Info property

The Sys Info property of the station's root **Config** component has a facets control, shown in the **Config** component's property sheet.

Figure 3. Global maxOverrideDuration facet added to Sys Info property of station's Config component

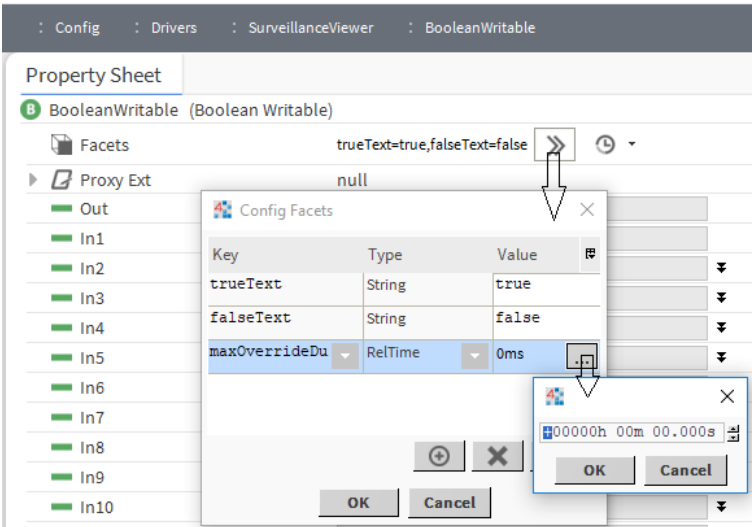


Adding this facet on the Sys Info property acts a global limit (station-wide) to a manual override action to all control points that do not have their own maxOverrideDuration facet.

Any writable control point

Each writable control point in the station can have a separately specified maximum override duration. If this facet is present, it overrides any global (Sys Info) maxOverrideDuration value.

Figure 4. Added maxOverrideDuration facet added at point level (overrides global setting)

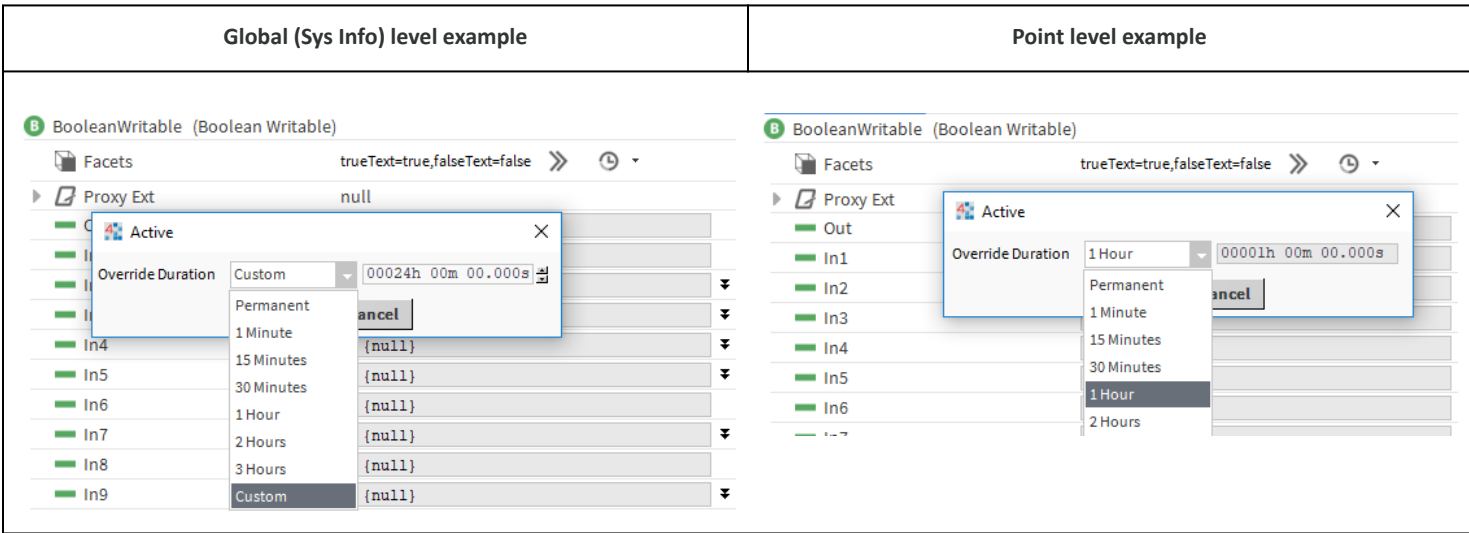


As shown, this maxOverrideDuration facet can be added along with any other facets in use by the control point. The example BooleanWritable point above already had configured facets for trueText and falseText.

Action menu examples

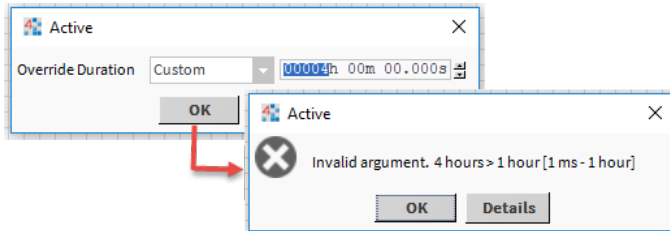
Example action menus from writable points with a maxOverrideDuration facet in effect are shown .

Figure 5. Example override action menus affected by maxOverrideDuration facet



If a system user attempts to invoke a Custom override over the specified `maxOverrideDuration` limit, an error window appears that shows the override duration range as shown below.

Figure 6. Custom override attempt over `maxOverrideDuration` limit produces error window



As shown above, the allowable duration range appears in [brackets], in this case [1ms - 1hour].

Parent topic: [About point facets](#)

Each of the four point categories provides two point versions: read-only and writable.

Read-only version

The read-only version of a control point represents a data item that provides information and cannot be changed. Unlike the writable point version, there are no input type properties for read-only points. These four types are: `BooleanPoint`, `NumericPoint`, `EnumPoint`, and `StringPoint`.

Note: As copied directly from the control palette, there is no application for read-only points. However, proxy points based upon read-only points, which are identical except for a non-null proxy extension and manner of creation, are both common and useful.

Writable version

The writable version of a control point represents a data item that can be changed as well as read (usually by the station). These types are: `BooleanWritable`, `NumericWritable`, `EnumWritable`, and `StringWritable`.

All writable points provide right-click actions. Override actions are evaluated within the priority scheme used by any writable point. In the case of the set action, the point's `Fallback` property is modified. `BooleanWritable` points also offer built-in minimum on/off timers.

An array of 16 *InN* inputs, each with a different priority level, plus a `Fallback` property are available to write a writable point's value. Level 1 is the highest priority, and level 16 is the lowest. By default, the point's value can also be set with an operator-issued action (right-click action), available at priority levels 8 (override) and 1 (emergency).

The 16 priority levels used by writable points are modeled after corresponding BACnet priority levels.

Priority levels conform to the following conventions, from highest to lowest:

1. Emergency (Manual Life Safety)—Unlinkable input, but available as action (command).

2. Automatic Life Safety
3. User Defined
4. User Defined
5. Critical Equipment Control
6. Minimum On/Off (BooleanWritable)
7. User Defined
8. Override (Manual Operator)—Unlinkable input, but available as action (command).
9. Demand Limiting
10. User Defined
11. Temperature Override
12. Stop Optimization
13. Start Optimization
14. Duty Cycling
15. Outside Air Optimization
16. Schedule

Note: Although priority levels are patterned after BACnet, there is no direct linkage to BACnet priorities, even with BACnet writable proxy points. Priority inputs of all writable points are strictly a station-centric implementation.

- **[Priority input scan](#)**
For any writable point, the effective input value is determined by a priority scan, looking for a non-auto action at level 1 (emergency), then the value at the highest valid input, going from level 2, to 3, and so on to level 16. (At level 8, any non-auto action is evaluated as valid).
- **[Priority linking rules](#)**
When linking to the priority inputs of a writable object, you must follow rules.
- **[About Boolean minimum on and off times](#)**
Each Boolean writable point has built-in timers to specify minimum on and/or minimum off times.
- **[About point actions](#)**
Writable points have actions, which, by default appear in a right-click menu when you select a point in a **Workbench** view or in the Nav tree.
- **[About override actions](#)**
Whenever a writable point is controlled from an action issued at either override level, it has an override status. By default, override status color is violet.
- **[About set \(Fallback\) action](#)**
If a writable point has a null or invalid value at inputs In1-In16 (this means that both override levels are currently configured for automatic update), the priority scan sets the Out slot to the value of the **Fallback** property.

- [Modifying default actions](#)
Unless all the defaults for actions of a writable point are acceptable (display names, all actions available, default user access), you may wish to modify action defaults. You can do this selectively from the slot sheet of the writable point.
- [Modifying an action name](#)
- [Modifying action access](#)
- [Modifying config flags for multiple points](#)
- [Modifying the display name for multiple points](#)

Parent topic: [About control points](#)

For any writable point, the effective input value is determined by a priority scan, looking for a non-auto action at level 1 (emergency), then the value at the highest valid input, going from level 2, to 3, and so on to level 16. (At level 8, any non-auto action is evaluated as valid).

Like almost all control execution, this priority scan is event-driven, meaning it occurs when any input value changes. An input's value typically comes from a link—however, for most inputs, you may enter a value directly in the point's property sheet (as an alternative source).

A valid input is one with none of the following status bits set:

- down
- fault
- disabled
- null
- stale

If all 16 priority levels are evaluated without a valid input (and without an action at levels 1 and 8), then the fallback value is used.

You can configure the writable point's Fallback property to be null, so that the point's Out has a null status in this condition. Depending on the specific control sequence and usage of the writable point, this may be an effective solution

However, by default, a set action exists on any writable point, which writes directly to the Fallback value. If you want a writable point to always have a Fallback of null, go to its slot sheet and set the Hidden config flag on the slot. Otherwise, a user can invoke a right-click command to set Fallback to any value.

Parent topic: [About point versions](#)

When linking to the priority inputs of a writable object, you must follow rules.

- Only one link per input (level).
- Levels 1 and 8 are unavailable for links. If a Boolean writable, level 6 is also unavailable.

Priority levels 1 and 8 are reserved for actions (emergency and override). Priority level 6 in a Boolean writable is reserved for minimum on/off times.

Note: Both rules vary from the Niagara priority input scheme, where a single priorityArray input was used for a writable object (AnalogOutput, BinaryOutput, and so forth). You can link that input to multiple priority-type outputs, including those with duplicate priority levels and/or levels also used for object commands (emergency and manual).

Parent topic: [About point versions](#)

Each Boolean writable point has built-in timers to specify minimum on and/or minimum off times.

The respective point properties are Min Active Time and Min Inactive Time. Usage is optional, and both properties work independently. Typical usage is to prevent short-cycling of equipment controlled by the point. Default property times for a Boolean writable are all zeros (00000h 00m 00s), which effectively disables a timer. In either property, you can specify any value needed using a mix of hours (h), minutes (m), and seconds (s) to enable that timer.

A state change transition to active or inactive triggers a minimum timer. This stores the new state value in the point's priority array (at priority level 6) for the duration of that timer. While a minimum timer is in effect, only input changes at a higher priority (5 or above) or an emergency action can affect the Out value.

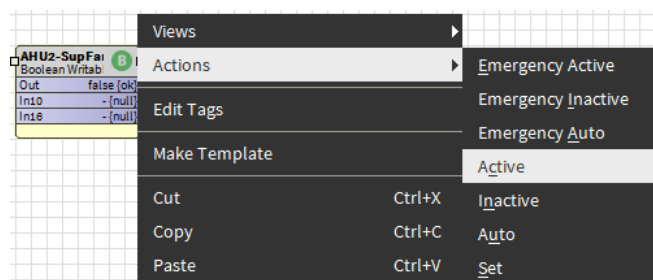
For example, a Boolean writable point controls a fan, with related properties set as follows:

- Starting with the fan stopped at schedule level, priority In16, if a user manually overrides using priority level In8, the fan runs for 90 seconds at priority level 6 (a higher level). After this period, the fan continues running at the override In8 level for the duration of the override.
- During the initial 90 seconds, a different override action (off or auto) is ineffective because the higher priority level 6 remains in control.
- Once stopped, the point's minimum off time keeps the fan off at priority level 6 for the specified duration (for example, 3 minutes and 5 seconds). During this period, only an emergency command or input change at In2–In5 can effect further change.

Parent topic: [About point versions](#)

Writable points have actions, which, by default appear in a right-click menu when you select a point in a Workbench view or in the Nav tree.

Figure 1. Actions available on right-click menu



An action is a slot that activates a behavior. Other control objects and extensions also have actions.

In the case of the four writable control points, default actions include the ability to:

Often, you modify a writable point's value using actions.

Override actions

The actions that override a point's value include Emergency Active, Emergency Inactive, Emergency Active, and Inactive.

You override a point's value at priority levels In1 (emergency override) and In8 (override). This is where control can be independently set or automatically initiated at either level. A level In8 override may last for a defined (or custom) length of time, as specified in the action's window.

Whenever a writable point is controlled from an action issued at either override level, it has an override status. By default, override status color is violet. A manual (level 8) override action to a point (not auto) prompts for an override duration

By default, a Permanent override is the first choice in the drop-down list—the override value will remain effective until the next time this action is auto'ed. Other timed durations are available, including a Custom selection in which a user specifies a duration in hours, minutes, and seconds.

If needed, you can limit the maximum duration of manual override using facets.

After clicking **OK**, the override action is issued to the point—if this is the highest effective priority level, the writable point operates under this control. If this is a timed override, the action is automatically auto'ed upon expiration of the override period.

The Set action

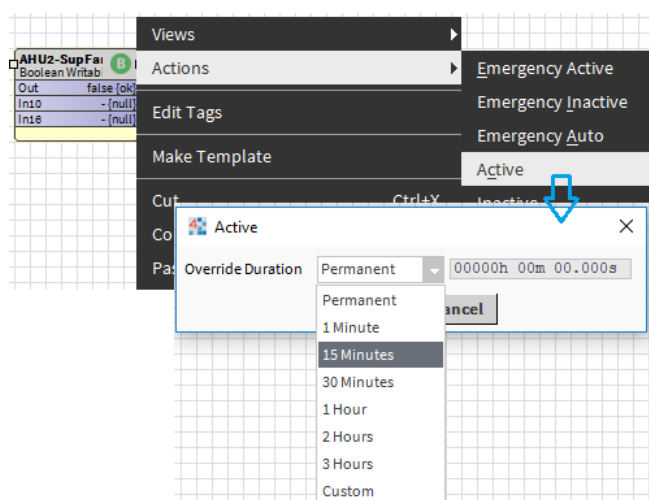
Set the value of the point's Fallback property.

Parent topic: [About point versions](#)

Whenever a writable point is controlled from an action issued at either override level, it has an override status. By default, override status color is violet.

A manual (In8) override action to a point (not auto) prompts for an override duration, see the following image.

Figure 1. Override Duration drop-down list



Note: Emergency overrides (In1) do not have durations—these overrides are permanent until you disable them by right-clicking the point and clicking **Actions > Auto**.

By default, a **Permanent** override is the first choice in the drop-down list. This override value remains effective until the next time this action is auto'd. Other timed durations are available, including a **Custom** selection in which a user specifies a duration in hours, minutes, and seconds.

If needed, you can limit the maximum duration of manual override using facets.

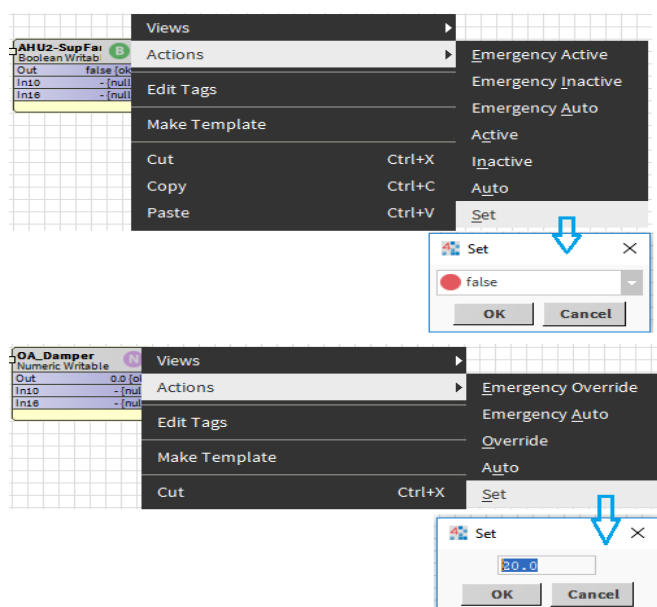
Clicking **OK**, issues the override action to the point. If this is the highest effective priority level, the writable point operates under this override control. If the override is timed, the priority scan returns to automatic control when the override period expires.

Parent topic: [About point versions](#)

If a writable point has a null or invalid value at inputs In1-In16 (this means that both override levels are currently configured for automatic update), the priority scan sets the Out slot to the value of the Fallback property.

By default, an operator-level user can change the Fallback property, by right-clicking on the point and clicking **Actions > Set**. This shows a window that displays the current Fallback value.

Figure 1. Set action prompt to change writable point's Fallback property



From the set action prompt, a **Cancel** leaves the current Fallback property unchanged. Otherwise, the Fallback property is set to the value entered (or currently displayed value).

Note: The set action does not display (or accept) a null value for Fallback. However, a Fallback of `null` can be entered on the point's **Property Sheet**.

A common application for this feature is with numeric writables used as setpoints, particularly under a **NiagaraNetwork**. As proxy points are always read-only points (not writable points), yet they inherit any actions from the source point, this feature provides user access to setpoints via Px graphics without creating additional proxy points. In particular, this set action works well with SetPoint type widgets found in the kitPx palette. For related details, see the *Niagara Graphics Guide*.

Each of the four constant kitControl components also provides a set action that works in a similar manner, including with kitPx widgets. However, a constant object (NumericConst, BooleanConst, etc.), has no priority inputs or Fallback property. The set action simply writes directly to the component's current **Out** slot. For details, see the *Niagara KitControl Guide*.

Parent topic: [About point versions](#)

Unless all the defaults for actions of a writable point are acceptable (display names, all actions available, default user access), you may wish to modify action defaults. You can do this selectively from the slot sheet of the writable point.

Or, using facets you can limit the duration of manual overrides.

The following sections provide more details on slot sheet techniques:

- Display names — Change how the point's action menu lists available actions
- Action access — Limit the actions that are made available, either by user level or for all users

Note: As a global alternative to editing display names on slot sheets, you can edit the default values of lexicon keys, in this case for the `control` module for action type slots. For details, see the *Niagara Lexicon Guide*.

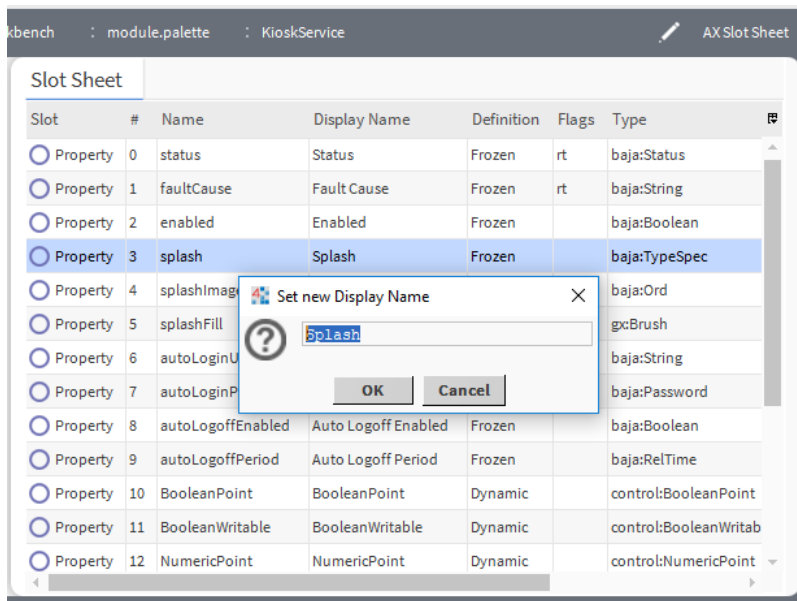
You can also modify the display name of the same action in multiple points in a single operation, using the (default) **Batch Editor** view of the station's **ProgramService (Config > Services > ProgramService)**. This is one of many examples of using the **Batch Editor**.

You can use the **Batch Editor** to modify a config flag of the same slot on multiple components in a single operation.

Display names

By default, action display names are generic (Emergency Inactive and so on). You can change the display name for any action. From the slot sheet, click on an action's Display Name for an editor. When you change a display name from defaults, it appears listed in bold.

Figure 1. Editing action display names from slot sheet



When a user invokes an action, the menu lists possible actions by more meaningful descriptors. For example, you could change the set action display name from Set to Set Fallback.

Action access

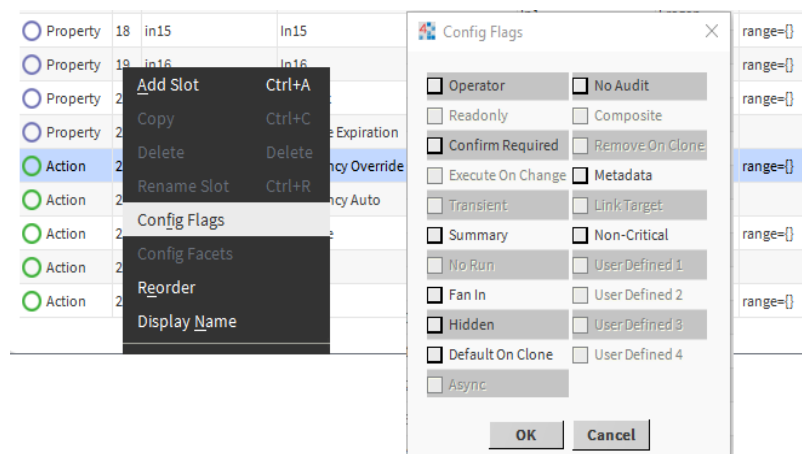
By default, for any writable point, all actions are available to any admin-level user, and all actions except emergency-level ones are available to an operator-level user. As needed, you can selectively hide actions (from any level user), or change default permissions for actions.

Note: From a Px widget, you can also disable Px access to a bound writable point's actions, by setting the `popupEnabled` binding property of the widget to false. In this case, access to the point's actions would still be available from the point's property sheet or in the wire sheet,

unless otherwise changed from its slot sheet. For related Px details, see the *Niagara Graphics Guide*.

From the slot sheet, do this by editing the action's config flags (right-click the action and select **Config Flags** as shown below).

Figure 2. Editing config flags of action to hide or change permission level



In the **Config Flags** editor, you click to assign or remove config flags. As pertains to action slots, the following flags are most often changed:

- Operator

If checked, only operator-level access is needed to invoke the action. If cleared, admin-level access is needed.

- Confirm Required

If checked, the action is invoked. By default, this flag is cleared.

- Hidden

If checked, the action does not appear (is hidden) from the action menu—for any user. You may wish to do this selectively for some actions, for example, the set action for Fallback access. (Note that a user with admin-level rights to the point may still access the point's slot sheet.)

Parent topic: [About point versions](#)

As needed, you can add one or more extensions to a point, each as a child of the point. Extensions add functionality to a point (or extend it) in a modular fashion.

Extensions are found in several palettes, including alarm, control, history, and kitControl. Point extensions include the standard proxy extension (one for each point) and also optional extensions.

There are three main categories of optional extensions:

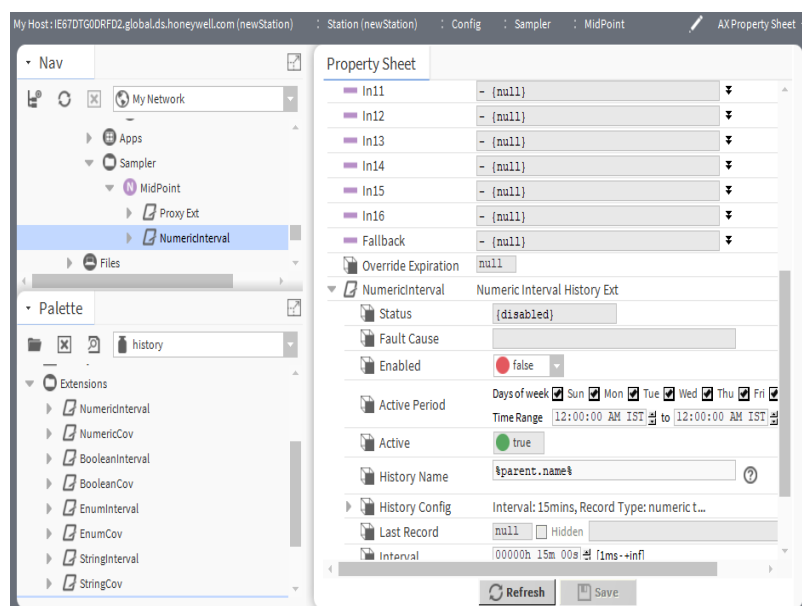
- Control extensions come from the control palette.
- Alarm extensions come from the alarm and kitControl palettes.
- history extensions come from the history palette.

You typically add an extension by either:

- dragging it into the **Property Sheet** of the point, or
- dropping it on the point's icon in the Nav tree.

A point's **Property Sheet** lists extensions below its normal (frozen) properties. You can expand each extension to view and modify its properties.

Figure 1. Extension expanded in a point's property sheet



The screen capture shows the properties for a Numeric Interval History extension that has been added to a numeric point (MidPoint).

If a point has multiple extensions, the station processes them in the same top-to-bottom order that they appear listed in that point's **Property Sheet**. To re-order extensions from the top of the point's **Property Sheet**, or on the point's icon in the Nav sidebar, right-click and select **Reorder**.

Note: If needed, you can also select and expose extension properties (for linking convenience) on the point's glyph by using the Composite editor of the parent point.

- **About the proxy extension**

Each point has a proxy extension (Proxy Ext), which is a frozen property. The proxy extension is important—it indicates how the point's data originate, including details specific to the parentage of the point's network and communications (driver).

- [About control extensions](#)
Control extensions perform additional processing on a point's received value. They are found in the **Extensions** folder of the control palette. As needed, you can add them to points along with alarm and history extensions.
- [About history extensions](#)
Add a history extension to any point to log historical data, that is, to collect a history. In a point history, each collected sample of slot **Out** has a date-timestamp, point status and value. Point history data is viewable in both a table and chart format.

Parent topic: [About control points](#)

Each point has a proxy extension (Proxy Ext), which is a frozen property. The proxy extension is important—it indicates how the point's data originate, including details specific to the parentage of the point's network and communications (driver).

A point's proxy extension is either:

- Null

For any point that you copy from the control palette (or add using the right-click menu), the proxy extension is null (NullProxyExt)—an empty placeholder. The station itself originates the point's default Out value.

Many kitControl components also have a NullProxyExt, as they are based upon ControlPoints. For details, see the *Niagara KitControl Guide*.

- <DriverType>

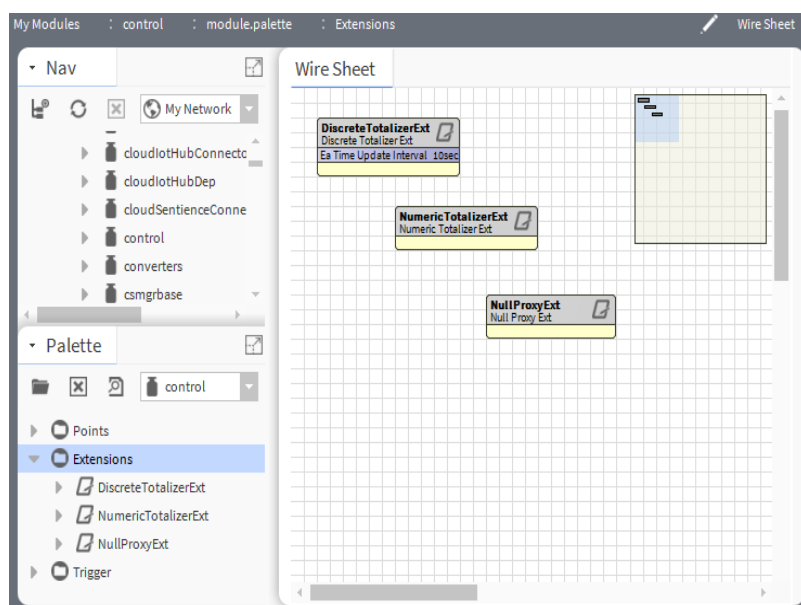
For any proxy point, meaning any of the 8 point types you create using the **Points** extension under a device represented by any of the network types, the proxy extension is <DriverType>ProxyExt. For example, a Boolean writable proxy point under the **Points** container of a BACnet Deevice has a proxy extension of **BacnetBooleanProxyExt**.

For any proxy point, its proxy extension contains information organized in child properties. Some properties may be unique to that specific driver type.

Parent topic: [About point extensions](#)

Control extensions perform additional processing on a point's received value. They are found in the **Extensions** folder of the control palette. As needed, you can add them to points along with alarm and history extensions.

Figure 1. Control extensions



There are relatively few types of control extensions. The following table lists all available control extension types and the applicable point parents.

Control extension type (palette:Folder)	Applies to point types		What it does
	(read-only)	Writable	
DiscreteTotalizerExt (control:Extensions)	BooleanPoint	BooleanWritable	Accumulates runtime and change of state (COS) count. Extension actions permit resetting (zeroing) the runtime and COS count.
	EnumPoint	EnumWritable	
	—	any object with single Boolean Out, e.g. kitControl: Logic object "And"	
NumericTotalizerExt (control:Extensions)	NumericPoint	NumericWritable	Accumulates numeric total using hourly or minutely totalization. Extension has action to reset (zero) total.
	—	any object with single Numeric Out, e.g. kitControl: Math object "Add"	
NullProxyExt (control:Extensions)	(standard for any point)		Adds other types of extensions to the parent component.

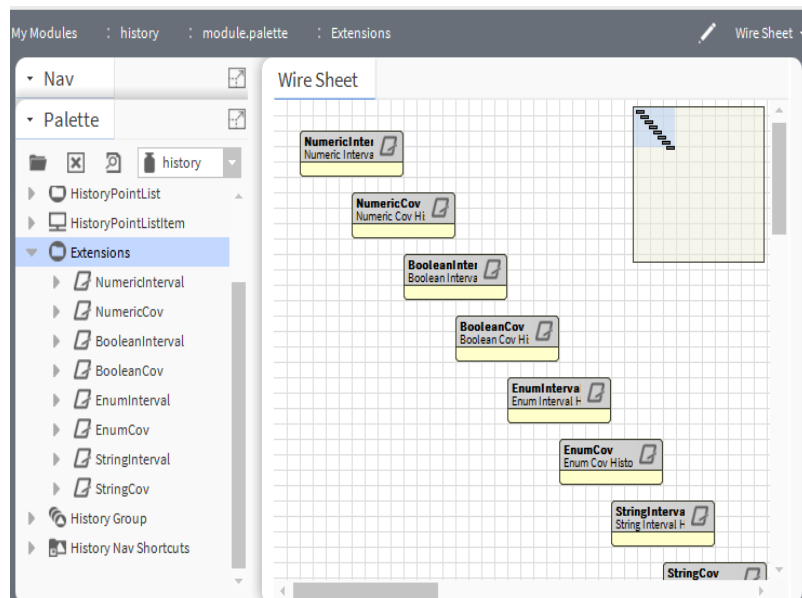
Parent topic: [About point extensions](#)

Add a history extension to any point to log historical data, that is, to collect a history. In a point history, each collected sample of slot Out has a date-timestamp, point status and value. Point history data is viewable in both a table and chart format.

These components are found in the Extensions folder of the history palette.

In Niagara 4.6 and later, if the Enable property is set to false, a copied history extension does not revert.

Figure 1. History extensions



Each history extension has various properties in two major groups:

- Collector properties determine how or when data are collected, such as active time, and (if applicable) either change tolerance or the collection interval time.
- History Config properties determine how the system stores the data. These properties include History Name, Capacity, and Full Policy.

The following lists all history extension types and the applicable point parents.

History extension type	Applies to point types		General description
	(read-only)	Writable	
NumericInterval	as above	as above	Collects upon a repeating time interval, as configured.
NumericCov	NumericPoint	NumericWritable	Collects upon each change of value (outside a specified tolerance) or change of status.
	—	any object with single Numeric Out, e.g. kitControl: Math object type “Add”	
BooleanInterval	as above	as above	Collects upon a repeating time interval, as configured.
BooleanCov	BooleanPoint	BooleanWritable	Collects upon each change of Boolean value (state) or status.
	—	any object with single Boolean Out, e.g. kitControl: Logic object type “And”	

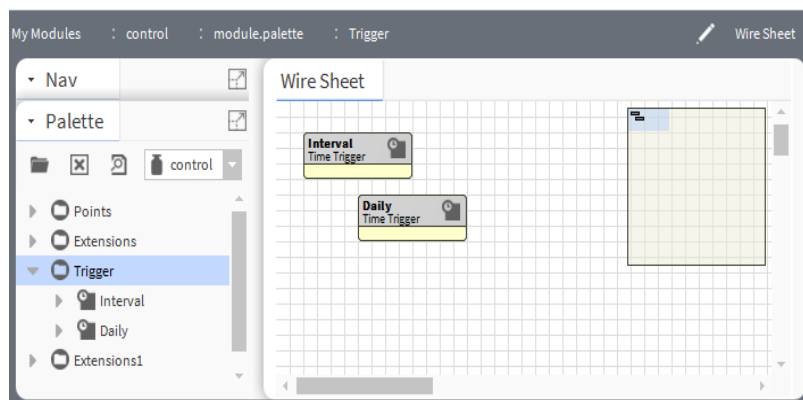
History extension type	Applies to point types		General description
	(read-only)	Writable	
EnumInterval	as above	as above	Collects upon a repeating time interval, as configured.
EnumCov	EnumPoint	EnumWritable	Collects upon each change of enumerated state or status.
	—	any object with single Enum Out	
StringInterval	as above	as above	Collects upon a repeating time interval, as configured.
StringCov	StringPoint	StringWritable	Collects upon each change of string value or status.
	—	any object with single String Out	

All history extensions have an Update History action available. This menu item provides a way to refresh the History Id after a rename. It applies the formatting property (as designated by enclosing % signs) of the Name Format field to the Id of the History Config Id. For example, if the History Name property under a history extension is set to `%parent . name%`, then the History Config Id is initially named based on this parent display name. However, if you rename the parent component, the History Config Id property does not automatically or immediately change. The Update History Id action invokes a renaming of the History Config Id based on the formatting property, so if the parent component (in this example case) is changed, the Update History Id action changes the Id property and, if different from the history name, it results in a change in the history name as well.

Parent topic: [About point extensions](#)

There are two time triggers in the control palette: **Interval** and **Daily**. These objects do not represent data, but instead regularly fire a topic.

Figure 1. Control triggers



The two control trigger types are:

- Interval fires at a regular, repeating intervals specified in its Trigger Mode property. For example, every n minutes, n hours, or whatever combination is needed.

- Daily fires once a day at a specific time and day of the week, as specified in its Trigger Mode property. For example, at 00:00:00 (midnight) all days of week except Sunday.

Each type has even more configuration capabilities to further define the fire time.

To use a trigger, you typically link it to a selected action of a point extension (for example a control, alarm, and history) to automate an action. Often, you use a trigger as a child of a particular point (sibling to the linked extension). Or, you can have a trigger in the same container as multiple points and link it to more than one point or point extension.

For example, a **Daily** trigger (defined for midnight) can be linked to the Reset Elapsed Active Time action of a Discrete Totalizer extension on a Boolean writable point. In this case, that point's Discrete Totalizer extension would only show runtime accumulated during the current day.

Other objects with trigger functions are found in various palettes. The most closely related is the **TriggerSchedule** object, found in the schedule palette.

Parent topic: [About control points](#)

Along with a point's value, its status is available at point Out. Status flags indicate status, which may get set singly or in combinations.

Status flags are typed by a unique text string (for example: alarm or down), and many have associated status colors (a background and a foreground). For example, the default status colors for alarm is white text on red background. Status without any flags set is considered normal (text {ok}), and is without color indication.

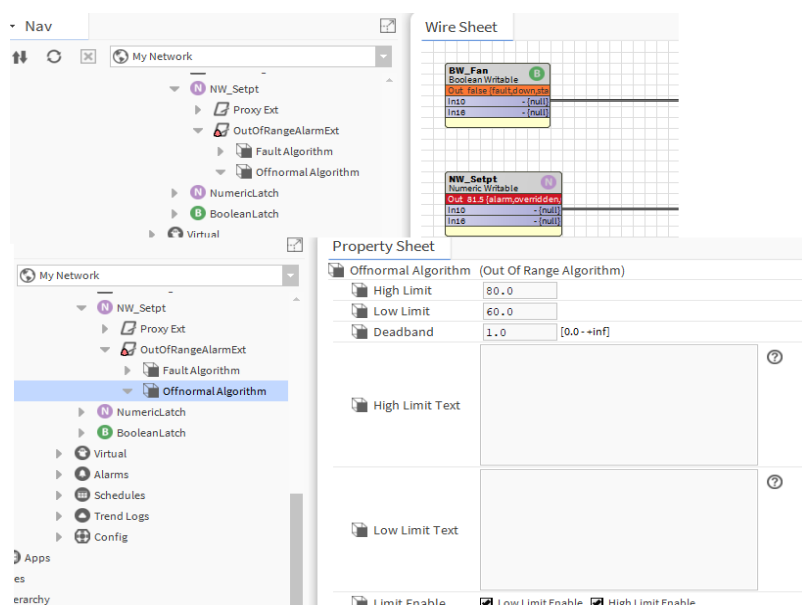
Note: For each installed lexicon, text strings for status flags (plus associated colors), are individually adjustable by editing default values in that host's `ba.ja` lexicon, using Workbench's Lexicon Editor. For a default English installation, to change default status appearance settings, edit the `en: ba.ja` lexicon.

Status flags are set differently depending on the type of point or control object. For simple control points (NullProxyExt) the following status flags are the only ones set and cleared:

- alarm indicates that a point is currently in an alarm condition as defined by its alarm extension.
- unackedAlarm indicates that a point has an alarm extension assigned to an alarm class that requires acknowledgment, but the last alarm event has not yet been acknowledged. The point may or may not be in alarm.
- override applies to writable points only, during a user-initiated action at priority levels In8 (override) or In1 (emergency). The override flag clears when the action times out, or when a user issues an Auto action at that same priority level.
- null indicates that a writable point has null or otherwise invalid value at In1 – In16, plus null configured as the Fallback value.

See the following image for examples of override and alarm status in simple control points.

Figure 1. Status with simple control points



- **Types of status flags**
Status is indicated by text on a colored background.
- **Priority of status indication**
Since status flags for a point or object can get set in combinations, status color indication uses a priority method.
- **Propagate Flags status option (linked Math and Logic objects)**
By default, kitControl objects maintain independent status flags from input-linked points. However, as a configuration option in each math or logic-type kitControl object (kitControl palette folders **Math** and **Logic**), you can specify Out status to propagate from input status.
- **About isValid status check**
When linking an input-type slot of a writable point or kitControl object, the value received at the input is processed (evaluated) by that point or object, if it is valid.

Parent topic: [About control points](#)

Status is indicated by text on a colored background.

The table lists the status types, illustrates the colors and explains what they mean.

Type	Default Colors, Example	Meaning
alarm	white text, red background 65.0 °F	Point currently has a value in an alarm range, as defined by property in its alarm extension.
fault	black text, orange background 208.8 °F	Originates from a proxy point only. Typically indicates a configuration or licensing error. If it occurs after normal operation, it may indicate a native fault in device, or the point's parent device has a fault status.
overridden	black text, magenta background 72.0 °F	Current point control is from an action, meaning a user-invoked command at either priority level 8 (override) or priority 1 (emergency).

Type	Default Colors, Example	Meaning
disabled	gray text, light gray background 79.0 °F	Originates from a proxy point only. Point (or its parent device or network) has been manually disabled (property enabled = false).
down	black text, yellow background 68.4 °F	Originates from a proxy point only. Driver communications to the parent device are currently lost, based upon the device status (Monitor) configuration for that network.
stale	black text, tan background 73.4 °F	Originates from a proxy point only. Driver communications have not received a requested response for this data item within the configured times (Tuning period).
null	(no color indication)	Current point control has entered a null state, vs. a specific value and priority level. Typical to fallback operation for a writable point. Note: If linking a null status Out to a simple data slot, the point's null value is processed. See the <i>Niagara KitControl Guide</i> for more details.
unackedAlarm	(no color indication)	Last point alarm event has not yet received user acknowledgment. Point's alarm extension uses alarm class requiring acknowledgment.

Parent topic: [About point status](#)

Since status flags for a point or object can get set in combinations, status color indication uses a priority method.

Among those 6 status flags with associated colors, priorities (and default colors) are:

1. disabled (dark gray): Proxy point origination only. Point may have other status flags set. Typically, you manually set and clear this status (unlike others). After disabled is set for a proxy point, it is no longer polled. Further status changes do not occur until disabled is cleared.
2. fault (orange): Typically proxy point origination only.
3. down (yellow): Proxy point origination only.
4. alarm (red): Point may have other status flags set.
5. stale (tan): Proxy point origination only.
6. overridden (magenta): Point may have other status flags set.

Note: Status types unackedAlarm and null do not affect the indicated status color. For more details on proxy point status, see *Niagara Drivers Guide*.

Parent topic: [About point status](#)

By default, kitControl objects maintain independent status flags from input-linked points. However, as a configuration option in each math or logic-type kitControl object (kitControl palette folders **Math** and **Logic**), you can specify Out status to propagate from input status.

The object's Propagate Flags property allows you to select any combination of the following status types for propagation:

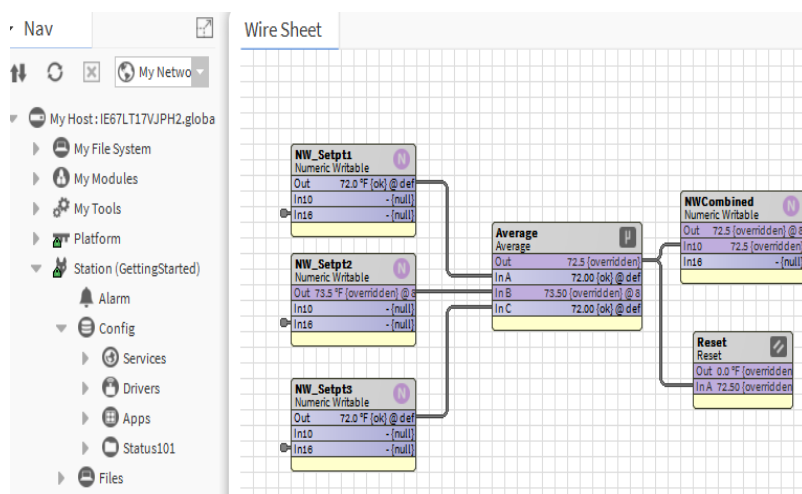
- disabled
- fault
- down
- alarm
- overridden

Note: If the math or logic object has multiple inputs, and you set the propagateFlags property to select one or more of the statuses above, simple OR logic is used across all inputs for the propagation of each selected status.

Depending on usage, status propagation may be extensive. Note that four of the five status types (all except alarm and overridden) are invalid status, meaning they cause the output of the object (if linked) to be considered invalid at its destination target.

As an example of status propagation, some number of NumericWritable points are used to establish setpoints, and you link them all to a Math: Average object for downstream zone control. In turn, the Average object feeds a Math: Reset object. Both math objects have override enabled in their propagateFlags property. A user issues an override (action) to one of the NumericWritable points, to override a setpoint.

Figure 1. Status propagation in linked writable points, kitControl objects



For the duration of the override, the linked Average object will also have an overridden status, as will the Reset object, and so on. However, note that the linked writable point (NWcombined) in this example does not have overridden status—status never propagates to any point.

Note: Before using this feature in an actual job, you should test and evaluate results to be sure it has the desired effect. For example, if a Logic or Math object is exposed in a graphic and appears

overridden, a user may (incorrectly) assume that they can right-click command (perform an action) on that kitControl object, based on status color indication.

Parent topic: [About point status](#)

When linking an input-type slot of a writable point or kitControl object, the value received at the input is processed (evaluated) by that point or object, if it is valid.

Note: A valid input is one with none of the following status flags set:

- down
 - fault
 - disabled
 - null
 - stale
-

If any of the above status bits are set at an input, that input value is not used.

- If a kitControl object with a single input, by default that object uses the last known valid received (at least until the input becomes valid again).
- If a kitControl object with a multiple inputs, only the valid inputs are evaluated.
- If a writable point, the priority scan continues.

Parent topic: [About point status](#)

Currently, a composite is something that Workbench allows you do to virtually any component in a station, notably control points and objects, and even folders that contain control logic. When you make a composite, you expose slots of child components in the glyph (object shape) of that parent (composited) component. This can simplify linking and promote reuse of control logic.

CAUTION: Composites have associated issues. For now, you should avoid making folder composites in your control logic, and instead use the composite feature only at the point/object level to expose extension slots (if necessary).

When you composite a component (say a control point, meaning its contents), you select specific slots in child components (say, properties and/or actions of its extensions) to be exposed in the shape of that point. Then, when looking at that point in the wire sheet view of its parent folder, you can see exposed properties of children as linkable slots (and/or available actions).

Note: If you are familiar with Niagara r2, the composite concept is similar to Bundle or Composite objects, only more flexible—you can expose slots in containers many levels down, for example. However, please see the Caution above.

Some composite examples

A few simple examples of composites are as follows:

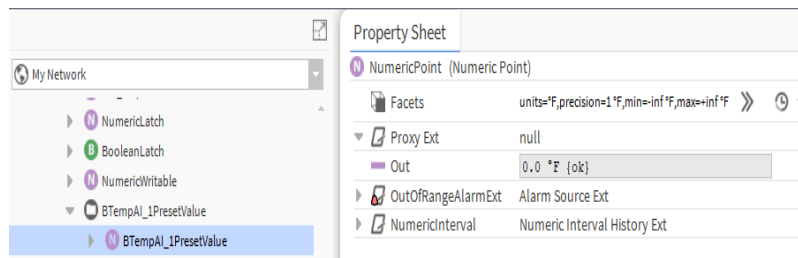
- Point-level composite
- Folder-level composite
- **Point-level composite**
The following image shows a proxy NumericPoint representing a space temperature value that has two extensions:
- **Folder-level composite**
This topic provides a simple example of three Math objects chained together.
- **Composite issues**
Although composites can simplify linking and make understanding logic flow easier, they always introduce performance issues. Perhaps the biggest issue is that once you link a dynamic value to a composite, for example the out of a proxy point, it essentially pinned into the subscribed state. This means that proxy point will always be polling (regardless of any other usage).

Parent topic: [About control points](#)

The following image shows a proxy NumericPoint representing a space temperature value that has two extensions:

- an alarm extension (OutOfRangeAlarmExt)
- a history extension (NumericInterval)

Figure 1. Property sheet of proxy point with two extensions

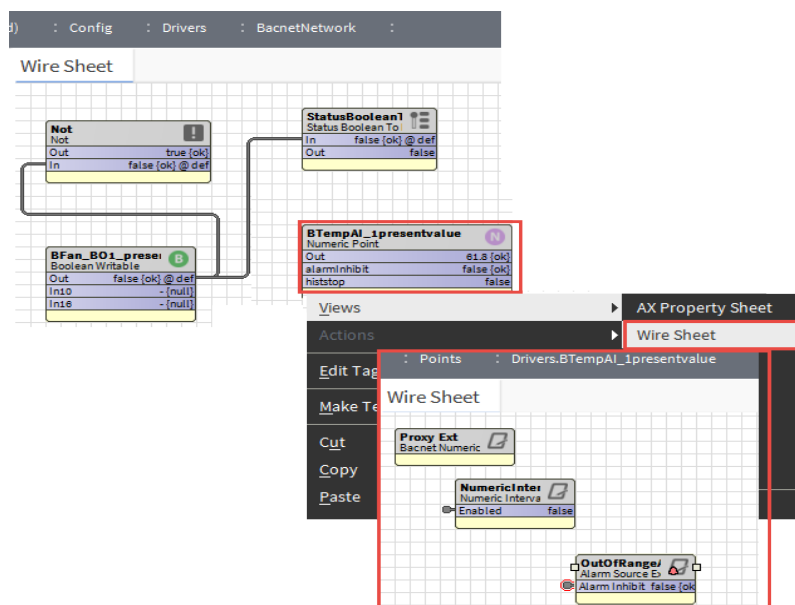


In this example, when another system-related BooleanWritable (representing the system fan) has a false (Off) status, it is desired that this temperature point be:

- disabled from alarming, and
- disabled from continued history collection

To achieve these disable functions, you must link the controlling source fan out to a slot of each extension (visible in point's wire sheet view, but not in the parent wire sheet for the point itself). Furthermore, other kitControl objects needed. Without making a composite, the necessary objects and links may appear.

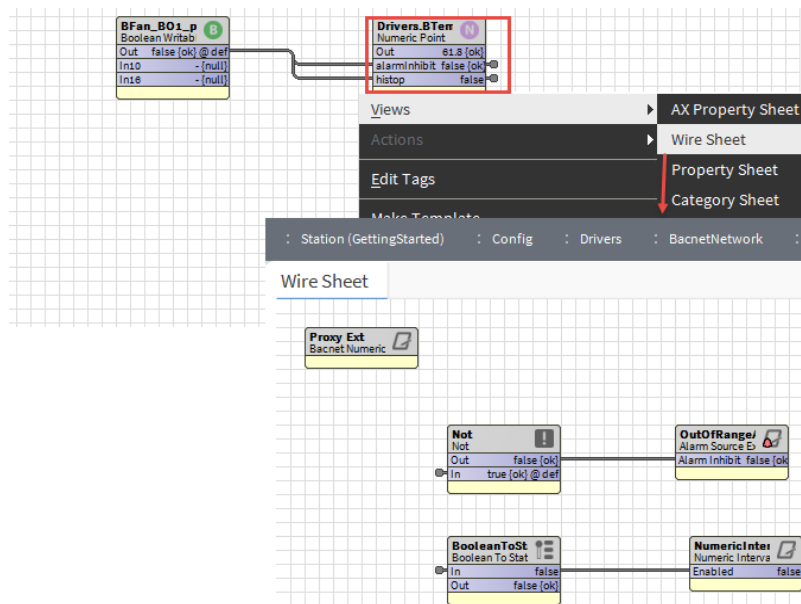
Figure 2. Proxy point example without composite



Notice that when looking at the proxy NumericPoint in the wire sheet of its parent folder, it is not apparent that this point has linked extensions.

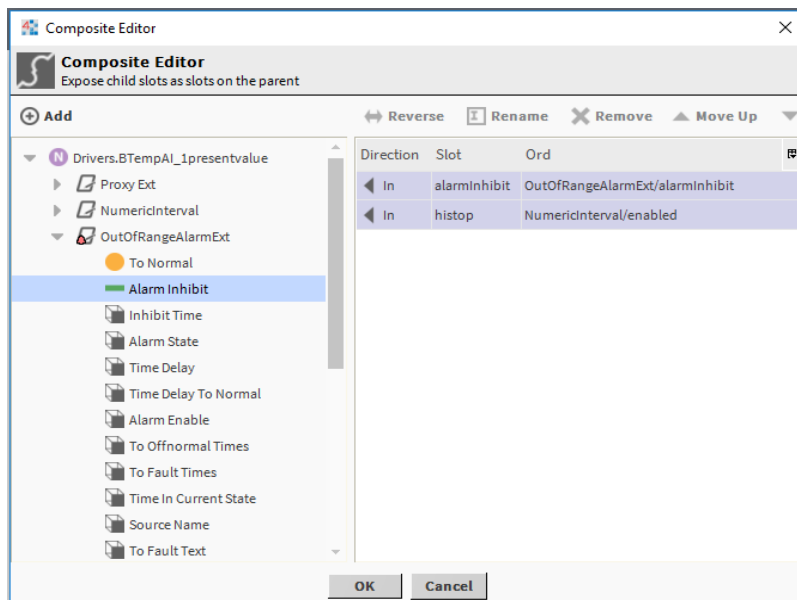
By making a composite of the NumericPoint (acting as the container for both the extensions plus the additional kitControl objects) you can simplify reuse and clarify available links. The following image shows the now-composited NumericWritable linked to the controlling BooleanWritable, and the wire sheet view of the NumericWritable that contains the needed kitControl objects.

Figure 3. Proxy point as composited container



In this example, the exposed input slots in the composite were renamed from **In** to **alarmInhibit** and **histStop** respectively, to clarify what each does. When looking at the **Composite Editor** for this example NumericPoint, it appears as given below.

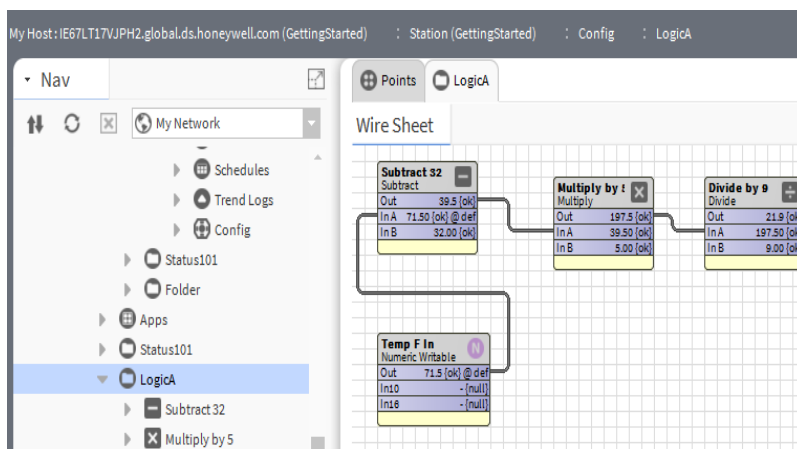
Figure 4. Composite Editor for example NumericPoint



Parent topic: [About composites](#)

This topic provides a simple example of three Math objects chained together.

Figure 1. Simple example of folder before compositing



The objects are located in a LogicA folder. Together, they perform a Celsius to Fahrenheit conversion. A NumericWritable is also shown linked to the first Math object to test.

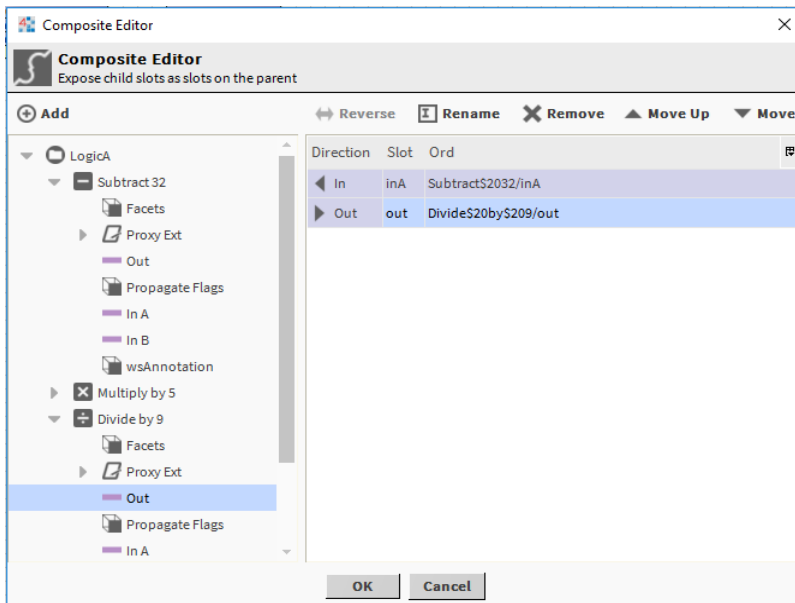
If this application was needed later, you could copy all ttree linked objects again and insert them in another (perhaps already crowded) wire sheet. However, the middle Multiply object reveals an intermediary result that is distracting.

Or, you could just create a new subfolder with only the three linked objects and then link directly to the child objects as needed (however, it would not be obvious from the parent's wire sheet that links to children in that folder were established).

It would be better to encapsulate this into a single object with only a single input (degrees F) and single output (degrees C). You can do something like this by compositing the parent container, in this example folder FolderA.

In this case, you would want to delete the link from the test NumericWritable first, then open the Composite Editor for the parent component FolderA. The Composite Editor lets you expand the tree of all contained components and "expose" those items of interest.

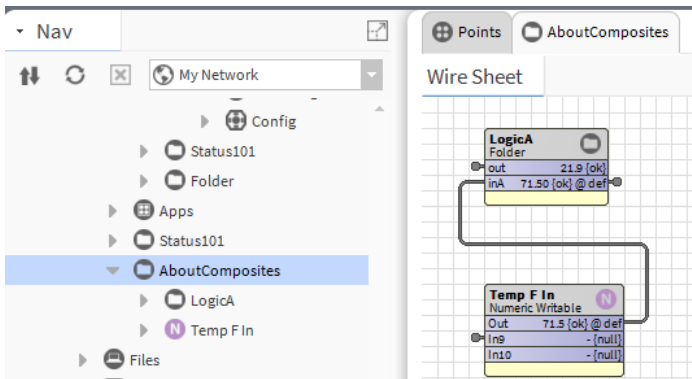
Figure 2. Launching and using the Composite Editor



In this example, only the **In A** of the first math object and the **Out** of the third math object is selected to be exposed. The **Composite Editor** provides a tree pane showing slots of points and objects (by clicking the expand controls), and a slot is exposed by simply double-clicking it. Other controls in the editor are available to rename, remove, reorder and reverse exposed items, but are not used here.

After clicking **OK** to perform the composite, the item composited (in this example, LogicA) shows exposed slots when viewed in its parent's wire sheet. The following image shows the test NumericWritable now linked to the composited LogicA folder.

Figure 3. Example LogicA folder showing exposed input and output



You could later reopen this folder's **Composite Editor** and rename the exposed properties differently, perhaps inDegF and outDegC (to make the purpose of the composited folder clearer). This would not affect the three (child) math objects in any way.

Parent topic: [About composites](#)

Although composites can simplify linking and make understanding logic flow easier, they always introduce performance issues. Perhaps the biggest issue is that once you link a dynamic value to a composite, for example the out of a proxy point, it essentially pinned into the subscribed state. This means that proxy point will always be polling (regardless of any other usage).

In addition, each item exposed in a composite represents a link, where each link consumes some small amount of station resources. If used excessively, composites could noticeably reduce the total capacity of the station.

Parent topic: [About composites](#)

An object on a Wire Sheet view of a data model is a point component that represents a piece of data.

- **Linking basic objects**
Object links define the direction of data flow.
- **Linking continuous objects**
You can maintain a continuous link state as you drag connection wires to link to multiple target objects from a single source object. This allows you to link from a single source to as many targets as you want without having to click on the source object to re-initiate each link.
- **Creating multiple links at the same time**
Link Mark is a feature that allows you to perform one-to-many, many-to-one, and many-to-many linking in a single operation.
- **Viewing links**
The link highlighting option (enabled by default) is available on the wire sheet. Link highlighting makes it easier to distinguish links on a crowded wire sheet.
- **Editing links**
You can edit a link directly from the wire sheet view using an **Edit** window without having to go to the source component's Link Sheet view
- **Organizing objects in a hierarchy**
In **Wire Sheet** as well as the browser-based **Web Wire Sheet**, you add relations among objects for purposes of building hierarchies. Relation links are directional and tag-based. They indicate how an object relates to another object. Relation linking functions in the same manner as object linking.
- **Zoom controls**
Complex wire sheets can be difficult to manage. The zoom controls let you magnify and reduce the screen images as needed.
- **Keeping all objects within view**
Having to scroll vertically and horizontally in the wire sheet view can become tiresome. You can configure this view to restrict its size so that all objects remain in view. If your model is complicated, you may need to configure multiple wire sheets.
- **Link navigation**
Link knobs provide easy navigation.
- **Using Web Wire Sheet**
Web **Wire Sheet** is an implementation of the **Wire Sheet** view that runs in your web browser and is accessible from your desktop or mobile device as part of the HTML5-based web interface. When you are logged in to the **Niagara** web interface via browser, the **Web Wire Sheet** is available for any component. The following gives you an overview of the Niagara 4.12 **Web Wire Sheet** user interface features.

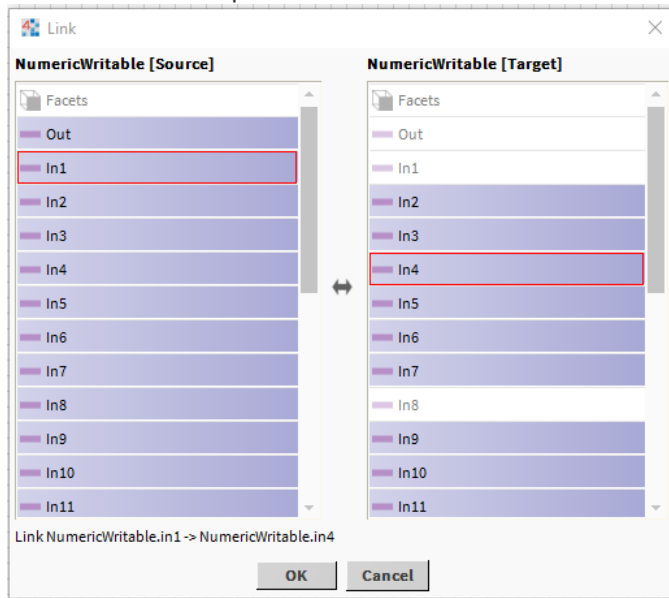
Object links define the direction of data flow.

1. Move the mouse over the In or Out node of a wire sheet object (glyph) until the area is highlighted and the mouse pointer changes to a white arrow.
2. Do one of the following:



- To complete a link, click and drag from the point of origin to the desired In or Out slot of the target object and release the mouse button.
- Drag a wire from the selected object to a vacant slot on another wire sheet object.

The **Link** window opens.



3. To create the link using the **Link** window, select the desired slots and click **OK**.

To identify the objects (slots), their names display at the top of their respective columns. The object in the left column is the source object; the object in the right column is the target object.

You cannot select invalid (dimmed) slots when dragging from an In to an Out slot or when using this window.

Red rectangles surrounding each selection indicate the link.

A summary of the currently selected slots displays above the **OK** and **Cancel** buttons.

Parent topic: [Wire Sheet object management](#)

You can maintain a continuous link state as you drag connection wires to link to multiple target objects from a single source object. This allows you to link from a single source to as many targets as you want without having to click on the source object to re-initiate each link.

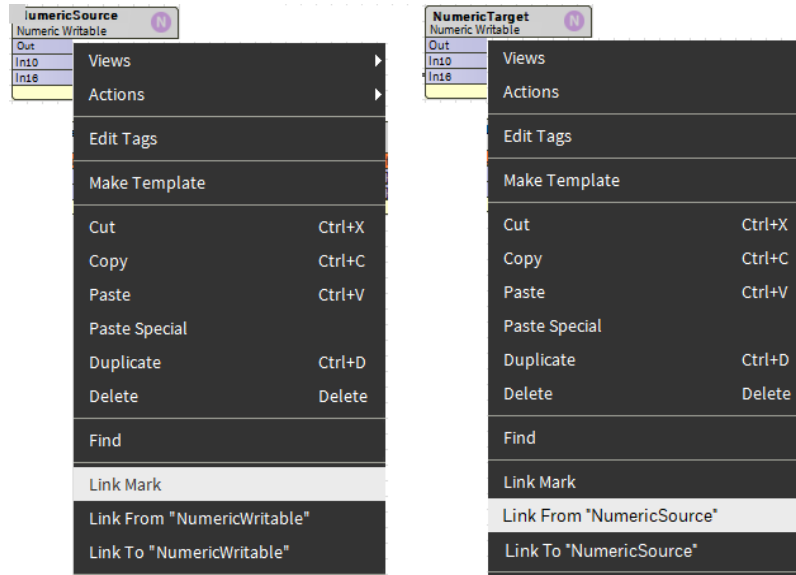
1. Hold the **Shift** key any time you release the mouse button.
If the mouse pointer is over a valid object node, a link is established or the **Link** window opens to complete the link. This continued link state is indicated by the target end of the wire sticking to and moving with the mouse pointer.
2. To deactivate the link state, release the **Shift** key and click anywhere or touch any key.

Parent topic: [Wire Sheet object management](#)

Link Mark is a feature that allows you to perform one-to-many, many-to-one, and many-to-many linking in a single operation.

1. To define one or more selected objects as a link source or target, select the **Link Mark** command.
This command specifies that the selected objects are to be one side of the link operation. The names of the marked objects display as part of the **Link Mark** or **Link From** command

in the menu.



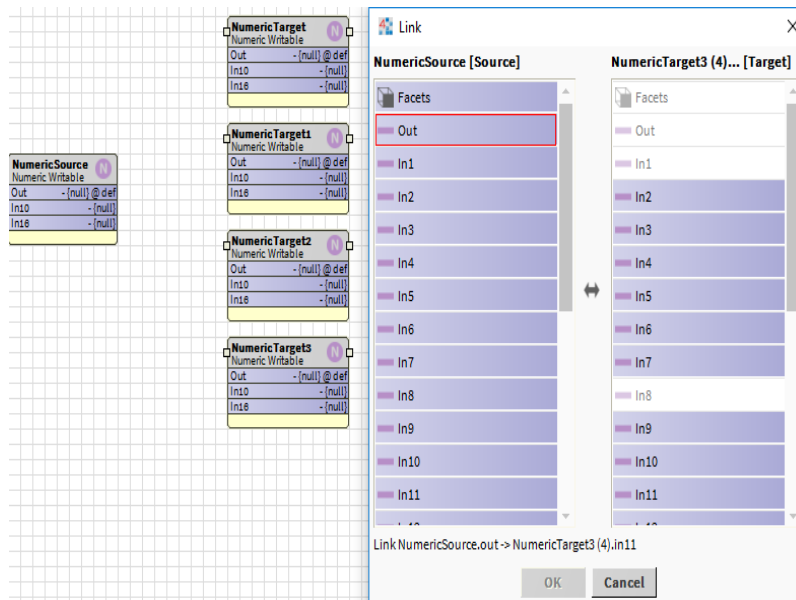
- To define one or more selected objects as a link source or link target, select the **Link From** command.

This command opens the **Link** window with the marked object as the source object. You can still change source and target roles in the window using the **Reverse** button.

- To define one or more selected objects as a link source or link target, select the **Link To** command.

This command opens the **Link** window with the marked object as the target object.

You can still change source and target roles in the window using the **Reverse** button.

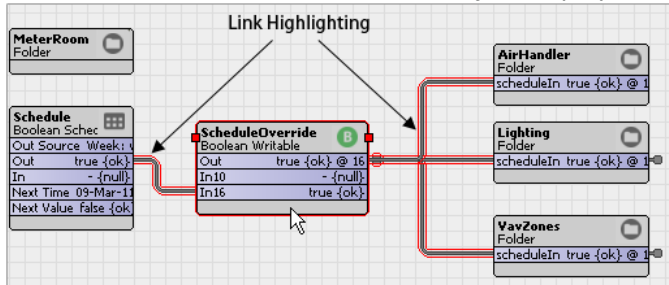


Parent topic: [Wire Sheet object management](#)

The link highlighting option (enabled by default) is available on the wire sheet. Link highlighting makes it easier to distinguish links on a crowded wire sheet.

1. Select a component on the **Wire Sheet**.

All links associated with the selected object display with a colored outline around them.



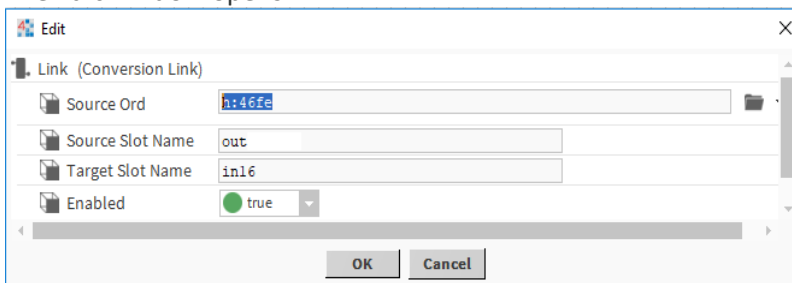
Note: Highlighted links do not mean that the LINK is selected.

2. Hold the shift key and select another component. Subsequent link highlights display a different color highlighting (the system uses up to 20 colors before repeating).
3. To customize the colors, edit the system.properties file (in the Nav tree lib folder, under **MyFiles > Sys Home**). You specify each color using the standard hexadecimal notation used for HTML color display.

Parent topic: [Wire Sheet object management](#)

You can edit a link directly from the wire sheet view using an **Edit** window without having to go to the source component's Link Sheet view

1. To open the **Edit** window, right-click a single link (not multiple links or knob links) and select **Edit** from the menu. The **Edit** window opens.

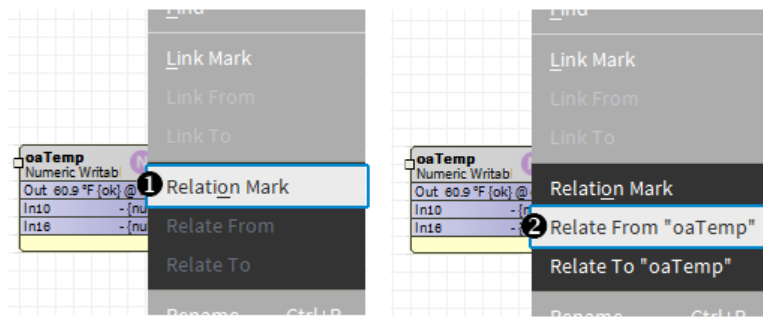


2. Use the property fields to change any of the following link property values: Source Ord, Source Slot Name, Target Slot Name, or Enabled and click **OK**.

Parent topic: [Wire Sheet object management](#)

In **Wire Sheet** as well as the browser-based Web **Wire Sheet**, you add relations among objects for purposes of building hierarchies. Relation links are directional and tag-based. They indicate how an object relates to another object. Relation linking functions in the same manner as object linking.

1. To define one or more selected objects as a relation source or relation target, select the **Relate Mark** command.

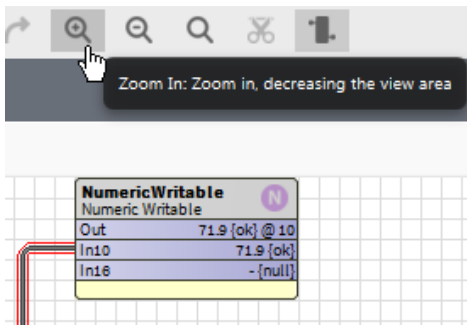


2. Select the **Relate From** command to define one or more selected objects as a relation source or relation target.
This command opens the **Relation** window with the marked object as the source object. Select a tag from the dropdown list.
3. Select the **Relate To** command to define one or more selected objects as a relation source or relation target.
This command opens the **Relation** window with the marked object as the target object.

Parent topic: [Wire Sheet object management](#)

Complex wire sheets can be difficult to manage. The zoom controls let you magnify and reduce the screen images as needed.

Figure 1. Zoom controls in the Wire Sheet view



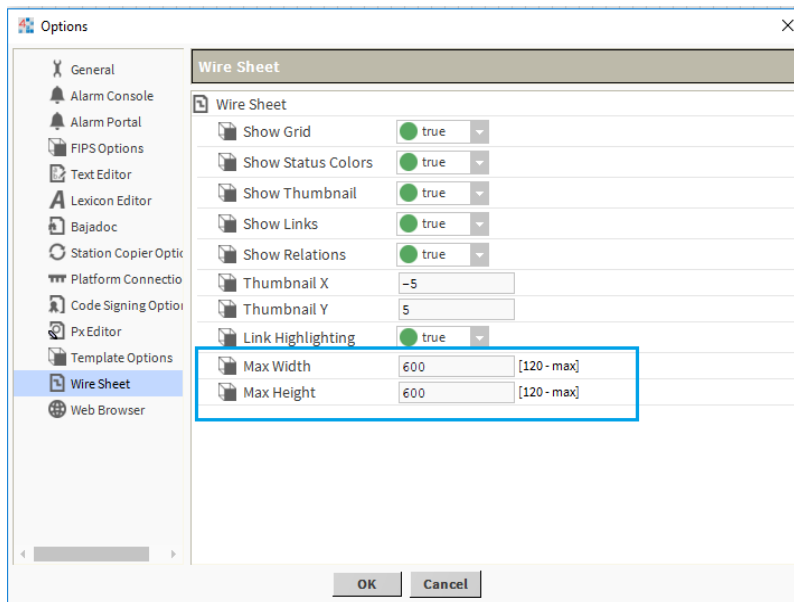
Zoom controls (buttons) appear on the Workbench tool bar when the **Wire Sheet** view is active.

- To enlarge the view, click the plus magnifying glass.
- To reduce the view, click minus magnifying glass
- To reset the view to its original size, click the empty magnifying glass.

Parent topic: [Wire Sheet object management](#)

Having to scroll vertically and horizontally in the wire sheet view can become tiresome. You can configure this view to restrict its size so that all objects remain in view. If your model is complicated, you may need to configure multiple wire sheets.

1. From the Workbench menu bar, click **Tools > Options**.
The **Options** window opens.
2. In the left pane of the **Options** window, click the **Wire Sheet** option.

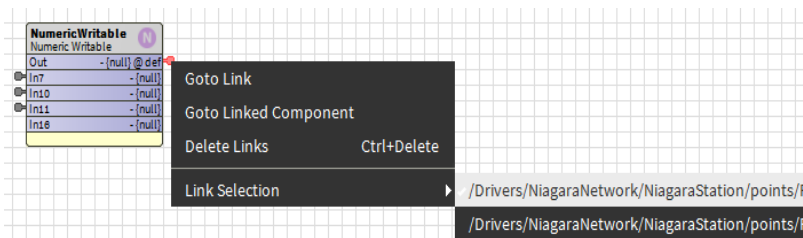


3. Enter a desired maximum value (in pixels) in the Max Width and Max Height fields and click **OK**.

Parent topic: [Wire Sheet object management](#)

Link knobs provide easy navigation.

Figure 1. Link Selection menu



- Off-view linking

Double-clicking on the knob hyperlink at the opposite end of the link, displays the **Wire Sheet** view with the linked knob highlighted.

- Goto Link command

Right-clicking on the knob displays a **GoToLink** command on the menu. Selecting this command hyperlinks to the component on the opposite end of the link, displaying that component's **Wire Sheet** view with the linked knob highlighted.

- Delete Links command

Selecting one or more off-view links (knobs) makes the **Delete Links** command available on the menu. This command removes all selected links and their associated off-view references. This is effective for multiple selected in-view and off-view (knob) links.

- Link selection

A **Link Selection** command is available from the menu when multiple links overlap on the wire sheet. Right-click on a link knob on the active wire sheet to select this command, then choose the desired link from the secondary menu.

Parent topic: [Wire Sheet object management](#)

Web **Wire Sheet** is an implementation of the **Wire Sheet** view that runs in your web browser and is accessible from your desktop or mobile device as part of the HTML5-based web interface. When you are logged in to the Niagara web interface via browser, the Web **Wire Sheet** is available for any component. The following gives you an overview of the Niagara 4.12 Web **Wire Sheet** user interface features.

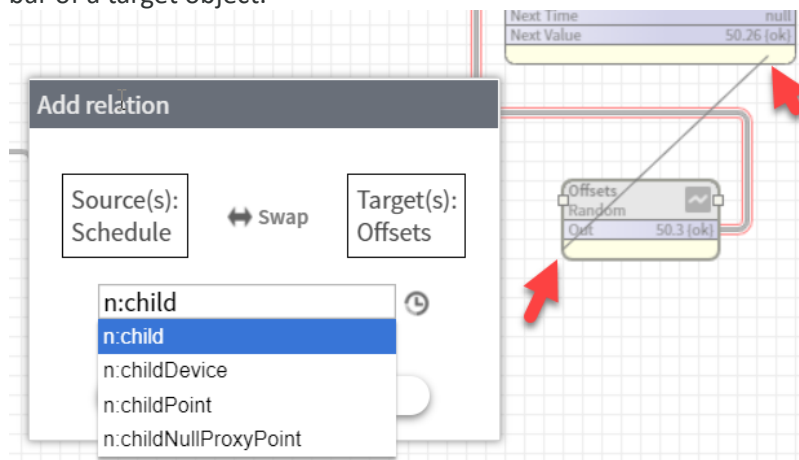
- [Linking and relating objects](#)
As of Niagara 4.12, the Web **Wire Sheet** offers enhanced features that facilitate and simplify the completion your tasks.
- [Navigating between off-view links](#)
Link knobs indicate that two or more object are linked but not located on the same Web **Wire Sheet**. By double-clicking the link knob, you can navigate to the connected off-view **Wire Sheet** object with the linked knob highlighted.

Parent topic: [Wire Sheet object management](#)

As of Niagara 4.12, the Web **Wire Sheet** offers enhanced features that facilitate and simplify the completion your tasks.

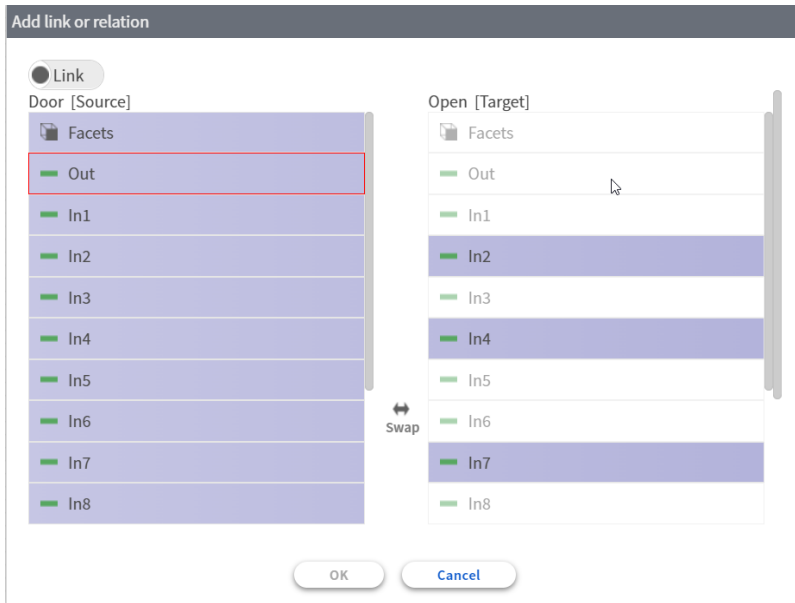
Relating objects using bottom bars

You can relate objects by dragging a wire from the bottom bar of a selected object to the bottom bar of a target object.



The **Add relation** window that opens allows you to define the source and target object as well as specify the relationship between source and target object.

You can link or relate by dragging a component from the palette, search side bar or Nav tree onto another object.



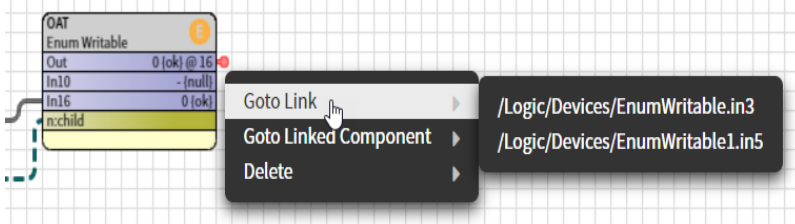
The **Add link or relation** window that opens allows you to choose between adding a link or adding a relation. The link/relate toggle remembers the setting of the previous action.

Parent topic: [Using Web Wire Sheet](#)

Link knobs indicate that two or more object are linked but not located on the same Web **Wire Sheet**. By double-clicking the link knob, you can navigate to the connected off-view **Wire Sheet** object with the linked knob highlighted.

Using the Goto Link command

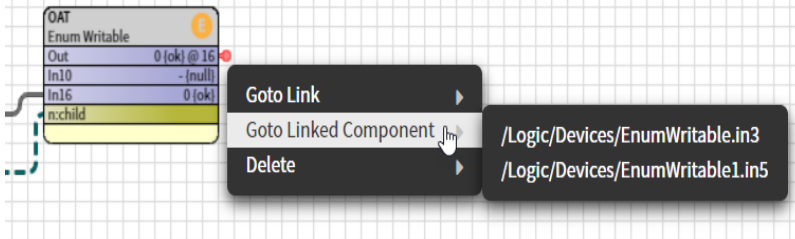
If you link from a slot of an object to multiple off-view objects, you can individually navigate between these objects by right-clicking the knob and selecting a link from the **GoTo Link** command menu. Selecting this command links you to the object on the other end of the link thereby displaying the object's **Wire Sheet** view with the linked knob highlighted.



Using the Goto Linked Component command

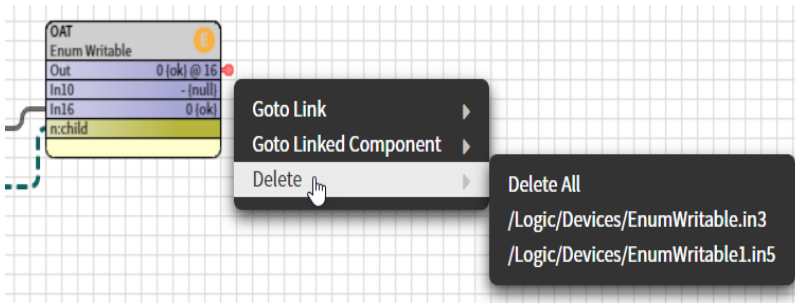
If you link from a slot of an object to multiple off-view components, you can individually navigate between these components by right-clicking the knob and selecting a link from the **GoTo Linked Component** command menu. Selecting this command links you to the component's property

sheet on the other end of the link thereby allowing you to make setting changes if needed.



Using the Delete command

If you linked from a slot of an object to multiple off-view objects, you can individually delete these object links by right-clicking the knob and selecting a link from the **Delete** command menu.



Parent topic: [Using Web Wire Sheet](#)

Any code that can run in a station, including program objects, program modules, provisioning robots, objects edited by the Batch Editor, and robots created by the Robot Editor must be signed so that each target station can verify that the code is trusted.

To sign code, you create a code-signing certificate in the Workbench User Key Store and sign it (or have it signed) using the private key of an intermediate or root CA certificate that resides in each station's User Trust Store. Then, at the time you compile each program object, module and robot, you sign it with the code-signing certificate.

When the code runs in a platform/station, the system verifies that it is trusted by comparing the signature in the code with the trusted signature in the intermediate or root CA certificate in the station's User Trust Store.

Additional features include signing a batch of code objects, signing the code contained in offline bogs, provisioning a job to install a code-signing certificate in the User Trust Store of multiple stations, signing legacy code when you recompile it, and signing code when migrating it from AX to N4.

- **[Code-signing warning and forced code signing](#)**
By default, program object signing is not currently (September, 2017) enforced, but will be enforced in a future version of the framework. The version in which program object signing will be enforced is yet to be determined.
- **[Creating a code-signing certificate](#)**
The system signs code objects using a code-signing certificate that is password protected. This procedure explains how to generate a code-signing certificate using **Workbench**. You may use a third-party tool to generate a code-signing certificate followed by importing it into your **Workbench User Key Store**. Such an imported certificate must have a code-signing set as its extended key usage. This is a standard certificate extension.
- **[Creating a CSR for the code-signing certificate](#)**
To verify that the code-signing certificate is trustworthy, it must be signed by the private key of an intermediate or root CA (Certificate Authority) certificate. While the system can sign code using a self-signed code-signing certificate, this practice is not recommended. The authenticity of a self-signed certificate cannot be verified by the target system. The root CA certificate used to sign your code-signing certificate may belong to your company, if it serves as its own CA, or it may belong to a trusted third-party CA, such as VeriSign or Thawte. Creating a CSR (Certificate Signing Request) is the first step in getting your code-signing certificate appropriately signed.
- **[Signing a certificate](#)**
Signing a certificate is the job of a CA (Certificate Authority). A variety of certificate-signing software tools are available. You are not required to use the framework and **Workbench** to sign certificates. This procedure documents how to sign certificates. It applies to companies who serve as their own CA. In a large installation, you use your root CA certificate to sign any intermediate certificates and the intermediate certificates to sign your server and code-signing certificates. In a small installation, you may use your root CA certificate to sign all certificates.
- **[Importing the signed certificate back into the User Key Store](#)**
Signing a certificate creates a .pem file, which is only intended for importing back into the **User Key Store** that contains the original certificate with the matching private key. For a server certificate this is the platform/station **User Key Store** that originally created the certificate and CSR. For an intermediate certificate or a code-signing certificate, this is, most likely, the **Workbench User Key Store** on the secure computer, which you use to sign other certificates.
- **[Configuring Workbench to sign program objects](#)**
To begin signing program objects, the code-signing certificate you created must be selected as the signing tool.

- **[Supplying the private key password](#)**
For each new Workbench session, the first time that the system signs a new program object you must supply the code-signing certificate's private key password. Thereafter, the system caches this password and you are not required to enter it again until you restart Workbench and build a new program object.
- **[Approving an exception](#)**
The station verifies the signature on each program object to ensure that the certificate chain is trusted. If the root CA certificate used to sign the code-signing certificate is in the station's **User Trust Store**, the code runs without further intervention. If the code signing-certificate is self-signed, or the root CA certificate is missing, the station reports an error, after which you may approve an exception.
- **[Approving an exception in an Editor](#)**
Signed code that is not trusted generates a warning message in the Program Editor, Batch and Robot Editors. This procedure documents how to approve an exception when the Program Editor returns this warning.
- **[Installing a certificate](#)**
If the **System Trust Stores** in your remote stations already contain the root CA certificate from the CA (Certificate Authority) that signed your intermediate, server or code-signing certificates, you do not need to run a provisioning job. If your company serves as its own CA, you must install a root CA or intermediate certificate in the **User Trust Stores** of each platform/station that serves as a client. Installing this certificate can be useful before running a signed provisioning robot on several stations.
- **[Recompiling and signing program objects](#)**
To sign all existing unsigned program objects, use the Program Recompile Tool view on the **Program Service**. This recompiles all program objects on the station, and, if code signing is configured, signs them using the code-signing certificate.
- **[Batch signing code in offline bogs](#)**
You can sign code in offline bogs just by using the editor and visiting each program object individually. Or you can point to a directory containing multiple stations (or offline bog files), and use this command-line tool to recompile and sign all the code at once as a batch.
- **[AX to N4 migration tool and code signing](#)**
When running the AX to N4 Migration Tool in 4.3 or later, any program object encountered is signed if a code-signing certificate has been configured in Workbench.
- **[Code-signing troubleshooting](#)**
This topic summarizes common errors when configuring code signing.

By default, program object signing is not currently (September, 2017) enforced, but will be enforced in a future version of the framework. The version in which program object signing will be enforced is yet to be determined.

Loading or running any objects you have created without signatures causes the following warning to appear in Application Director:

```
WARNING [date] Program not signed. The ability to run unsigned
programs will be removed in a future release.
```

where `[date]` is replaced by today's date and time.

To override the current default, and require program objects to be signed when they run, configure your station to run with the system property `program.requireSigning=true`. You configure this property in the `system.properties` file. When set to true, the system fails to load any unsigned program object or any signed object with a certificate that is not trusted. Program modules are exceptions to this rule.

When a program object with a certificate that is not trusted fails to load, the system adds the certificate to the User Trust Store placing a red shield with a white X to the left of the certificate row. To create an exception for the certificate, click the **Approve** button. Consider this as a temporary convenience that enables you continue to use the system before your code-signing certificate is officially signed by an intermediate certificate or the root certificate of a Certificate Authority.

Parent topic: [Signing program objects](#)

The system signs code objects using a code-signing certificate that is password protected. This procedure explains how to generate a code-signing certificate using Workbench. You may use a third-party tool to generate a code-signing certificate followed by importing it into your Workbench User Key Store. Such an imported certificate must have a code-signing set as its extended key usage. This is a standard certificate extension.

You are working in Workbench running on a Supervisor or engineering PC.

1. Click **Tools > Certificate Management**.
The **Certificate Management** view opens.
2. Click the **New** button at the bottom of the view.
The **Generate Self Signed Certificate** window opens.

3. Fill in the properties.
In addition to the required properties, define your Locality (L) (city) and State/Province (ST). Without these properties the system reports an error message.

Country Code is a two-character ISO code (refer to the ISO CODE column at countrycode.org).

Choose Code Signing for Certificate Usage.

The **OK** button activates when all required information is provided.

4. To create the certificate, click **OK**.

The **Private Key Password** window opens.



5. Enter a strong password and click **OK**.
Your password must be at least 10 characters long. At least one character must be a digit; one must be lower case; and one must be upper case.
The system submits the certificate for processing in the background.
6. When the certificate has been created, click **OK**.
This self-signed certificate appears as a row in the User Key Store tab, identified by a yellow shield icon (🛡️).

Protect this certificate and password! If someone steals your certificate and knows your password, they could damage your operation by using your certificate to sign their own malicious code.

Parent topic: [Signing program objects](#)

To verify that the code-signing certificate is trustworthy, it must be signed by the private key of an intermediate or root CA (Certificate Authority) certificate. While the system can sign code using a self-signed code-signing certificate, this practice is not recommended. The authenticity of a self-signed certificate cannot be verified by the target system. The root CA certificate used to sign your code-signing certificate may belong to your company, if it serves as its own CA, or it may belong to a trusted third-party CA, such as VeriSign or Thawte. Creating a CSR (Certificate Signing Request) is the first step in getting your code-signing certificate appropriately signed.

You are using Workbench running on a PC.

1. If necessary, navigate to the Certificate Management view and select the code-signing certificate.
The view opens to the User Key Store.
2. Select the code-signing certificate and click the **Cert Request** button at the bottom of the view.
3. Confirm that the certificate properties are correct and click **OK**.
The **Certificate Manager** prompts you for the private key password.
4. Enter the password you assigned to the code-signing certificate and click **OK**.
The system displays the certManagement folder from which to choose the location to store the CSR.

The Alias for the certificate is used as the file name of the CSR. The extension is .csr.

5. Use the default folder, or select a different folder in which to store the CSR and click **Save**.
The system displays, CSR generation complete.
6. To confirm completion, click **OK**.
7. If an external CA, such as VeriSign or Thawte, will sign your code-signing certificate, follow the CSR submission procedure as required by the CA.
The CA verifies that you are who you claim to be, that the certificate is for your organization, and other important information. They then return a signed code-signing certificate (.pem file) to you (usually by email).

Parent topic: [Signing program objects](#)

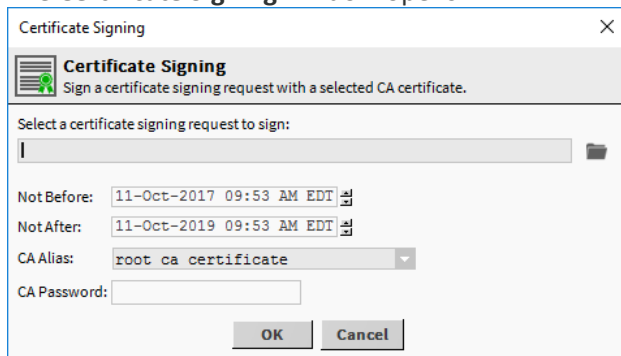
Signing a certificate is the job of a CA (Certificate Authority). A variety of certificate-signing software tools are available. You are not required to use the framework and Workbench to sign certificates. This procedure documents how to sign certificates. It applies to companies who serve as their own CA. In a large installation, you use your root CA certificate to sign any intermediate certificates and the intermediate certificates to sign your server and code-signing certificates. In a small installation, you may use your root CA certificate to sign all certificates.

- You are working in Workbench on a physically and electronically secure PC that is never connected to the Internet, and is used exclusively to sign certificates.
- The root CA or intermediate certificate that will do the signing is in the Workbench User Key Store.
- You know the password of the CA signing certificate (root or intermediate) that will sign the certificate(s).
- You have one or more CSR files (signing requests) ready to sign.

Note: To ensure network security, always sign certificates using Workbench on a computer that is disconnected from the Internet and from the company LAN. Maintain this computer in a physically secure location.

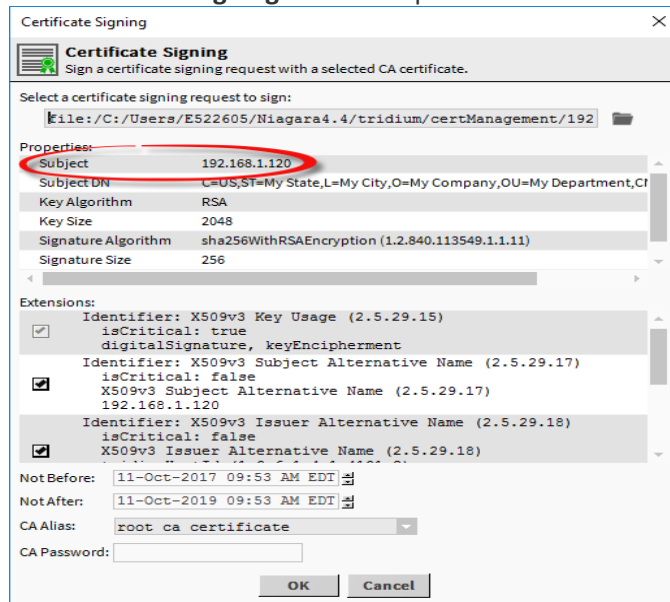
1. In Workbench on your physically and electronically secure (and never connected to the Internet) PC that is used exclusively to sign certificates, click **Tools > Certificate Signer Tool**.

The **Certificate Signing** window opens.



2. Click the folder icon, locate, and open the CSR for the certificate you wish to sign.

The **Certificate Signing** window expands to show certificate details.



3. Confirm that this is the correct CSR by checking the **Subject**.
4. Select the date on which the certificate becomes effective (Not Before) and the date after which it expires (Not After).
5. For CA Alias, use the drop-down list to select the certificate (root or intermediate) whose private key will sign this certificate.
6. Supply the CA certificate's password and click **OK**.

Signing is done by the private key of the root or intermediate certificate.

The same file folder, C: /Users/ [username] /Niagara4.x/certManagement, displays with the file name (extension: .pem) filled in for you.

You may modify this file structure to aid in the management of these files.

7. To complete the signing, click **Save**.
8. Copy the signed certificate .pem file to a thumb drive and import it back into the User Key Store of the computer that created the certificate and generated the CSR.

You can repeat this procedure for each CSR.

Note: In Niagara there is added support for bulk certificate signing. For more details refer to the "Signing multiple certificates" topic.

Parent topic: [Signing program objects](#)

Signing a certificate creates a .pem file, which is only intended for importing back into the **User Key Store** that contains the original certificate with the matching private key. For a server certificate this is the platform/station **User Key Store** that originally created the certificate and CSR. For an intermediate certificate or a code-signing certificate, this is, most likely, the Workbench **User Key Store** on the secure computer, which you use to sign other certificates.

You have the signed .pem files. The focus is on the User Key Store in the appropriate stores location (Workbench or platform/station).

1. Click **Import**.

2. Locate and select the signed certificate's .pem file (the output of the certificate signer or the .pem file you received from a third-party CA) and click **Open**.
The **Certificate Import** window opens.
3. Confirm that you are importing the correct certificate and click **OK**.

If the Alias of the certificate you are importing is not the same as the Alias of the certificate you are replacing, the system prompts you for the Alias of the certificate to replace.

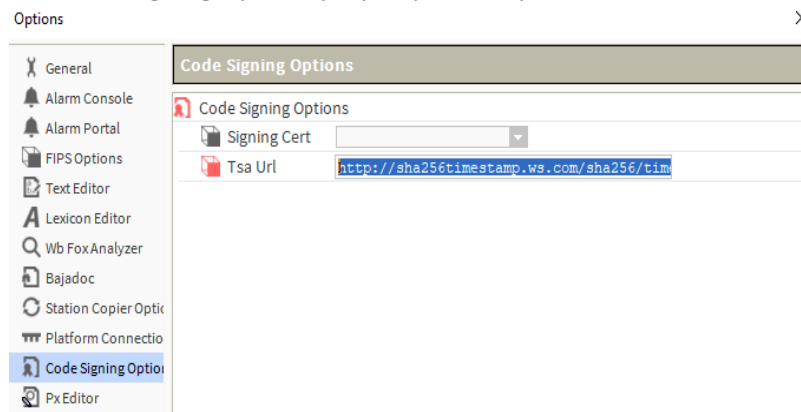
4. If needed, enter the Alias and click **OK**.
The green shield icon (✓) replaces the yellow shield icon (⚠) next to the certificate Alias in the **User Key Store** tab.
5. Using the operating system, delete the .pem file(s) from the secure Workbench computer.

Parent topic: [Signing program objects](#)

To begin signing program objects, the code-signing certificate you created must be selected as the signing tool.

The code-signing certificate exists.

1. In Workbench, click **Tools > Options**, and click **Code Signing Options**.
The Code Signing Options property sheet opens.



2. From the Signing Cert drop-down list, select your code-signing certificate.
The drop-down menu lists only certificates whose key usage is designated as Code Signing. If there is only one code-signing certificate in your User Key Store, this will be the only option.
3. If desired, set the Tsa Url (Timestamp authority) to a valid timestamp authority.
This property defaults to the URL. Time stamping a program object signature establishes trust even after a code-signing certificate expires. If your program object signatures are not time-stamped, they cannot be validated past the expiration date of the code-signing certificate.

Note: In framework versions 4.2 and 4.3, Tsa Url defaults to the now unavailable Geotrust TSA. In version 4.4, support was added for SHA-256 timestamps and the default was updated to the URL. If you are using versions 4.2 or 4.3, the recommended setting for Tsa Url is:

<http://timestamp.digicert.com>

If you leave the default TSA in 4.2 and 4.3 set to Geotrust TSA, code signing will not work and you will run into errors due to the Geotrust TSA going off line.

4. To complete the configuration, click **OK**.

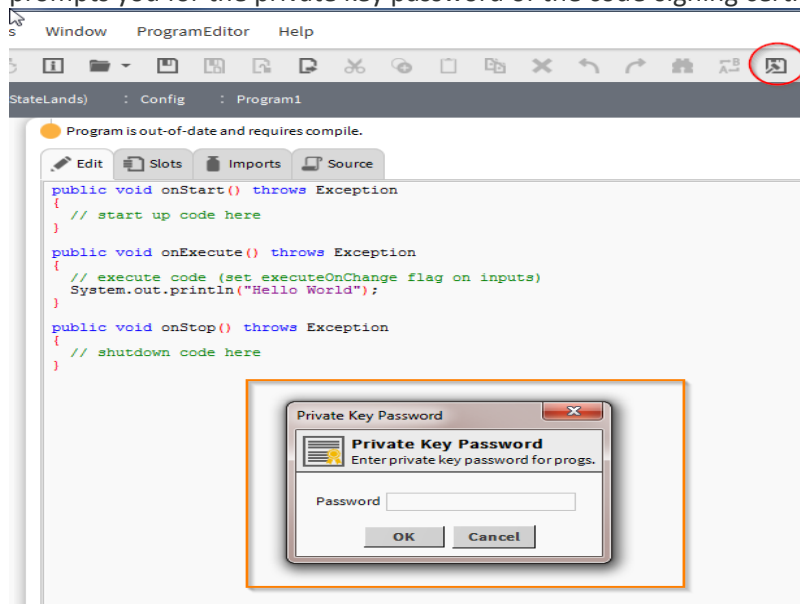
While this configuration procedure works if your code-signing certificate is self-signed or signed by a trusted intermediate or root CA certificate, using the latter is preferred. In fact, without revisiting this configuration procedure, you could set up a self-signed code-signing certificate, and sign it later. However, if you do this, you must re-sign any code that you signed prior to getting your code-signing certificate signed.

Parent topic: [Signing program objects](#)

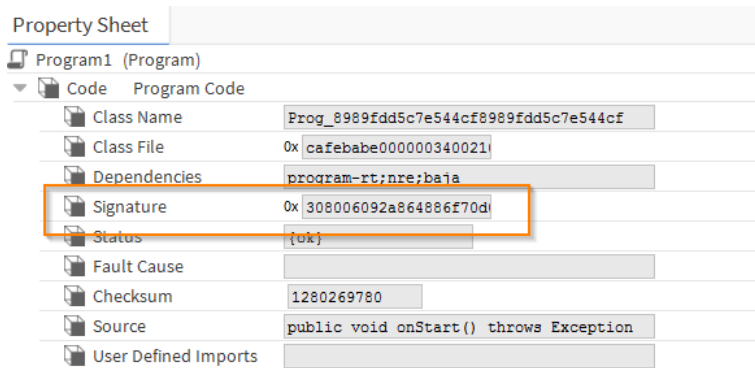
For each new Workbench session, the first time that the system signs a new program object you must supply the code-signing certificate's private key password. Thereafter, the system caches this password and you are not required to enter it again until you restart Workbench and build a new program object.

1. Compile a Program Object.

The first time a program object is built during the current Workbench session, the system prompts you for the private key password of the code-signing certificate.



2. Enter the password you created when generating the certificate.
3. Verify that the new signature property on Code is displayed.

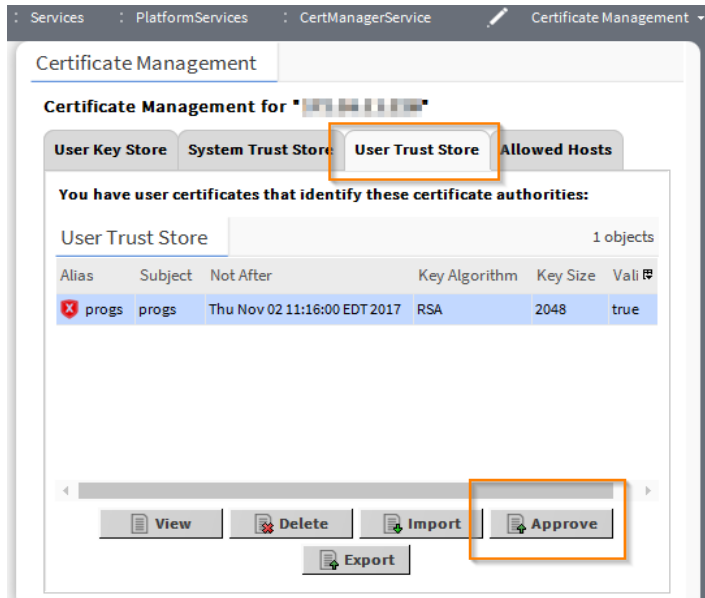


Parent topic: [Signing program objects](#)

The station verifies the signature on each program object to ensure that the certificate chain is trusted. If the root CA certificate used to sign the code-signing certificate is in the station's User Trust Store, the code runs without further intervention. If the code signing-certificate is self-signed, or the root CA certificate is missing, the station reports an error, after which you may approve an exception.

Your program object has been signed by a self-signed code-signing certificate, or the code-signing certificate is not trusted.

1. The signed program runs at least once.
The system displays an error message and adds the certificate to the station's User Trust Store.



The red shield with the white X in the Certificate Management view, User Trust Store tab, indicates the untrustworthy condition of the code-signing certificate.

2. If you know that the certificate is safe, approve an exception by clicking the **Approve** button.

This exception is similar to accepting the self-signed certificate when you initially log in to a platform or station. The ability to approve this exception is provided for convenience. Your

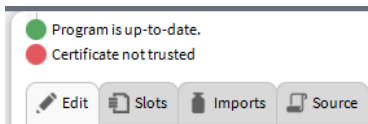
system is much more secure when you follow the recommended practice of signing code-signing certificates with the private key of your company's or a third party's root CA certificate, which is in the station's User Trust Store.

Parent topic: [Signing program objects](#)

Signed code that is not trusted generates a warning message in the Program Editor, Batch and Robot Editors. This procedure documents how to approve an exception when the Program Editor returns this warning.

Your code has been signed by a self-signed code-signing certificate, or the root CA certificate used to sign the code is not in the User Trust Store.

1. Open a program in the Program Editor.



A red warning indicator (circle) in the Program Editor indicates that the code-signing certificate used to sign the code is not trusted.

2. If you are confident that the code-signing certificate can be trusted, approve an exception by clicking the **Install Certificate** button.



The system approves the certificate.

When running the Batch Editor or Robot Editor from the **Program Service**, the system automatically prompts you to approve an exception for the code-signing certificate.

Approving exceptions is a convenience to allow you to continue working without interruption. To ensure more robust security, always sign your code-signing certificate with a root CA certificate (your company's or the root certificate provided by a third-party CA, such as VeriSign or Thawate), and make sure that the root CA certificate is in each station's User Trust Store.

Parent topic: [Signing program objects](#)

If the **System Trust Stores** in your remote stations already contain the root CA certificate from the CA (Certificate Authority) that signed your intermediate, server or code-signing certificates, you do not need to run a provisioning job. If your company serves as its own CA, you must install a root CA or intermediate certificate in the **User Trust Stores** of each platform/station that serves as a client. Installing this certificate can be useful before running a signed provisioning robot on several stations.

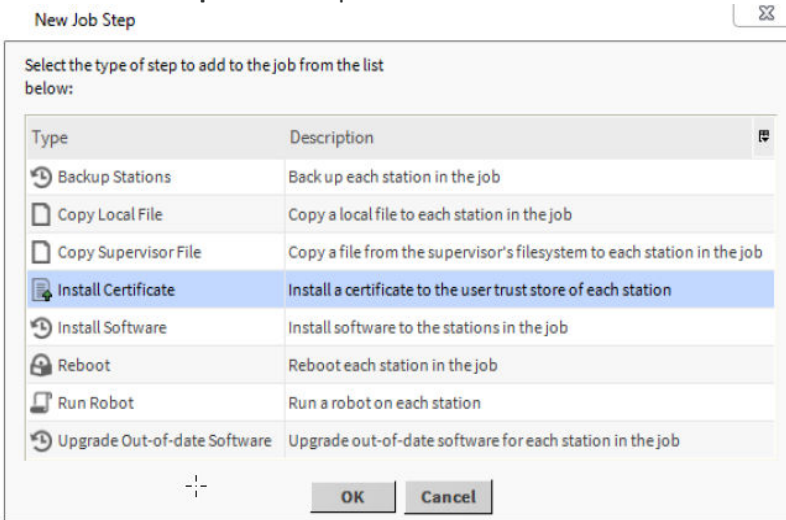
The **BatchJobService** is available under **Services**. The **ProvisioningNwExt** component is available under the **NiagaraNetwork**. The **Niagara Network Job Builder** (one-time) or **Niagara Network Prototype View** (reoccurring job) is open.

This is a recurring provisioning job that uses the **NiagaraNetworkJobPrototype**.

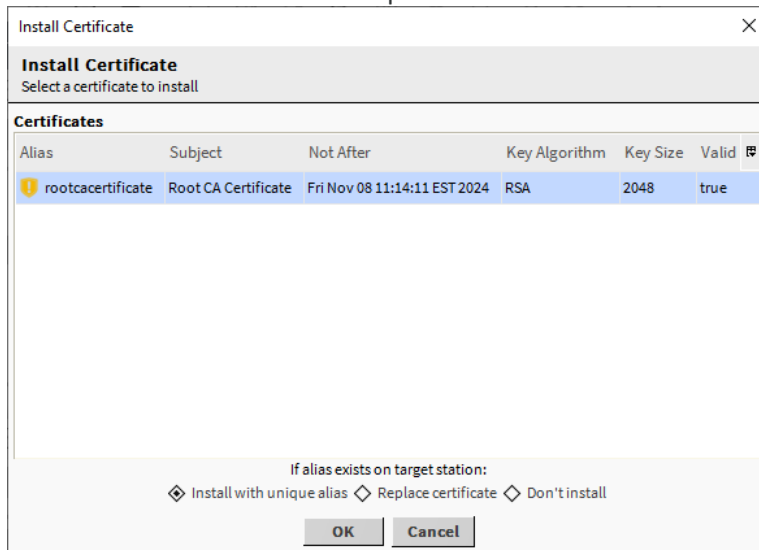
1. On your Supervisor, expand **Config > Services > PlatformServices**, double-click **CertManagerService** and click the **User Trust Store** tab.

Note: You cannot complete this procedure if you import the certificate into the Workbench **User Trust Store** of your Supervisor PC.

2. Click the **Import** button, navigate to the location on the thumb drive that contains the root CA certificate and click **Open**.
3. Confirm that the Subject of the certificate identifies it as the root CA certificate and click **OK**.
The system imports the certificate in preparation for the provisioning job.
4. Navigate to the location in the station where you manage provisioning jobs.
5. Drag a **NiagaraNetworkJobPrototype** component to this location and name the component something like, "Root CA certificate provisioning."
6. Double-click the new component.
The **Niagara Network Prototype View** opens.
7. In the top pane, **Provisioning steps to run**, click  (Add).
The **New Job Step** window opens.



8. Select **Install Certificate** and click **OK**.
The **Install Certificate** window opens.

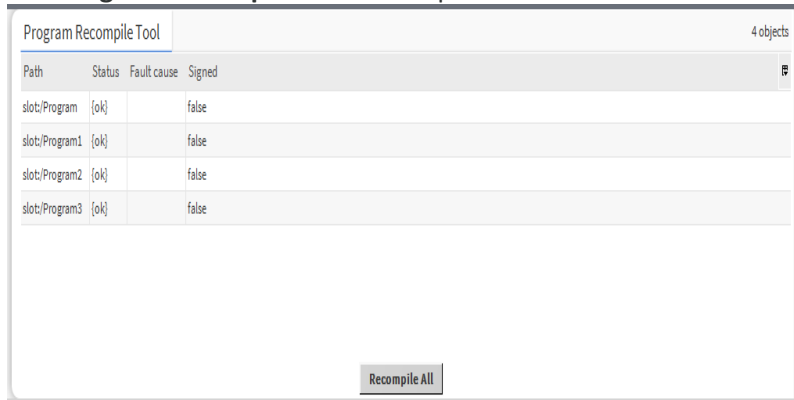


9. To complete the installation, select the root CA certificate and click **OK**.
10. Define the stations to include in the job.
The system copies the certificate from the Supervisor station's **User Trust Store** to the **User Trust Stores** of the other clients.

Parent topic: [Signing program objects](#)

To sign all existing unsigned program objects, use the Program Recompile Tool view on the **Program Service**. This recompiles all program objects on the station, and, if code signing is configured, signs them using the code-signing certificate.

1. Access the **Program Service**.
2. Click **Program Recompile**.
The **Program Recompile Tool** view opens.



This view lists every program object in the station.

3. Click **Recompile All**.
The system recompiles and signs every program object using the currently-selected code-signing certificate.

Parent topic: [Signing program objects](#)

You can sign code in offline bogs just by using the editor and visiting each program object individually. Or you can point to a directory containing multiple stations (or offline bog files), and use this command-line tool to recompile and sign all the code at once as a batch.

nre program syntax

```
nre program:com.tridium.program.ui.RecompileTool
usage:
RecompileTool <dir|bogfile> [flags]
parameters:
dir|bogfile      Directory containing bog file(s),
or individual bog file.
optional flags:
-alias:<arg>     Alias of a code signing certificate
                  in user key store to sign programs with.
                  Defaults to workbench code signing
                  options if not provided.
-password:<arg>  Private key password for the
                  signing certificate. Will be prompted
                  if not provided.
-tsaUrl:<arg>    Time stamp authority url to use
                  for timestamping program signatures.
                  Defaults to workbench code signing options
                  if not provided.
```

Parent topic: [Signing program objects](#)

When running the AX to N4 Migration Tool in 4.3 or later, any program object encountered is signed if a code-signing certificate has been configured in Workbench.

When the migrator encounters the first program object that needs signing, it prompts you for your code-signing certificate password.

Parent topic: [Signing program objects](#)

This topic summarizes common errors when configuring code signing.

Code signing is not working.

If you are using framework version 4.2 or 4.3 check the default **Tsa Url** property in the **Code Signing Options** view. This property defaults to Geotrust TSA, which is not a valid TSA (Timestamp Authority) option.

Parent topic: [Signing program objects](#)

You can customize your Workbench interface as well as many of the settings in the Workbench environment. Some settings require an acknowledgement via the **OK** button, while others are invoked immediately and saved automatically on exiting Workbench. For example, if you exit Workbench with four tabbed windows in the view pane, Workbench displays the same four tabs the next time you open it.

- [About Workbench themes](#)
Workbench themes customize the user interface.
- [Replacing the Loading Splash Screen](#)
A default splash animated .gif appears while Workbench is loading. You can remove or replace this .gif.
- [Configuring the web browser allowlist \(system.properties\)](#)
In Niagara, using Workbench in a web browser is governed by the use of a allowlist, which adds security and customizability. The allowlist specifies exactly to which web addresses Workbench can navigate. Consequently, you can no longer use Workbench to freely browse external web addresses unless you configure the allowlist to allow it.
- [Editing the PxEditor New popup submenu](#)
You can change the menu items that display on the context-sensitive popup menu by editing the files under the Workbench subfolder. For example, the following image shows the newWidgets.bog file that has been edited so that, when working in the **PxEditor** view, a boundLabel widget appears as one of the options in the popup menu under the **New** submenu.
- [Changing the time format using the Lexicon](#)
- [Changing the time format using Facets](#)

Workbench themes customize the user interface.

A theme is a predefined collection of design elements that determine the way an application appears on screen. Within the framework, a theme is a module that controls the appearance of Workbench on the local computer. It defines fonts and colors, as well as the icons used in the display. Choosing a different theme affects only the framework's appearance, there is no other impact.

You can customize your Workbench display by selecting the theme that you prefer. Workbench provides standard built-in themes as part of the default application environment:

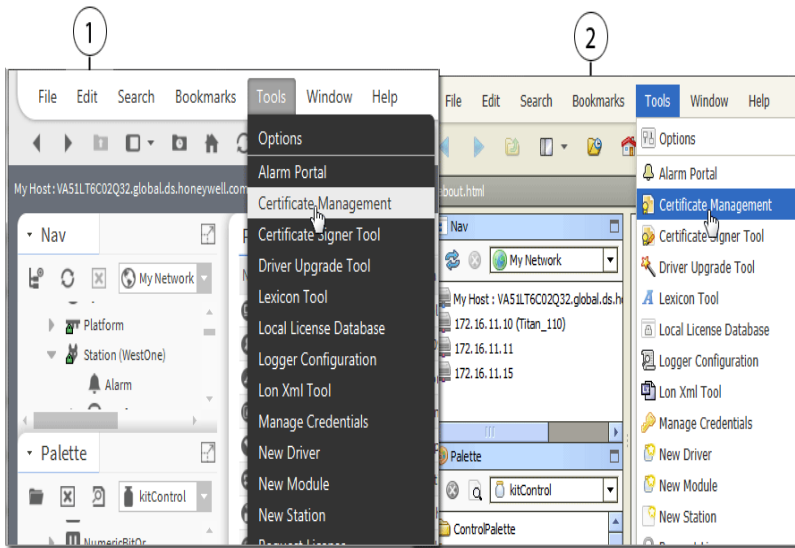
- Zebra (shown below as theme 1 on the left)

This theme has a low contrast, grayscale coloring and is the default theme.

- Lucid (shown below as theme 2 on the right)

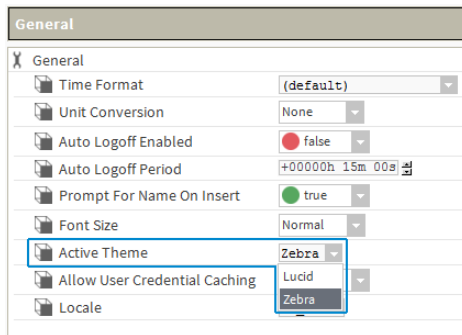
Displays a blue and gray color scheme.

Figure 1. Zebra (left) and Lucid (right)



To change the Workbench theme, click **Tools > Options** to see the active theme in the **General** tab. Click the **Active theme** drop-down arrow and click on a different theme option to select it as shown above. For the theme change to take effect, exit Workbench and restart it.

Figure 2. Active Theme drop-down in General settings



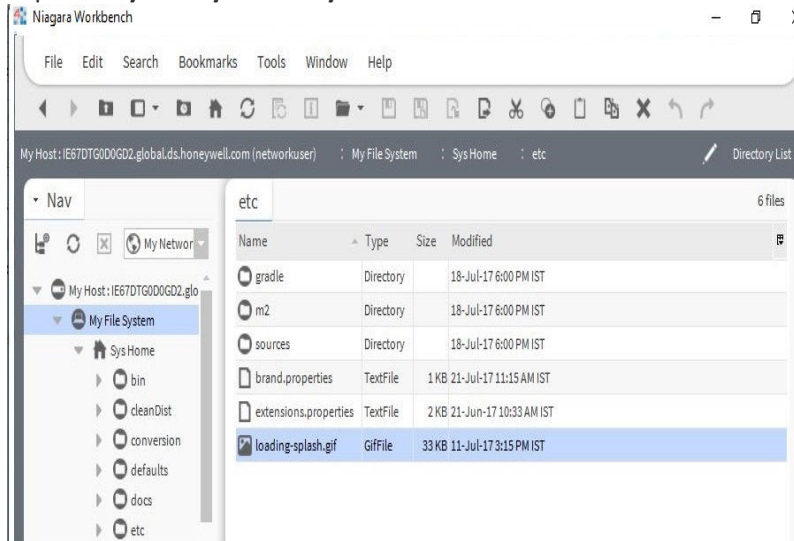
Note: To save changes made in Workbench Options, such as a theme change, you must close/exit Workbench using **File** menu options or click the window's close box. Closing Workbench in the console does not save your changes.

Parent topic: [Customizing the Workbench environment](#)

A default splash animated .gif appears while Workbench is loading. You can remove or replace this .gif.

You are working in Workbench. You have a file named loading-splash.gif with which to replace the default file. The replacement file must be named loading-splash.gif.

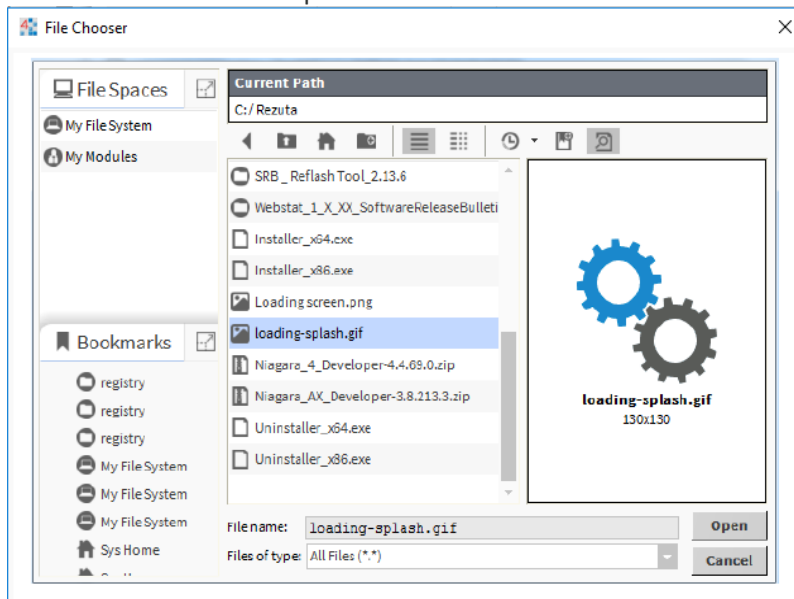
1. Expand **My File System > Sys Home > etc** in the Nav



2. To view the default .gif, double-click loading-splash.gif
The .gif appears in the **Image Viewer**.



3. To replace this .gif, first delete the existing loading-splash.gif file.
4. To configure Workbench with no loading splash screen, delete the .gif file.
5. To select the new .gif file, double-click on the **etc** folder in the Nav tree, right-click the **Directory List** table and select the **Copy From** option in the list.
A **File Chooser** window opens.



6. Locate the new .gif on your local drive, select it and click **Open**.
The new .gif replaces the default .gif.

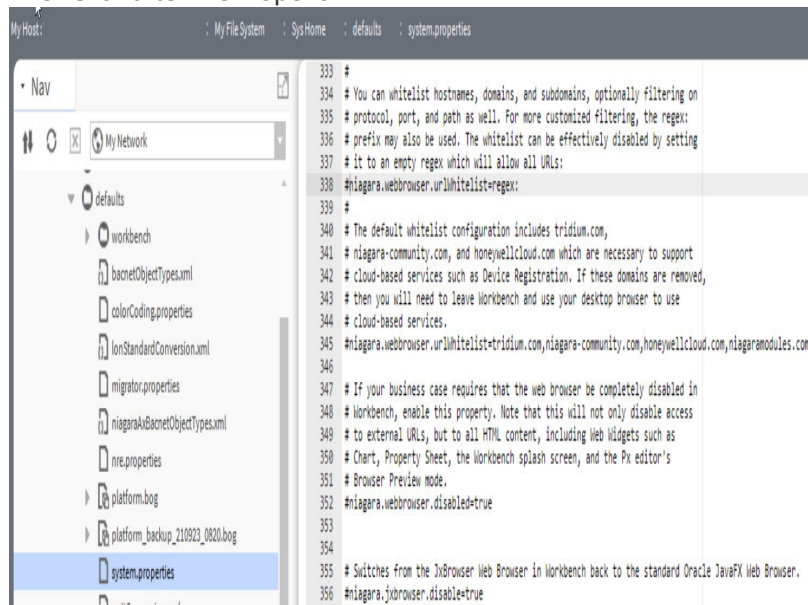
Parent topic: [Customizing the Workbench environment](#)

In Niagara, using Workbench in a web browser is governed by the use of a allowlist, which adds security and customizability. The allowlist specifies exactly to which web addresses Workbench can navigate. Consequently, you can no longer use Workbench to freely browse external web addresses unless you configure the allowlist to allow it.

You are aware of any security implications or organizational policies before allowing Workbench to browse the web unrestricted.

You configure the whitelist with approved hostnames, domains, and subdomains, optionally filtering on protocol, port, and path. For more customized filtering, you may also use the regex: prefix. If desired, you can completely disable the web browser.

1. To navigate to the !defaults/system.properties file, expand **My Host > My File System > Sys Home > defaults** in the **Nav** tree and double-click **system.properties**. The **Text Editor** view opens.



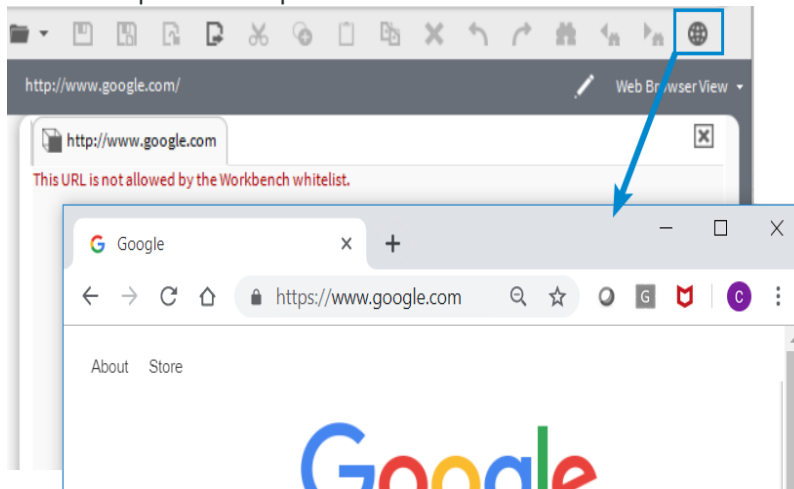
2. Scroll to the `niagara.webbrowser.urlWhitelist` property.
3. Enter a comma-separated list of URL patterns that you have decided are acceptable for Workbench to navigate to.

To allow navigation to:	Enter the following value:
Web pages served by a particular hostname such as your localhost	<code>niagara.webbrowser.urlWhitelist=hostname</code>
Web server on any subdomain of a given domain such as <code>domain.com</code> , <code>www.domain.com</code> , <code>subdomain.domain.com</code>	<code>niagara.webbrowser.urlWhitelist=domain.com</code>
Specific subdomain such as allowing <code>www.domain.com</code> , but not <code>domain.com</code>	<code>niagara.webbrowser.urlWhitelist=www.domain.com</code>
Localhost but using only the specified protocol and port number, for example, using port 8088	<code>niagara.webbrowser.urlWhitelist=https://localhost:8088</code>
Any URL at <code>domain.com</code> , but filtering with a partial path such as <code>/public/</code>	<code>niagara.webbrowser.urlWhitelist=domain.com/public/</code>

To allow navigation to:	Enter the following value:
Additional customizing is possible using regex syntax. The regex will match on any substring of the URL. For example, <code>regex:a</code> would match any URL that contains the letter “a”. Similarly, to match on any file ending in <code>.htm</code> at <code>domain.com</code> , but no other files	<code>niagara.webbrowser.urlWhitelist=regex:domain.com/.*/*.htm\$</code>
To specify multiple URL patterns, enter them in a comma-separated list	For example: <code>niagara.webbrowser.urlWhitelist=localhost,central.com,bacnet.org</code>
To effectively disable the whitelist, set it to an empty regex, which will match on any URL. Workbench can then be used to navigate to any URL. To set the property to an empty regex, enter:	<code>niagara.webbrowser.urlWhitelist=regex:</code>
To disable the web browser altogether, set the <code>niagara.webbrowser.disabled</code> property. Disabling the web browser, not only disables access to external URLs, but to all HTML content including the Workbench splash screen, the Px Editor Browser Preview mode, and all Web Widgets such as Property Sheet and Web Chart. To set the property enter:	<code>niagara.webbrowser.disabled=true</code>
Other configuration options:	
Starting in Niagara 4.10u7, hyperlinks to certain ORDs containing queries may resolve to an optimized ORD form for a given result, which can sometimes avoid hyperlink ORDs that grow too large. For example, Niagara may replace a query ORD with a (more efficient) Nav ORD that resolves to the same result when the hyperlink is processed. The main side effect of this optimization is that the path bar (or URL in the browser) may look different (uses the optimized Nav ORD form) for the same result. To disable this optimization and side effect, set this system property to true.	<code>niagara.hyperlink.ord.substitution.disabled=false</code>
Lockout settings for failed attempts to update password due to incorrect current password. The default value is 5 minutes, 5 times within 1-minute window and in seconds.	<code>niagara.updatePassword.lockoutWindowUpdatePasswordFailed</code> <code>niagara.updatePassword.lockoutPeriodUpdatePasswordFailed</code> <code>niagara.updatePassword.maxBadAttemptCurrentPasswordFailed</code>
This system property, when set to true, will configure Bouncy Castle JSSE to include server name parameters in the TLS handshake for outgoing (client) connections. This is required for FOXS connections that must traverse a reverse proxy that routes requests to hosts based on the SNI extension contained in the ClientHello message. If a target host is specified by IP address, and this property is set to true, a reverse DNS lookup may be used to obtain the hostname. Note that if the name service is not trustworthy, enabling reverse name lookup may be susceptible to man-in-the-middle attacks.	<code>org.bouncycastle.jsse.client.assumeOriginalHostName=</code>
Niagara 4.14 introduces a Web CategoryBrowser feature that can be used to view and modify categories from the browser. This system property can be turned on to hide this view.	<code>niagara.hideWebCategoryBrowser=true</code>

To allow navigation to:	Enter the following value:
This system property specifies a ';' separated list of Signature Algorithms, and is used to determine if a certificate chain's root certificate should be excluded when presented to a client during a TLS handshake. If a certificate chain root certificate has a key purpose of Certificate Authority, and it's signature algorithm matches a value specified in this list, it will be removed from the certificate chain sent to a client in the ServerHello message.	<code>niagara.web.excludeRootCAFromCertChainForSigAlgs=SHA</code>
These properties allow you to increase the size limits of the various key store types from the default value of 500. Caution: Ensure that a larger value does not overload the Niagara daemon or station.	<code>niagara.crypto.maxKeyStoreEntries=500</code> <code>niagara.crypto.maxUserTrustStoreEntries=500</code> <code>niagara.crypto.maxExemptionStoreEntries=500</code>

- On completion, save your configuration changes.
- For your changes to take effect, exit Workbench by selecting **File > Exit** and restart it.
- To use the allowlist, navigate to a URL using Workbench.
- If the site is not allowed by the allowlist, click the **Open In Desktop Browser** command (🌐) in the toolbar to access the site in your desktop web browser. The desktop browser opens.

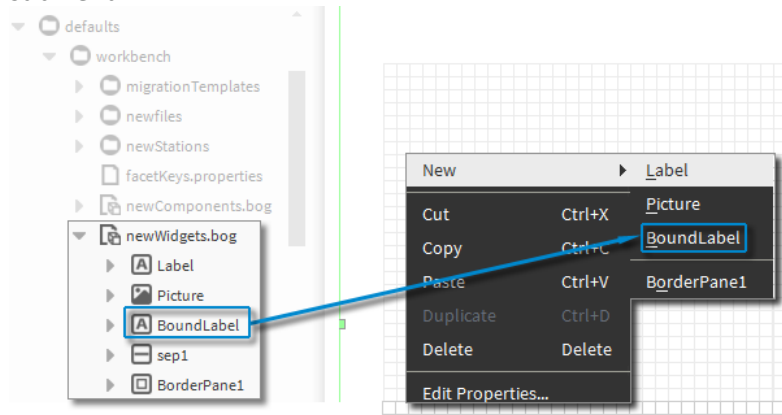


Parent topic: [Customizing the Workbench environment](#)

You can change the menu items that display on the context-sensitive popup menu by editing the files under the Workbench subfolder. For example, the following image shows the `newWidgets.bog` file that has been edited so that, when working in the **PxEditor** view, a `boundLabel` widget appears as one of the options in the popup menu under the **New** submenu.

- In the Nav tree under **My File System**, expand `SysHome/defaults/workbench/newWidgets.bog`.
- Open the kitPx palette.
- Drag the `BoundLabel` widget onto the `newWidgets.bog` file.

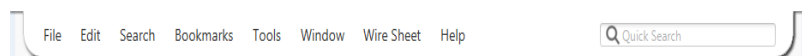
Adding a widget to the newWidgets.bog file adds the item to the **PxEditor New** popup submenu.



Parent topic: [Customizing the Workbench environment](#)

This topic introduces the menu that appear in the Workbench menu bar:

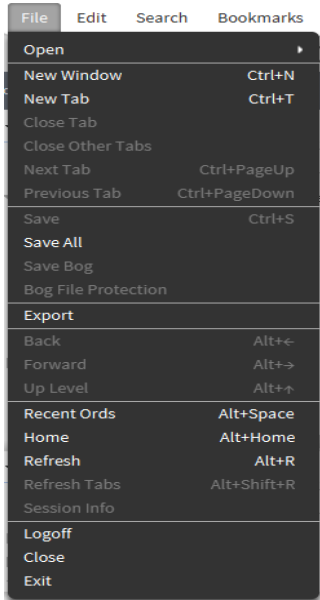
Figure 1. Menu bar



- File menu
- Edit menu
- Search menu
- Bookmarks menu
- Tools menu
- Window menu
- Px Editor menu
- History Ext Manager menu
- Help menu
- [About the File menu](#)
This menu in the menu bar provides a set of file-related options.
- [About the Edit menu](#)
This menu supports standard edit options.
- [About the Search menu](#)
This menu provides search functionality.
- [About the Bookmarks menu](#)
This menu customize the interface for your specific needs.
- [About the tools menu](#)
This section describes some of the standard tools available in Workbench. These tools provide views that are designed to facilitate specific tasks – from managing credentials to monitoring alarms. All of the tools described are available from the **Tools** menu. However, navigational links to some tool also appear in the Nav side bar and in the **Nav Container View**.
- [About the Tools menu](#)
The Tools menu in the menu bar has the following options:
- [About the Window menu](#)
The Window menu in the menu bar has the following options:
- [About the Px Editor menu](#)
The **Px Editor** menu appears in the menu bar when Px Editor is the active view. The Px Editor has the following context-sensitive options:
- [About the History Ext Manager menu](#)
The **History Ext Manager** menu appears in the menu bar when History Extension Manager is the active view. The History Ext Manager menu has the following options:
- [About the Help menu](#)
The Help menu in the menu bar has the following options:
- [Popup menus](#)
Workbench provides view-specific, or context-specific commands for editing components in many of the views. Following, is a list of the standard Workbench popup (right-click) menus.

This menu in the menu bar provides a set of file-related options.

Figure 1. File menu



Menu item	Description
Open	This menu has submenus: • Ord opens an ORD chooser window. You may directly type in or copy and paste an ORD. If you specify no view, the default view for the ORD opens.. • File opens the File Chooser from which you can locate a file. • Directory opens a Directory Chooser from which to locate a directory folder. • Query opens the Bql Query Builder. • Open Platform opens a Connect window for connecting to the platform (PC or controller). – Host - This is the address of the computer running the Platform that you wish to access. – Port - This is the port used. – Password - This is the password given to you by your system administrator. Remember these credentials - This will save this connection in your connection list. Once you successfully connect to the Platform, it will appear in the tree and the available tools will be displayed. • Station: You open a Station by selecting File > Open > Open Station from the main menu. This will cause the following window to be presented so that you can complete each field. – Address - This is the address of the computer running the Station that you wish to access. – User name - This is the user name given to you by your system administrator. – Password - This is the password given to you by your system administrator.

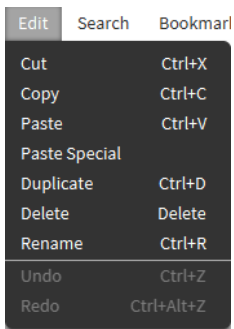
Menu item	Description
	– Remember these credentials - This will save this connection in your connection list. Once you successfully connect to the Station, it will appear in the tree and your home page will be displayed. If you will be away from the system, you should close the Station to prevent unauthorized access. Find Stations: You may find Stations by selecting File > Open > Find Stations from the main menu. This command finds all the stations running on the network. It will search for 5 seconds, then display a Table of all the stations found. You may display additional columns about each station using the options button (on the right of the header). Double-click a Station to open a Fox connection to it.
New Window	Creates a new window identical to the current one. This allows you to view multiple views concurrently.
New Tab	Creates a new tab identical to the current one. This allows you to view multiple views in the same Window. You can hold down the Ctrl key during a double-click to hyperlink into the new tab. You can also right-click to get a menu on tabs to close the tab, or close all other tabs.
Close Tab	Closes the existing tabs.
Next Tab	Selects the next tab. The shortcut is Ctrl + PageUp.
Previous Tab	Selects the Previous tab. The shortcut is Ctrl + PageUp.
Save	Saves the value of the Component.
Save All	Save all open views. This saves all Tabs when browsing with multiple Tabs.
Save Bog	Save the Component changes to the Bog File.
Bog File Protection	Protects the Bog File.
Export	Provides the capability to Export the current view. When it appears dimmed, Export is not available.
Back	Goes to the previous view. The shortcut is Alt + Left.
Forward	Goes to the next view. You must have used the Back command previously. The shortcut is Alt + Right.
Up Level	Goes to the next level up. The shortcut is Alt + Up.
Recent Ords	Sees recent Ords. The shortcut is Alt + Space.
Home	Goes to the home view. The shortcut is Alt + Home.
Refresh	Refreshes the current view. Holding down the Ctrl key clears various Workbench caches. As of Niagara 4.10u4 and later, holding down the Ctrl key is required to get a new browser instance.
Refresh Tabs	Refreshes all tabs that are open in the station.
Session Info	Gives the information about the running station.
Logoff	Log offs the Station at any time.

Menu item	Description
Close	You may close the current window at any time, if it is not the primary window, by any of the following methods: • Selecting File > Close from the main menu. • Clicking the window in the top left corner and selecting Close from the menu. • Clicking the close icon in the top right corner of the window.
Exit	You exit the system at any time by selecting File > Exit from the main menu. This differs from logoff in that it also causes the user interface to stop.

Parent topic: [Menus](#)

This menu supports standard edit options.

Figure 1. Edit menu



Menu item	Description
Cut	Copies the current selection to the clipboard, changes its color to gray, and deletes it when you Paste it somewhere else. If the selection is a string, Cut copies the selection to the clipboard and removes it from the string. To use it, select an item and • In general, right-click the component(s) and click Cut in the menu. (To Cut more than one component, hold the Ctrl key and click to select the items.) • In the Nav tree, right-click and click Cut in the menu. • In the Wire Sheet, select the component and click the Cut button (✂). • In the Wire Sheet, select the component and click Edit > Cut from the main menu.

Menu item	Description
	• In a window other than the n browser window, select the i to Cut and click the shortcut Ctrl + X (hold down Ctrl and enter X).
Copy	Copies the current contents of the clipboard to the destination component as a set of new dynamic properties. If the selection is a string Copy copies the current text selection and places it in the destination component. To Copy a Component • In general, right-click the component and click Copy in the menu. • In the Wire Sheet, right-click, select the component and click the Copy in the menu. • In the Wire Sheet, select the component and click the Copy button (). • In the Wire Sheet, select the component and click Edit > C from the main menu. • In a window other than the n browser window, select the i to Copy and click the shortcut key Ctrl + C (hold down Ctrl and enter C). • To copy a component, you m Copy it from the Palette or a existing Database. You can follow one the following steps: – In the Palette sidebar, expand a module (like control) and sub-direct (like Points), Right-click a Component (like BooleanWritable) and select Copy. – In the namespaces side (tree), expand System, Modules, a module (lik control), Files, module.palette, and su directory (like Points), Right-click on a Component (like BooleanWritable) and select Copy.

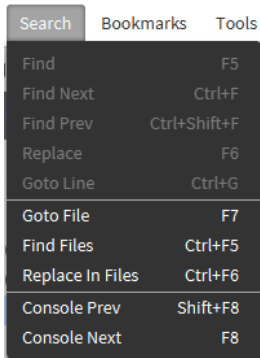
Menu item	Description
	– OR right-click on a component that you want to Copy from a Wire Sheet, Property Sheet or the tree and select Copy
Paste	Copies the contents of the clipboard to the destination component as a set of new dynamic properties. If the target is a string, Paste places a reference to the source that was cut or copied in the clipboard. It may be accessed by cutting or copying component(s): • In general, right-click the component(s) and click Paste in the menu. (To Cut more than one component, hold the Ctrl key and click to select the items.) • In the Nav tree, right-click and click Paste in the menu. • In the Wire Sheet, select the component and click the Paste button (). • In the Wire Sheet, select the component and click Edit > Paste from the main menu. • In a window other than the main browser window, select the item to Copy and click the shortcut key Ctrl + V (hold down Ctrl and enter V). You can use Drag to Cut and Paste in one operation. You can add a Component to a running Station or an offline File file. If you select Help > Guide On Target with a Component selected, you will get the Guide for that Component. If you select Help > Bajadoc On Target with a Component selected, you will get the bajadoc for that Component. The BooleanWritable bajadoc is a module://control/doc/javax/baja/control/BBooleanWritable.bajadoc do as follows: • In the Wire Sheet, right-click the background and select Paste

Menu item	Description
	to add a new component. The new Component is created and selected. In the Nav tree, right-click on the container and select Paste to add the new component inside the container.
Paste Special	Copies the contents of the clipboard to the destination component when it is a Special Transferable.
Duplicate	Copies the current selection and places a duplicate in the same Container. This function may be accessed from the menu under Edit (shortcut Ctrl + D) or from the Duplicate button () on the toolbar.
Delete	Removes the current selection from its parent container. It may be accessed by selecting an item and: - In general, right-click the component(s) and click Delete in the menu. (To Delete more than one component, hold the Ctrl key and click to select the items.) - In the Nav tree, right-click and click Delete in the menu. - In the Wire Sheet, select the component and click the Delete button (). - In the Wire Sheet, select the component and click Edit > Delete from the main menu.
Undo	Reverses the last action as if it had not been performed. It is only available after certain actions as follows: - Paste component - Cut component - Link - Delete Links
Redo	Restores an Action after Undo has removed it. It is only available after a successful Undo.

Parent topic: [Menus](#)

This menu provides search functionality.

Figure 1. Search menu



Menu item	Description
Find	Searches in the file for the selected string. You can Match Case or Match Word . The shortcut is F5.
Find Next	Finds the next occurrence of the selected string. The shortcut is Ctrl + F (hold down Ctrl and press F).
Find Prev	Finds the previous occurrence of the selected string. The shortcut is Ctrl + Shift + F (hold down Ctrl and Shift and press F).
Replace	Replaces the next occurrence in the file. The shortcut is F6.
Goto Line	Goes to a line number in the file. The shortcut is Ctrl + G (hold down Ctrl and press G).
Goto File	Goes to a file. The shortcut is F7
Find Files	Finds the all occurrences of a string in files. You can Match Case or Match Word . You can choose Text to Find and Files To Find . The shortcut is Ctrl + F (hold down Ctrl and press F5).
Replace Files	Replaces the all occurrences in the files. The shortcut is Ctrl + F6 (hold down Ctrl and press F6).
Console Prev	Goes to the previous console error. The shortcut is Shift + F8.
Console Next	Goes to the next console error. The shortcut is F8.

Parent topic: [Menus](#)

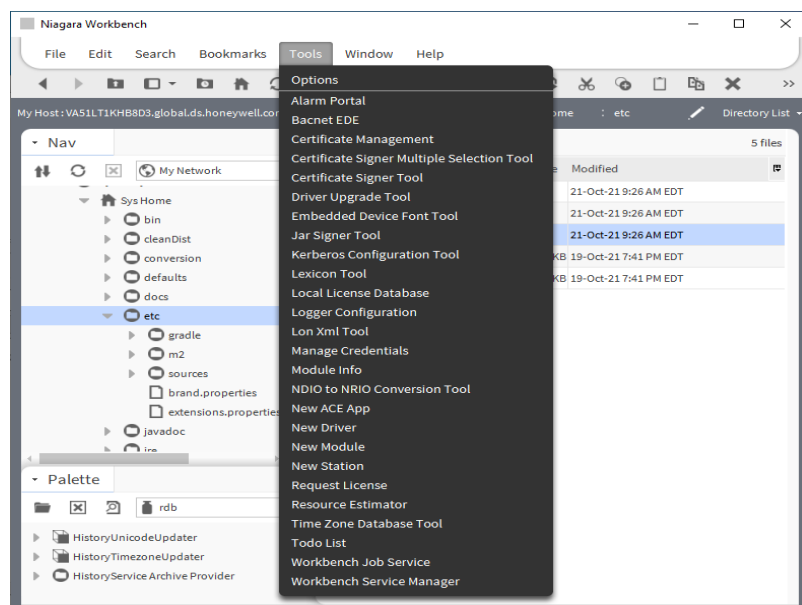
This menu customize the interface for your specific needs.

Menu item	Description
Add To Bookmarks	Adds a bookmark at the current view or Property Sheet .
Bookmark All Tabs	Adds a bookmark for all currently-open tabs.
Manage Bookmarks	Opens the Manage Bookmarks window where you can group bookmarks and perform other functions. The bookmarks you add expand the Bookmarks menu.

Parent topic: [Menus](#)

This section describes some of the standard tools available in Workbench. These tools provide views that are designed to facilitate specific tasks – from managing credentials to monitoring alarms. All of the tools described are available from the **Tools** menu. However, navigational links to some tool also appear in the Nav side bar and in the **Nav Container View**.

Figure 1. Tools menu



When you select one of these tools, Workbench adds it to the **My Tools** node in the Nav tree. Upon opening Workbench, no **My Tools** node appears in the Nav tree. It appears as soon as you click **Tools** and select a tool. It persists for the current session. When you close Workbench, **My Tools** disappears.

- [Options](#)
Use the Workbench **options** window to customize your Workbench interface and to set other preferences. Open this window by selecting **Tools > Options** from the menu bar.
- [Alarm portal](#)
The **Alarm Portal** tool allows you to view and acknowledge alarms from many different stations using a single viewer (portal).
- [Bacnet EDE](#)
This tool opens the **Bacnet EDE (Engineering Data Exchange) Property Sheet** and makes the Bacnet EDE service available under the **My Tools** menu.
- [Security management tools](#)
Three tools are available in Workbench to manage and sign PKI (Public Key Infrastructure certificates).
- [Driver Upgrade Tool](#)
Use this tool to upgrade the driver of remote station from the supervisor.
- [Embedded Device Font Tool](#)
Use this tool to create files containing precalculated font metrics for specific fonts. Embedded controllers must source font metrics from these files in order to correctly render text labels used in Hx/Px graphics installed on the controller.
- [Jar Signer Tool](#)
In Niagara, the **Jar Signer Tool** is useful for non-developers or anyone using an unsigned legacy module that is no longer supported by the vendor. The Jar Signer allows you to sign an unsigned *.jar file using a code signing certificate.

- [Kerberos Configuration Tool](#)
This tool provides an editor view, **Basic Krb5 Conf Editor**, with which to configure certain properties of an existing Kerberos configuration file (krb5.conf). Kerberos authentication requires the ability to acquire Kerberos tickets that can be forwarded. The editor allows you to enable/disable the Forwardable property.
- [Lexicon Tool](#)
This tool controls the text of the user interface. You can use it to translate the user interface.
- [Local License Database](#)
The Workbench **License Manager** view is available, when you select **Local License Database** from the main menu.
- [Logger Configuration tool](#)
The **Logger Configuration** tool allows you to manage log settings for the local Workbench. Using the **Logger Configuration** view you can add and remove log categories and change the applied severity level, which determines the amount of data that is displayed.
- [Lon Xml Tool](#)
The **Lon Xml Tool** is available from the **Tools** menu. The **Lon Xml Create** view helps you make your own Lon Xml (.lnml) file for a device, using the source .xif file and (if necessary) other resource files, as available from the device's manufacturer. The tool's window provides the necessary input fields.
- [Manage Credentials](#)
This tool, or **Credentials Manager** provides a window to access and open any previously remembered (cached) connections from your Workbench, including both platform and station connections.
- [Module Info View](#)
In Niagara, the **Module Info View** checks the signatures of modules installed on your local machine. The view lists all of the currently installed modules (in the !modules folder) each with a signature status icon. Like the **Software Manager** view, you can double-click (or right-click) each row to view more details.
- [NDIO to NRIO Conversion Tool](#)
In Niagara, you can use the **NDIO to NRIO Conversion Tool** to easily convert all NDIO objects to NRIO objects. This is necessary when replacing a legacy JACE model with the JACE-8000 or JACE-9000, which only support NRIO. While NRIO and NDIO objects have similar configuration properties, they are not interchangeable. So you cannot simply move objects from NDIO modules to NRIO modules, you must run the conversion tool.
- [New ACE App](#)
This tool identifies an application used for ACE (Analysis, Control and Evaluation). Creating such applications expands the framework's capabilities.
- [New Driver Wizard](#)
The **New Driver Module Wizard** is available when you select **Tools > New Driver** from the main menu. The first of four wizard window is shown below.
- [New Module Wizard](#)
This tool presents a series of windows that help you create and save a new module. It can be selected from the main Menu by selecting **Tools > New Module**.
- [New Station wizard](#)
This tool uses templates to define the starting contents of a new station. It can be selected from the main Menu by selecting **Tools > New Station**.
- [Request License](#)
This tool opens a Bind License form in your Workbench PC's default browser. By default, the only pre-filled field in this form is the host ID of your PC. You can request a license by submitting the form.
- [Resource Estimator](#)
The **Resource Estimator** tool is available from the **Tools** menu. Use it as a worksheet to help you estimate the total number of station resources that you will use in a projected station implementation. The resource estimator view is, as shown below.
- [Todo List](#)
The **Todo List** is available when you select it from the **Tools** menu. This tool is provided to

help you with organizing, prioritizing and tracking tasks in Workbench. The Todo list view is shown.

- **[Workbench Fox Analyzer](#)**

This tool is designed to help Analyze the Fox Traffic between Workbench and any connected Stations. The live Workbench view is useful for analyzing traffic in real-time. Using the tool, a number of filters can be applied to the component to pick up only desired Fox traffic. Once any settings have been modified, you must refresh any live Fox Analyzer views for the changes to take affect.

- **[Workbench Job Service](#)**

The **Workbench Job Service** view, shown below, is available when you select it from the **Tools** menu. This job service tool keeps track of all Workbench jobs—these are jobs that are not initiated under a specific station, but initiated by the Workbench environment.

- **[Workbench Library Service](#)**

In Niagara, the **Workbench Library Service** is available in the **Tools** menu. Enabled by default, the purpose of the **Library Service** is to store versioned station artifacts (i.e. files) in Workbench.

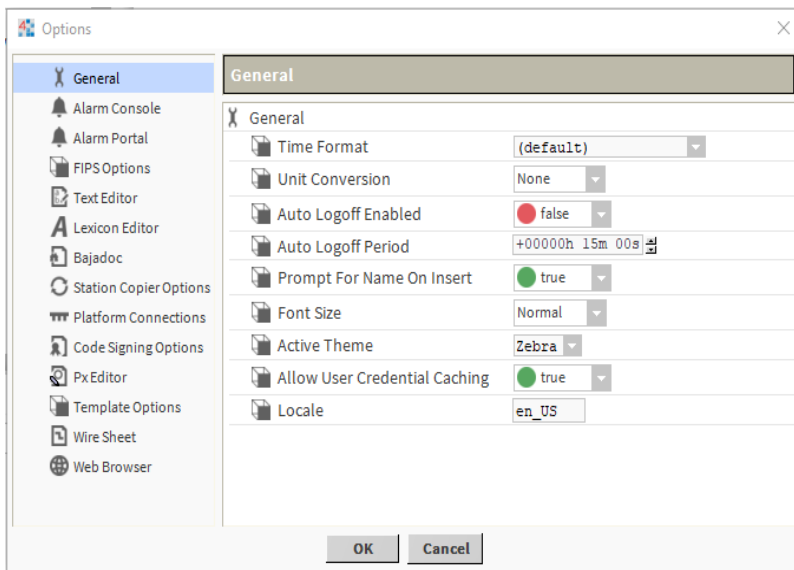
- **[Workbench Service Manager](#)**

The **Workbench Service Manager** view is available when you select it from the **Tools** menu. The view, shown below, is used to manage the life cycle and configuration of all services.

Parent topic: [Menus](#)

Use the Workbench **options** window to customize your Workbench interface and to set other preferences. Open this window by selecting **Tools > Options** from the menu bar.

Figure 1. Options window



The following options are available in the **Options** window:

- General
- Alarm Console
- Alarm Portal

- FIPS Options (latest version of Niagara)
- Text Editor
- Lexicon Editor
- Bajadoc
- Station Copier Options
- Platform Connection
- Code Signing Options
- Px Editor
- Template Options
- Wire Sheet
- Web Browser

General options include the active Workbench theme.

- **General options**
General properties include settings for a variety of Workbench display and behavior options.
- **Alarm Console**
Alarm Console options allow you to customize both the appearance and behavior of the alarm console.
- **Alarm Portal**
Alarm Portal options allow you to customize both the appearance and behavior of the Portal Alarm Console.
- **FIPS Options**
In Niagara, Workbench may be used to commission remote controllers to run in FIPS mode, whether or not Workbench itself is running in FIPS mode.
- **Text Editor**
These options offer a whole range of ways to customize the presentation and behavior characteristics of the text editor tool. Settings include text and symbol color coding options, as well as key bindings for shortcut keys. The text editor options properties are shown .
- **Lexicon Editor**
These options offer ways to customize the default settings of the Lexicon Editor. The Lexicon Editor options properties are shown and described in **Lexicon Editor** view. All of these properties may be overridden using the options that are available in the Lexicon Editor.
- **Bajadoc**
Baja reference documentation includes both Java API details as well as Baja slot documentation.
- **Station Copier Options**
This feature allows you to easily ignore critical data when copying stations to and from a platform. These properties allow you to configure station transfer options from the Workbench view. The properties include options to specify which directories and files to ignore when copying a station (by use of space delimited pattern filters). You can also use the property options to set default values for copying station alarm, history, job and critical data directories (hidden).

- **Platform Connections**
In Niagara, Workbench requires that the user remove the default platform user account and change the default system passphrase prior to completing a platform connection. These requirements are configurable via the **Platform Connections** options under **Tools > Options**.
- **Code Signing Options**
This view defines the code-signing certificate and timestamp URL the framework uses to sign all program objects (program modules, provisioning robots, objects edited by the Batch Editor, and robots created by the Program Service's Robot Editor).
- **Px Editor**
These options offer ways to customize the presentation and behavior characteristics of the **Px Editor** view.
- **Template Options**
You can use this option when template files are generated. It determines whether the template should include minor versions of Niagara modules after identifying the component and Px file dependencies.
- **Wire Sheet**
Wire Sheet options allow you to customize the appearance of the **Wire Sheet** view.
- **Web Browser**
This option allows you to customize the web browser by enabling or disabling the Web Development Tools.

Parent topic: [About the tools menu](#)

General properties include settings for a variety of Workbench display and behavior options.

Figure 1. General Workbench properties

General	
Time Format	(default)
Unit Conversion	None
Auto Logoff Enabled	☒ false
Auto Logoff Period	+000000h 15m 00s
Prompt For Name On Insert	☒ true
Font Size	Normal
Active Theme	Zebra
Allow User Credential Caching	☒ true
Locale	en_US

Note: The Time Format and Unit Conversion parameters affect values that are displayed when connected to a station using Workbench—regardless of the User preferences (set under User Manager). The User preferences that are set under the User Manager are in effect when connected to a station by a browser.

- **Time Format**
Choose from a format option to set the way that time values are displayed by default.
- **Unit Conversion**
Choosing the English or Metric option converts values that are displayed in Workbench to the chosen unit type. Selecting None leaves units in the state that is assigned at the point facet.

- Auto Logoff Enabled

When set to true this option automatically logs you off after a configurable amount of time where no user input occurs. This is the equivalent to right-clicking your connection and selecting “Disconnect”; you are telling the station, “I am finished.” Note that the Workbench auto logoff options apply to platform connections as well.

When set to false this option disables the auto logoff functionality for the Workbench only, meaning Workbench will not log you off.

Note: Separate auto logoff properties exist for the station which are configured via the UserService and User accounts. These two auto logoff methods, in Workbench and in the station, function independently of each other. For details, see the “baja-UserService” topic in the *Station Security Guide*.

- Auto Logoff Period

This specifies the period of time until the Workbench logs off a user due to inactivity. The default time period is 15 minutes.

- Prompt For Name On Insert

When set to true, Workbench displays a **Name** window, when a new item is added to the workspace.

- Font Size

Choose between Normal and Large font options for Workbench display.

- Active Theme

Choose between Zebra or Lucid “built-in” theme options for Workbench display.

- Allow User Credential Caching

If set to true (default), Workbench client access of a host (platform) or station allows a checkbox option to **Remember these credentials** in the **Authentication** window. If selected, the connection credentials are then cached and available in the Workbench **Credentials Manager (Tools > ManageCredentials)**.

This is a convenience mechanism. However, for security best practices it is recommended to globally disable user credential caching by setting this property to false. This way that option is unavailable in the **Authentication** window.

- Locale

Specifies the locale used by the Workbench VM, typically with a two-digit (ISO 639) code. Formerly, the Workbench locale had to be specified in the !lib/system properties file.

Parent topic: [Options](#)

Alarm Console options allow you to customize both the appearance and behavior of the alarm console.

Figure 1. Alarm Console

Alarm Console

- Notes Required On Ack: false
- Sounds Enabled: true
- Default Sound File: module://alarm/com/tridium/alarm/ui/sounds/smallBeep.wav
- Continuous Alarm: false
- Continuous Alarm Delay: +000000h 00m 10s
- Low Priority Color: cyan
- Mid Priority Color: yellow
- High Priority Color: red
- Time Zone Display: Console
- Alarm Class Mapping: Mapping >>
- Alarm Ack Responses: >>
- View Instructions: false

Alarm Console options include the following:

Property	Value	Description
Notes Required on Ack	true or false (defaults to false)	Causes the Notes window to open whenever you initiate an alarm acknowledgement from the alarm console, if this option is set to true.
Sounds Enabled	true or false (defaults to true)	Causes a sound to accompany an alarm. You can also set this value under the Alarms menu in the Workbench main menu when the Alarm Console view is active.
Default Sound File	file path	Sets the path to the default sound file.
Continuous Alarm	true or false (defaults to false)	Causes an alarm to repeat continually, until it is acknowledged or cleared, if this option is set to true. This option works together with the Continuous Alarm Delay property. You can also set this value under the Alarms menu in the Workbench main menu when the Alarm Console view is active.
Continuous Alarm Delay	hours, minutes, and seconds	When Continuous Alarm is enabled, this property causes a pause time between in the continuous alarm sound. The continuous alarm is interrupted for a time equal to the value of this property.
Low Priority Color	color chooser (defaults to cyan)	Selects the color for low-priority alarms. This color combines with the other priority colors to produce a range of colors that are assigned to the possible range of alarm priorities (1 - 255).
Mid Priority Color	color chooser (defaults to yellow)	Selects the color for mid priority alarms. This color combines with the other priority colors to produce a range of colors that are assigned to the possible range of alarm priorities (1 - 255).

Property	Value	Description
High Priority Color	color chooser (defaults to red)	Selects the color for high-priority alarms. This color combines with the other priority colors to produce a range of colors that are assigned to the possible range of alarm priorities (1 - 255). The framework assigns the highest priority (priority 1) the color of the High Priority Color setting. It assigns the Mid Priority Color and Low Priority Color likewise, to the mid and low priority alarms. The framework assigns colors to alarm priorities that fall between these priority levels on a color-scale along a path defined by the three assigned colors.
Time Zone Display	drop-down list (defaults to Console)	Controls the display of alarm record timestamp values in the time zone of the alarm console view (Console) or in the time zone of the alarm source (Source).
Alarm Class Mapping	path	Provides a way for you to create alarm classes and map specific alarms to classes.
Alarm Ack Responses	text	Creates one or more text entries that you can use to displays the Notes window when acknowledging an alarm. When the Notes Required on Ack property is set to true, the Notes window displays an additional option list containing any entries you create with this property. Use the button to open the associated Edit window and add, edit, or remove response options, as desired. When the Notes Required On Ack is set to false, these Alarm Ack responses are not visible.
View Instructions	true or false (defaults)	Causes the alarm Instructions pane to display across the bottom of the Alarm Console. true displays instructions in the pane for any single selected alarm that has associated instructions. false does not display the instructions in a pane across the bottom of the console.

Parent topic: [Options](#)

Alarm Portal options allow you to customize both the appearance and behavior of the Portal Alarm Console.

Figure 1. Alarm Portal

Alarm Portal	
Alarm Portal	
Tray Icon Enabled	☒ true
Alarm Popup Enabled	☒ true
Alarm Popup Always On Top	☒ true
Alarm Popup Uncloseable	☒ true
Kiosk Mode	☐ false
Reconnect Interval	+000000h 02m 00s
Default Time Range	Today

Alarm Portal options include the following:

Property	Value	Description
Tray Icon Enabled	true (default) or false	Controls the display of an alarm icon. true enables the display of an alarm icon in the system tray when the alarm portal is active. false disables the display of an alarm icon in the system tray.
Alarm Popup Enabled	true (default) or false	Controls the display of an alarm popup window. true displays an alarm popup window when the alarm portal is active. false does not display this window.
Alarm Popup Always On Top	true (default) or false	Controls the location of the alarm popup window. Causes the alarm popup window to stay on top of other windows when the alarm portal is active, if set to true.
Alarm Popup Uncloseable	true (defaults) or false	Controls if an operator can close the alarm popup window. true makes the alarm popup window uncloseable when the alarm portal is active. false allows the alarm console to be closed.
Kiosk Mode	true or false (default)	Configures the use of kiosk mode for the alarm console. true configures the alarm console to open in kiosk mode the next time it starts. false maintains the standard mode. For related information, refer to the <i>NiagaraAX Graphics Guide</i> ,
Reconnect Interval	hours, minutes, seconds	Defines when to attempt to reconnect a disconnected alarm console. If a console is disconnected, the framework attempts a reconnect within this interval.

Property	Value	Description
Default Time Range	time range	Selects the default time range that displays in the Portal Alarm Console pane of the Alarm Console view (Console).

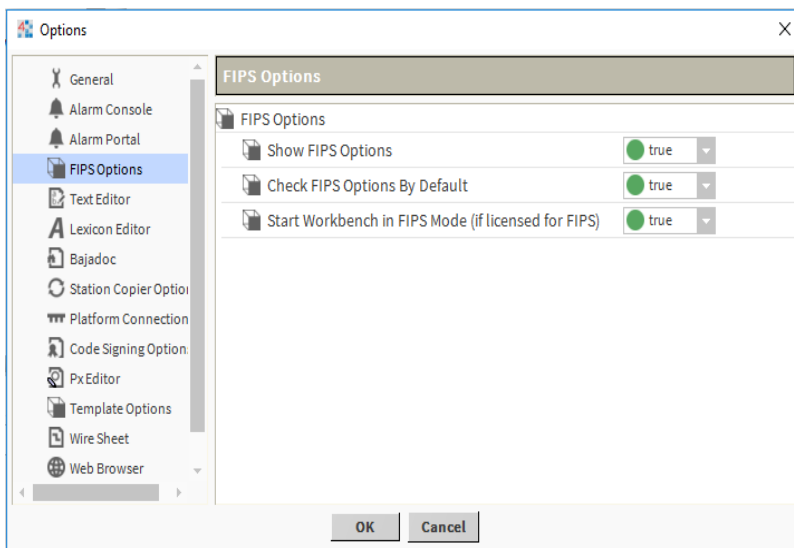
Parent topic: [Options](#)

In Niagara, Workbench may be used to commission remote controllers to run in FIPS mode, whether or not Workbench itself is running in FIPS mode.

In order to make FIPS options visible in various windows, go to **Tools > Options > FIPS Options**, and set **Show FIPS Options** to true.

If you would like the various FIPS options to be selected by default, set the **Check FIPS Options By Default** option to true.

Figure 1. FIPS Options

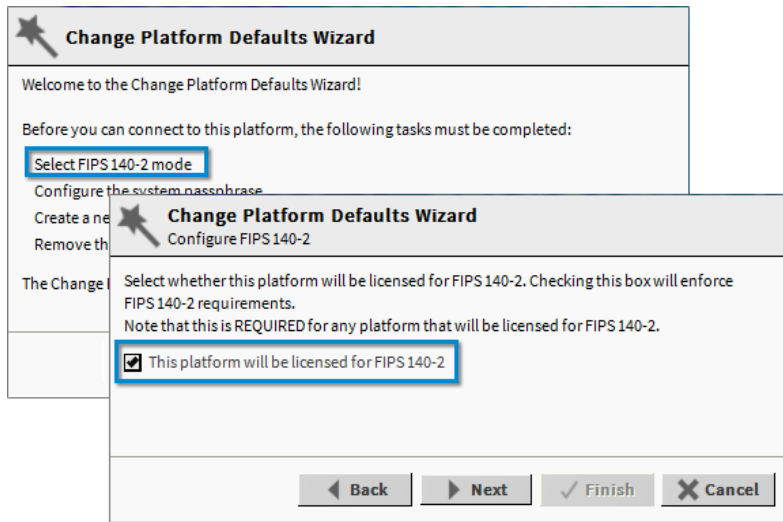


Setting **Show FIPS Options** to true causes certain FIPS options to be visible during the following tasks:

- Changing the default platform credentials via the **Change Platform Defaults Wizard**

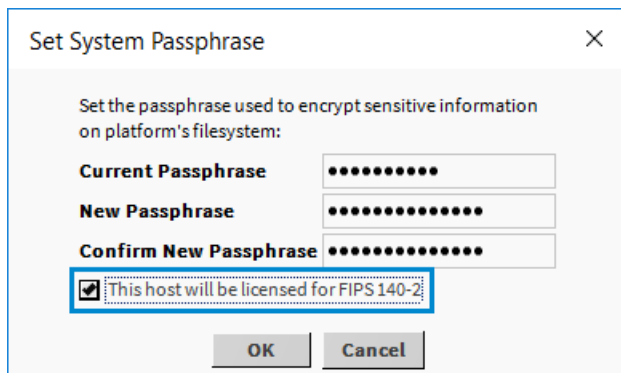
If **Show FIPS Options** is set to true, the **Change Platform Defaults Wizard** adds a step: “Select FIPS 140-2 mode”, as shown. This indicates that in a subsequent step the wizard displays a checkbox labeled, “This platform will be licensed for FIPS 140-2”. Clicking this checkbox enforces FIPS password strength requirements. If not checked, the platform does not consider a password FIPS-compliant, even if it technically meets the requirements. Also, if both **FIPS Options** are set to true, by default this checkbox is visible and selected. In that situation, the wizard enforces FIPS password strength requirements by default.

Figure 2. Change Platform Defaults Wizard step to select FIPS mode



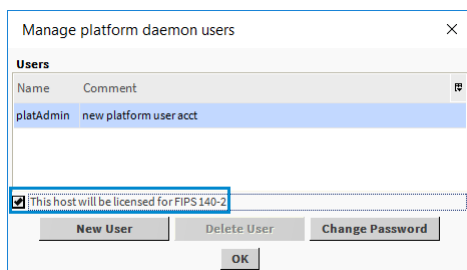
- Changing the system passphrase via the **System Passphrase** command in **Platform Administration**

Figure 3. FIPS Option in Set System Passphrase window



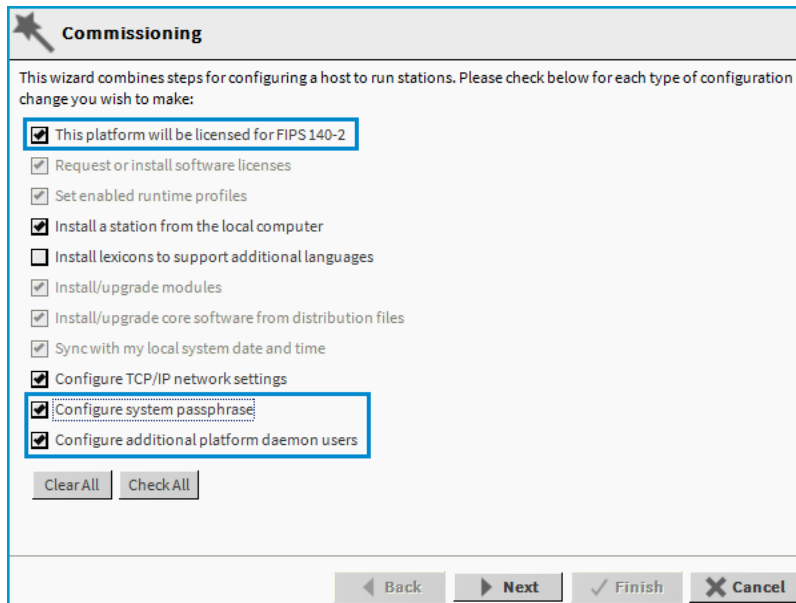
- Changing the platform user passwords via the **User Accounts** command in **Platform Administration**

Figure 4. FIPS Option in Manage platform daemon users



- Setting the system passphrase and platform user passwords during **Commissioning**

Figure 5. FIPS Option in Commissioning



Note: To install a FIPS license to a particular host, the **FIPS Options** described above must be set to true.

FIPS Compliant Passwords in Workbench

Workbench running in FIPS mode enforces strong passwords for operations, such as exporting certificates, setting passwords on certificates, and logging in to stations.

FIPS-compliant passwords must be at least 14 characters in length. This applies to most passwords, such as user passwords (platform and station), certificate passwords, the system passphrase, etc. Some passwords are excluded from this rule, such as passwords destined to be used with an external server, such as an email server.

Parent topic: [Options](#)

These options offer a whole range of ways to customize the presentation and behavior characteristics of the text editor tool. Settings include text and symbol color coding options, as well as key bindings for shortcut keys. The text editor options properties are shown .

Figure 1. Text Editor

Text Editor

Text Editor

Show Spaces false ▾

Show Tabs false ▾

Show Newlines false ▾

Tab To Space Conversion 2 [0 - max]

Show Margin 0 [0 - max]

Color Coding Color Coding

Key Bindings Key Bindings

Undo Navigation true ▾

Match Parens true ▾

Match Braces true ▾

Match Brackets true ▾

Word Right To End Of Word true ▾

Property	Value	Description
Show Spaces	true or false (default)	
Show Tabs	true or false (default)	
Show Newlines	true or false (default)	
Tab To Space Conversion	number (defaults to 2)	
Show Margin	number (defaults to 0)	
Color Coding	additional properties	
Key Bindings	additional properties	
Undo Navigation	true (default) or false	
Match Parens	true (default) or false	
Match Braces	true (default) or false	
Match Brackets	true (default) or false	
Word Right to End of Word	true (default) or false	

Parent topic: [Options](#)

These options offer ways to customize the default settings of the Lexicon Editor. The Lexicon Editor options properties are shown and described in **Lexicon Editor** view. All of these properties may be overridden using the options that are available in the Lexicon Editor.

Figure 1. Lexicon Editor

Lexicon Editor

Lexicon Editor

Hide Ref Values

☐ false

Hide Accelerators

☐ false

Hide Colors

☐ false

Hide Fonts

☐ false

Hide Types

☐ false

Show Value Filter

All

Property	Value	Description
Hide Ref Values	true or false (default)	
Hide Accelerators	true or false (default)	
Hide Colors	true or false (default)	
Hide Fonts	true or false (default)	
Hide Types	true or false (default)	
Show Value Filter	drop-down list	

Parent topic: [Options](#)

Baja reference documentation includes both Java API details as well as Baja slot documentation.

Figure 1. Bajadoc

Bajadoc

Bajadoc

Show Baja Only

☐ false

Flatten Inheritance

☒ true

Property	Value	Description
Show Baja Only	true or false (defaults to false)	Controls how much bajadoc to display. true displays only the reference documentation for slots (Properties, Actions, and Topics). false also displays documentation—methods and fields—on the Java constructors.
Flatten Inheritance	true or false (defaults to false)	Flattens the inheritance hierarchy into a single set of documentation. false displays only the Java members and Baja slots declared in the specified class. true displays all Java members and Baja slots inherited from super classes.

Parent topic: [Options](#)

This feature allows you to easily ignore critical data when copying stations to and from a platform. These properties allow you to configure station transfer options from the Workbench view. The properties include options to specify which directories and files to ignore when copying a station (by use of space delimited pattern filters). You can also use the property options to set default values for copying station alarm, history, job and critical data directories (hidden).

Figure 1. Station Copier Options

Station Copier Options	
Station Copier Options	
File Pattern Exclude List	*.lock console.* config.bog.b* config_ba:
Directory Pattern Exclude List	
Copy Remote Station Alarm Data	☐ false
Copy Remote Station History Data	☐ false

- File Pattern Exclude List

Use this option to set a pattern filter that excludes any specified file types from being copied with the station.

- Directory Pattern Exclude List

Use this option to set a pattern filter that excludes any specified directories from being copied with the station. For example, if you wanted to exclude copying all directories in the station that begin with the letters “lighting”, then you could type “lighting*” in this field.

- Copy options: (Remote Station Alarm Data, Remote Station History Data)

These properties allow you to copy (true) or not copy (false) certain station data (from the Remote Station to the Workbench).

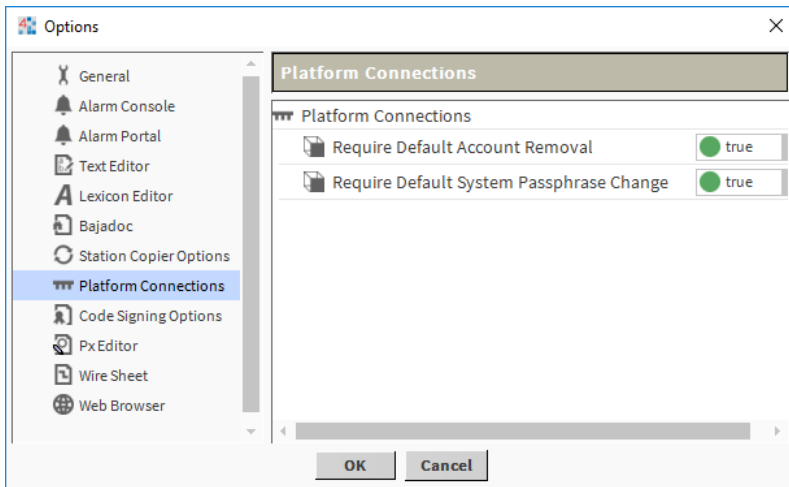
Note: Copying histories and alarm data using the Station Copier is supported only when copying *from* the remote station *to* Workbench, not the reverse. For more details, see “Station copy direction” in the *Niagara Platform Guide*.

Parent topic: [Options](#)

In Niagara, Workbench requires that the user remove the default platform user account and change the default system passphrase prior to completing a platform connection. These requirements are configurable via the **Platform Connections** options under **Tools > Options**.

Option properties

Figure 1. Platform Connections



This view allows you to configure whether or not the system prompts the user to remove the default platform user account and/or to change the default system passphrase on the first platform connection. These options are offered as a convenience. For example, if another workflow already prompts for these changes, setting one or both of these options to false can prevent redundant prompts.

These platform connection options are “true” by default, so that if Workbench detects either of the following conditions when making a platform connection it launches the **Change Platform Defaults Wizard** which steps you through the required changes:

- The system passphrase of the remote platform is the default value.
- The platform credentials of the remote platform are factory default values.

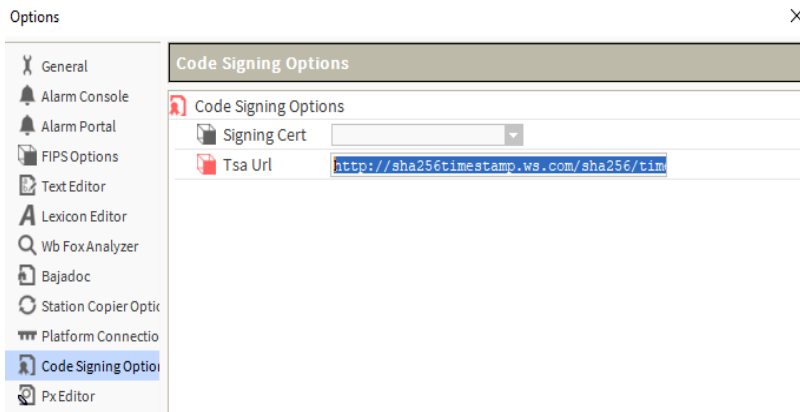
Name	Value	Description
Require Default Account Removal	true (default) or false	Controls what happens to the platform account. The admin account is required only the first time you connect to a platform. Deleting it is important to prevent someone with malicious intent from easily logging in to your system. true configures the system to require you to delete the default platform account after you connect to the platform for the first time. false allows the default admin account to remain, pending its deletion by another process.
Require Default System Passphrase Change	true (default) or false	Determines if a new system passphrase is required. The system passphrase protects access to the platform. A strong passphrase protects a platform from unauthorized access.

Name	Value	Description
		true configures the system to require you to create a new, strong passphrase after you connect to the platform for the first time. false preserves the default passphrase pending its change by another process.

Parent topic: [Options](#)

This view defines the code-signing certificate and timestamp URL the framework uses to sign all program objects (program modules, provisioning robots, objects edited by the Batch Editor, and robots created by the Program Service's Robot Editor).

Code Signing Options



Property	Value	Description
Signing Cert	text	Defines the name (Alias) of the code-signing certificate.
Tsa Url	URL	Selects a service to time stamp each program object signature. This stamp establishes trust even after a code-signing certificate expires. If program object signatures are not time-stamped, they cannot be validated past the expiration date of the code-signing certificate.

Parent topic: [Options](#)

These options offer ways to customize the presentation and behavior characteristics of the **Px Editor** view.

The **Px Editor** options properties are shown and described in the following list:
Figure 1. Px Editor options

Px Editor

Px Editor

Show Grid ☒ true

Grid Size [0 - max]

Grid Color

Use Snap ☒ true

Snap Size [0 - max]

Show Hatch ☒ true

Hatch Color

Preserve Identities ☒ false

Animate Bindings ☒ true

Neqlize Mode

Neqlize Excluded Relations

Use Service Excluded Relations ☒ true (Use excluded relations from the tag dictionary service and those defined here)

Neqlize Excluded Tags

Use Service Excluded Tags ☒ true (Use excluded tags from the tag dictionary service and those defined here)

Property	Value	Description
Show Grid	true or false (defaults to true)	Sets the default condition of the Px editor grid. Select true to make the grid visible by default or select false to make the grid hidden by default. Either setting may be changed at any time using the PxEditor menu.
Grid Size	number (defaults to 10)	Sets the size of the grid in the Px editor.
Grid Color	hex number	Sets the color of the grid in the Px editor. Click in the color field to display the Color Choose window. Use the Color Choose to set the color that you want to assign to the grid.
Use Snap	true or false (defaults to true)	Sets the default condition of the Snap feature in the Px editor. Select true to make objects snap to locations when they are at a distance equal to the Snap Size. Select false to disable the snap feature. Either setting may be changed at any time from the PxEditor menu.
Snap Size	number (defaults to 10)	Set an integer value in this field to define the interval between successive snaps.
Show Hatch	true or false (defaults to true)	Sets the default condition of the Px editor hatching that displays on objects on the Px editor canvas. Select true to make the hatching visible by default or select false to make the hatching hidden by default. Either setting may be changed at any time using the PxEditor menu.
Hatch Color	hex number	Sets the color of the hatching in the Px editor. Click in the color field to display the Color Choose window. Use the Color Chooser to set the color that you want to assign to the Px editor hatching.

Property	Value	Description
Preserve Identities	true or false (default)	Controls the encoding of names and handles on a Px page. true explicitly turns on support for encoding all names and handles on a Px page. false explicitly turns off support for encoding all names and handles on the page.
Animate Bindings	true (default) or false	Controls if the Px page displays live binding data. true displays live binding data for widgets in Px Edit mode. false provides no data animation in the Edit mode, although animation does occur, as expected, in View mode.
Neqlize Mode	drop-down list (defaults to Traverse if possible)	Selects the technique used to convert a slot path ORD to a tag-based NEQL query ORD. Traverse if possible attempts to find a traverse query but falls back to a select query. Traverse only attempts to find a traverse query only. Select only attempts to find a select query only.
Neqlize Excluded Relations	(empty by default)	Controls user-entered values for relation pattern filters used to exclude relations when converting slot path ORDs to traverse NEQL query ORDs.
Use Service Excluded Relations	true (default), false	Controls if user values for relation patterns are appended to TagDictionaryService values. true appends user values for relation pattern filters to the TagDictionaryService values. false uses the user values exclusively for relation pattern filters and ignores the TagDictionaryService .
Neqlize Excluded Tags	(empty by default)	Configures the user-entered values for tag pattern filters used to exclude tags when converting slot path ORDs to NEQL query ords.
Use Service Excluded Tags	true (default), false	Controls if user values for tag pattern filters are appended to TagDictionaryService values. true appends user values for tag pattern filters to the TagDictionaryService values. false uses the user values exclusively for tag pattern filters and ignores the TagDictionaryService .

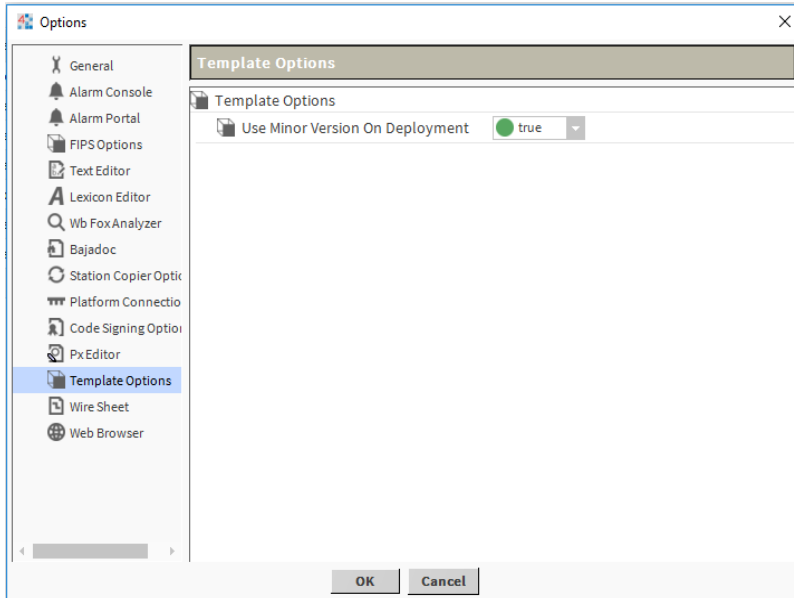
Parent topic: [Options](#)

You can use this option when template files are generated. It determines whether the template should include minor versions of Niagara modules after identifying the component and Px file dependencies.

If Use Minor Version On Deployment is set to true, a template created in Niagara have dependencies on minor version of Niagara modules.

If Use Minor Version On Deployment is set to false, the template states dependencies on the major version of Niagara 4 modules. This allows template that is created in later versions to be deployed to stations running previous minor versions.

Figure 1. Workbench Template Options



Parent topic: [Options](#)

Wire Sheet options allow you to customize the appearance of the **Wire Sheet** view.

Figure 1. Wire Sheet

Wire Sheet		
Wire Sheet		
Show Grid	☒ true	
Show Status Colors	☒ true	
Show Thumbnail	☒ true	
Show Links	☒ true	
Show Relations	☒ true	
Thumbnail X	-5	
Thumbnail Y	5	
Link Highlighting	☒ true	
Max Width	600	[120 - max]
Max Height	600	[120 - max]

Property	Value	Description
Show Grid	true or false (defaults to true)	Displays a grid background on the wire sheet view, if set to true.
Show Status Colors	true or false (defaults to true)	Displays a different color for each status when it appears in the wire sheet, if set to true.

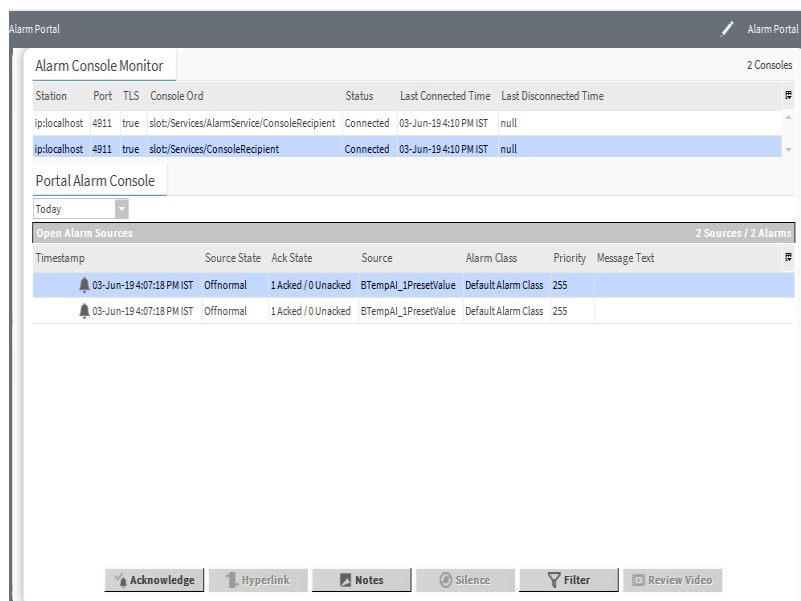
Property	Value	Description
Show Thumbnail	true or false (defaults to true)	Provides a small thumbnail view of the whole wire sheet for orientation and navigation purposes, if set to true .
Show Links	true or false (defaults to true)	Displays links.
Show Relations	true or false (defaults to true)	Displays relations.
Thumbnail X	number (defaults to -5)	Sets the X axis for the default position of the thumbnail view.
Thumbnail Y	number (defaults to 5)	Sets the Y axis for the default position of the thumbnail view.
Link Highlighting	true or false (defaults to true)	Turns on link highlighting, if set to true .
Max Width	number (defaults to 600)	Specifies a maximum size for the wire sheet width.
Max Height	number (defaults to 600)	Specifies a maximum size for the wire sheet height.

Parent topic: [Options](#)

This option allows you to customize the web browser by enabling or disabling the Web Development Tools.

Parent topic: [Options](#)

The **Alarm Portal** tool allows you to view and acknowledge alarms from many different stations using a single viewer (portal).



The alarm portal has two resizable panes:

- The top pane, **Alarm Console Monitor**, lists the stations displayed in the lower pane, **Open Alarm Sources**.

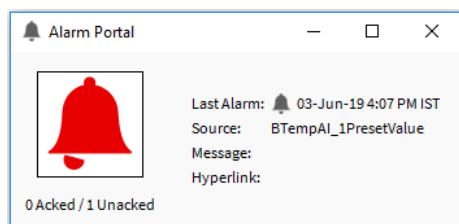
To add alarm consoles from different stations, right click in the **Alarm Console Monitor** (top pane) and choosing the **Add Alarm Console** menu selection to initiate the **Add Alarm Console** wizard.

To view individual alarm record to see all information on the alarm. Select an alarm and double-click it to see the **Viewing Alarm Record**. Available commands include:

To see alarm details, double-click on any alarm listed in the **Open Alarm Sources** pane.

The alarm portal tool, when enabled, also places the alarm icon in your system tray and an alarm window.

Figure 1. Alarm portal window



To set options for the alarm portal, click the **Tools > Options** menu.

Parent topic: [About the tools menu](#)

This tool opens the **Bacnet EDE** (Engineering Data Exchange) **Property Sheet** and makes the Bacnet EDE service available under the **My Tools** menu.

This tool configures Store Time for the **BacnetNetwork**'s Bacnet EDE devices.

Parent topic: [About the tools menu](#)

Three tools are available in Workbench to manage and sign PKI (Public Key Infrastructure certificates).

Use the **Certificate Management** view to secure communication within a **NiagaraNetwork**. Certificates secure TLS connections to the host.

Use the **Certificate Signer Tool** to sign intermediate digital certificates.

Use the **Certificate Signer Multiple Selection Tool** to perform bulk certificate signing.

For more details, see the "Secure communication" chapter in the *Niagara Station Security Guide*.

Parent topic: [About the tools menu](#)

Use this tool to upgrade the driver of remote station from the supervisor.

Figure 1. Driver Upgrade Tool

Upgrade
Enter connection information

Host: localhost

Username: Pacriml\H159983 Password: Port: 3011

Platform Connection: Pacriml\H159983

Station Connection: admin

Back Next Finish Cancel

Driver Upgrade Tool allows you to upgrade the driver in the remote station, it can be selected from the main menu by selecting **Tools > Driver Upgrade Tool**. This opens the **Upgrade** window to enter remote station credential and click **Next** to upgrade the driver of remote station.

Parent topic: [About the tools menu](#)

Use this tool to create files containing precalculated font metrics for specific fonts. Embedded controllers must source font metrics from these files in order to correctly render text labels used in Hx/Px graphics installed on the controller.

Figure 1. The Embedded Device Font Tool

Embedded Device Font Tool
1 of 2

Use this tool to create files containing precalculated font metrics for specific fonts. Embedded devices must source font metrics from these files, since they do not have the ability to calculate font metrics themselves. See this tool's page in the Help Documentation for more details.

Select an output directory for your .font files:
file:/D:/niagara/r43/niagara_user_home/shared/microFont

Select a system font you'd like to generate .font files for:
Arial

Provide a comma separated list of font sizes for the system font selected above:
8,9,10,11,12,14,16,18,20,22,24,26,28,36,48,72

Select font styles for the system font selected above:
☒ Plain ☐ Bold ☐ Italic ☐ Bold + Italic

Enter the number of characters (code points) you'd like from the system font selected above:
☒ Some Characters ☐ All Characters

The first 256 [256 - 65536] characters.

Back Next Finish Cancel

Note that by default each installation includes a font module (fonts.jar) containing font metrics for a few standard fonts. However, they may not be the same fonts as those used in Hx/Px graphics installed on an embedded controller.

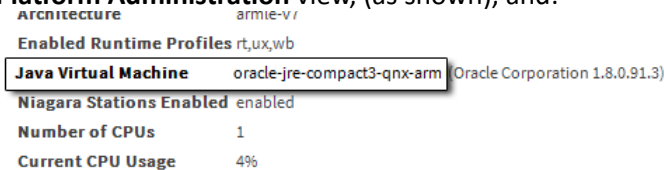
Also, note that Hx/Px graphics using multiple fonts may require that you use this tool to create a separate font metric file for each font. Multiple font files can then be bundled in a new module for ease of installation on embedded controllers.

Naturally, a best practice would be to limit font usage in Hx/Px graphics to a select few fonts that you create font files for or to limit usage only to the fonts contained in the default font module.

When should I use this tool?

Use this tool when all of the following are true:

- You are viewing an Hx/Px page that is being served up from a device using the Java compact3 profile. Confirm this by checking the Java Virtual Machine type listed in the **Platform Administration** view, (as shown), and:



- You notice one or more text labels in your Hx/Px page are cut off or overly wide, and
- The improperly sized label on your Hx/Px page is using a font that is not included in the default font module.

Default font module

By default the installation includes a fonts module that contains .font files with font metrics for the first 256 code points for the font families, point sizes, and styles listed in the following table.

Font families	Arial, Courier New, Source Sans Pro (Zebra Theme default font), Tahoma (Lucid Theme default font)
Font point sizes	8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28, 36, 48, 72
Font styles	Plain, Bold, Italic, Bold + Italic

For a given font family listed in the table (such as Arial) there is a separate font file for each combination of point size plus font style. For example, a separate font file is available in the default font module for each of the following combinations:

Arial 8pt Arial 8pt Bold Arial 8pt Italic Arial 8pt Bold Italic Arial 9pt Arial 9pt Bold Arial 9pt Italic
Arial 9pt Bold Italic ... etc.

Usage notes

Follow the on-screen instructions in the tool to generate the .font files you need. For best results, generate metrics using this tool on the same operating system that you will be using to view Hx/Px pages in a web browser.

You can install the .font files you have generated and save it to the fonts folder location on the embedded device, as shown in the example below:

My File System/User Home/shared/microFont
and

C:\Niagara\Niagara-4.XX.X.XX\jre\lib\fonts

Finally, restart your station so that your new .font files are properly loaded.

For developers

You can package your .font files into a separate module or include them in your theme's module.

1. Make sure your .font files are contained in a folder named microFont that is located at the root of your module.
2. You'll need to add the following `<def>` block to your new module's module-include.xml file:

```
<defs>    <def name="microFont" value="YourNewModuleNameHere"
  ↗ ↖</def>
```
3. Add the following code to your module's module-include.xml file:

```
<installation noRunningStation="true"↗
```

This invokes a station reboot upon module installation, ensuring that your new .font files are properly loaded.

Loading order

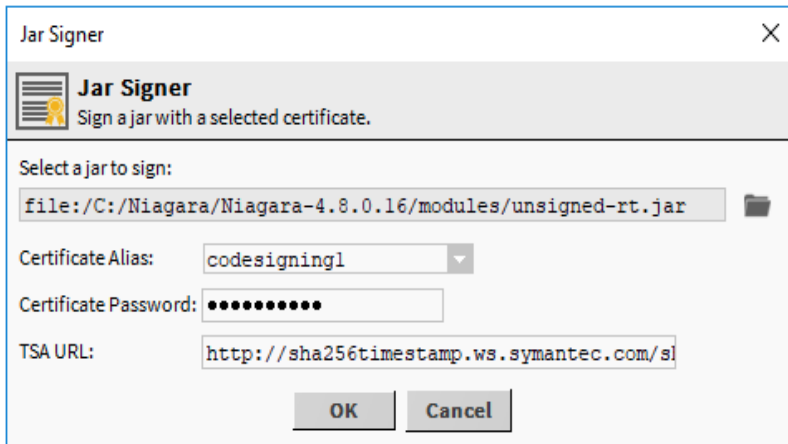
The framework will search for .font files in three locations in the following order:

1. From My File System/User Home/shared/microFont.
2. From modules that contain a microFont folder with .font files inside and a `<def>` in their module-include.xml.
3. From the default fonts module (module://fonts/microFont).

Parent topic: [About the tools menu](#)

In Niagara, the **Jar Signer Tool** is useful for non-developers or anyone using an unsigned legacy module that is no longer supported by the vendor. The Jar Signer allows you to sign an unsigned *.jar file using a code signing certificate.

Figure 1. Jar Signer Tool



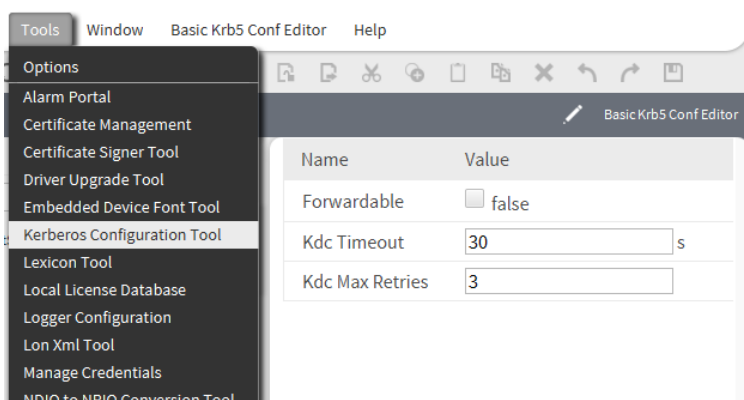
Using only signed modules provides some assurance that the code came from a trusted source and reduces the risk of installing malicious code. Another reason is to comply with module signing requirements which are gradually being phased-in over the next few releases. For more details, see *Niagara Third Party Module Signing*.

To use the **Jar Signer Tool**, you must first have a code signing certificate. If necessary, you can create one using the **Certificate Management** tool (for details, see *Niagara Station Security Guide*).

Parent topic: [About the tools menu](#)

This tool provides an editor view, **Basic Krb5 Conf Editor**, with which to configure certain properties of an existing Kerberos configuration file (krb5.conf). Kerberos authentication requires the ability to acquire Kerberos tickets that can be forwarded. The editor allows you to enable/disable the Forwardable property.

Figure 1. Basic Krb5 Conf Editor view



Additionally, the **Kerberos Configuration Tool** includes the **Advanced Krb5 Conf Editor** located under the Views drop-down list. The view provides a simple text editor which you can use to manually edit an existing Kerberos configuration file (krb5.conf) or to create a new one.

For more details, see the *Niagara LDAP Guide*.

Parent topic: [About the tools menu](#)

This tool controls the text of the user interface. You can use it to translate the user interface.

For more detailed information, see the *Niagara Lexicon Guide*.

For details on installing (edited) file-based lexicons and (new/edited) module-based lexicons in remote platforms, refer to the *Niagara Platform Guide*.

- **[Lexicon Report view](#)**

The **Lexicon Report** view is available via the Workbench **Tools** menu, by selecting **Lexicon Tool**. On the left side of this view, the **Lexicon** pane shows a list of all module-based and file-based lexicon locales installed on your Workbench PC. As needed, you click one of these lexicons to select it. Selecting a lexicon shows various status parameters about each lexicon (broken down by module) in the table columns on the right side.

- **[Lexicon Editor view](#)**

The **Lexicon Editor** view (shown) lets you view and edit the contents of lexicon files installed on your Workbench PC. Typically, you access the editor from the **Lexicon Report** view with a lexicon selected on the left side, by double-clicking a module in the table on the right side. The lexicon editor displays a table listing the defined keys for that module, with columns for mapping to a localized value override for the default value.

- **[Lexicon Module Builder](#)**

The **Lexicon Module Builder** view (shown) available in the allows you to bundle multiple lexicon files into a module for ease of distribution. The lexicons available for use are those lexicon files (modulename.lexicon) located in the !lexicon folder. You can build new lexicon modules or replace existing ones.

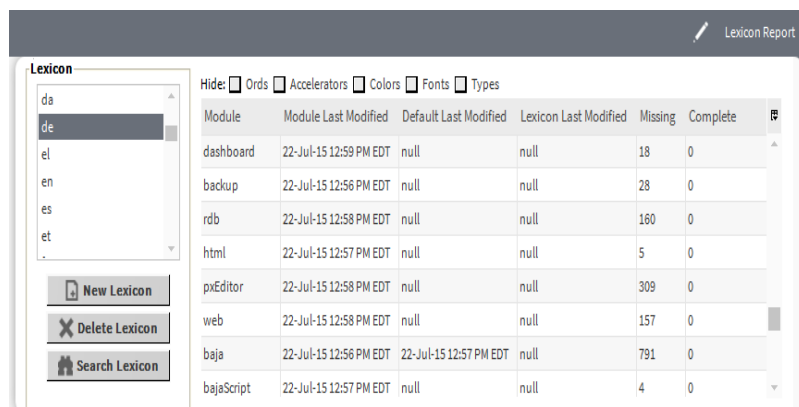
- **[Lexicon Module Migrator](#)**

The **Lexicon Module Migrator** view (shown) allows you to migrate AX-3.8 lexicon modules to the Niagara 4.0 installation.

Parent topic: [About the tools menu](#)

The **Lexicon Report** view is available via the Workbench **Tools** menu, by selecting **Lexicon Tool**. On the left side of this view, the **Lexicon** pane shows a list of all module-based and file-based lexicon locales installed on your Workbench PC. As needed, you click one of these lexicons to select it. Selecting a lexicon shows various status parameters about each lexicon (broken down by module) in the table columns on the right side.

Figure 1. Lexicon report view



Check boxes at the top of report view allow you to hide values that could affect missing status counts.

This view lets you add (new) lexicons, delete unwanted lexicons, or search all available lexicon property values that contain a given text. Additionally, you can double-click any module row in the table to launch the **Lexicon Editor** view, which shows the contents of that lexicon file.

For details, see the *Niagara Lexicon Guide*.

Parent topic: [Lexicon Tool](#)

The **Lexicon Editor** view (shown) lets you view and edit the contents of lexicon files installed on your Workbench PC. Typically, you access the editor from the **Lexicon Report** view with a lexicon selected on the left side, by double-clicking a module in the table on the right side. The lexicon editor displays a table listing the defined keys for that module, with columns for mapping to a localized value override for the default value.

Figure 1. Lexicon editor view

At the top of this view, you can search for specified text on either the lexicon property Key, the property Default, or the custom Value. Clicking any row in the table populates the **Key** field. At the bottom of the view, clicking the **Add New Key** button enables the Key field, as well. The **Value** text field allows you to modify the value of the selected key (or add a value for a new key). The **Value** property has associated **Color Chooser** and **Ord Chooser** buttons. These options allow you to browse for a specific color or ORD element. Update the table with the **Value** field entry by clicking the **Update Value** button. Write changes to the lexicon file by you clicking the **Save** button in the toolbar.

For detailed information on using the lexicon editor view, see the *Niagara Lexicon Guide*.

Parent topic: [Lexicon Tool](#)

The **Lexicon Module Builder** view (shown) available in the allows you to bundle multiple lexicon files into a module for ease of distribution. The lexicons available for use are those lexicon files (modulename.lexicon) located in the !lexicon folder. You can build new lexicon modules or replace existing ones.

Figure 1. Lexicon module builder view

Lexicon Module Builder

Module Name: **Browse...**

Preferred Symbol:

Module Description:

Vendor Name:

Version:

Brand Pattern:

☐ Delete lexicon files after build

Source	Lexicon	File	Path
☒	file	en	control.lexicon

Build Module

In this view there are text fields for defining module information, a **Browse** button that allows you to locate existing lexicon modules, a selection table of available lexicon files that can be selected for inclusion, a check box to delete source files after the build is completed, and a **Build Module** button to initiate the build.

On completion of the build, you will see a confirmation message indicating that the module was constructed and placed in the !modules folder.

For detailed information on using the lexicon module builder view, see the *Niagara Lexicon Guide*.

Parent topic: [Lexicon Tool](#)

The **Lexicon Module Migrator** view (shown) allows you to migrate AX-3.8 lexicon modules to the Niagara 4.0 installation.

Figure 1. Lexicon Module Migrator view and migrated lexicon modules in user home folder

Lexicon Module Migrator

Destination Folder: C:\Users\XYZ123\Niagara4.0\<brandId> **Change Destination...**

Select Lexicon Modules...

AX Lexicon Module	Module Path
testLexicon.jar	C:\niagara\Niagara-3.8.34\miscLexModules\testLexicon.jar
deJaLexicon.jar	C:\niagara\Niagara-3.8.34\miscLexModules\deJaLexicon.jar

Migrate Modules

You can select modules to migrate from anywhere in the PC file system. By default, the destination for migrated lexicon files is the Workbench user home. However, you can change the destination by selecting an alternate location.

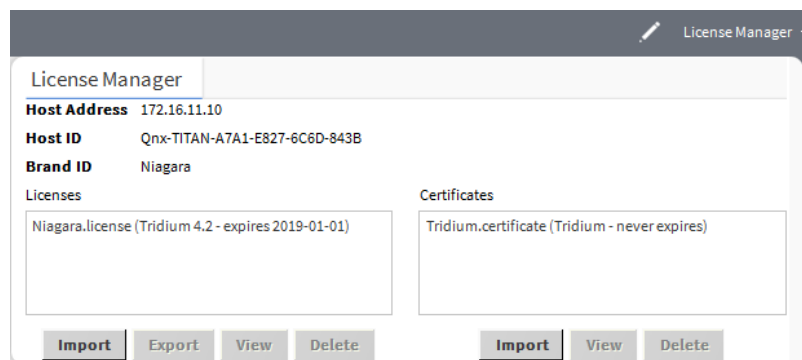
For detailed information on using the Lexicon Module Migrator, see the *Niagara Lexicon Guide*.

Parent topic: [Lexicon Tool](#)

The Workbench **License Manager** view is available, when you select **Local License Database** from the main menu.

The Workbench **License Manager** view lets you browse and manage the contents of your local license database.

Figure 1. License Manager



This view provides a two-pane window into all the license files and parent host ID folders, where:

- Left pane provides tree navigation, where you can expand folders and click (to select) license files.
- Right pane shows the text contents of any selected license file.

Buttons at the bottom of this view provide a way to manage the contents of your local license database, and are described as follows:

- **Import** adds license file(s) from a local license file or license archive (.lar) file.
- **Export** saves all licenses (or any selected licenses) locally, as a license archive file.
- **Delete** removes licenses from your local license database.
- **Sync Online**, typically available if you have Internet connectivity, updates all licenses (or any selected licenses) in your local license database with the most current versions, via the online licensing server.

Note: For details, see the *Niagara Platform Guide*.

Parent topic: [About the tools menu](#)

The **Logger Configuration** tool allows you to manage log settings for the local Workbench. Using the **Logger Configuration** view you can add and remove log categories and change the applied severity level, which determines the amount of data that is displayed.

Logging is an effective tool for troubleshooting and debugging station communications problems. Each driver and device installed on the connected station has a log category based on its type.

In Niagara 4 logging is handled by the Java.util.logging (JUL) utility which supports multiple concurrent log handlers and provides hierarchical logging support. Also in systems without the Workbench connection, this capability allows you to enable/disable loggers using a web browser.

Logging configuration settings for local or Workbench logs are stored in the !logging/logging.properties file which exists in two locations, your Workbench User Home and the daemon User Home, and are maintained by the framework.

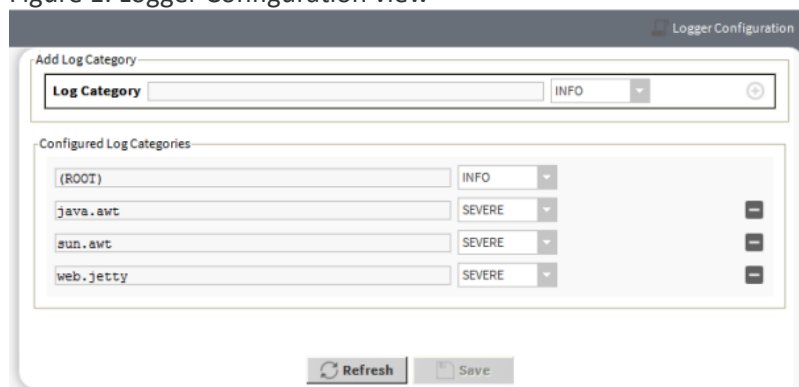
The Logger Configuration tool has seven severity levels which differs from the number of log levels used in NiagaraAX. Log levels used are mapped as shown here:

Table 1. NiagaraAX vs. Niagara 4 Severity Level Mapping

NiagaraAX Log Levels	Niagara 4 Log Levels
	Off
Error	Severe
Warning	Warning
Message	Info
	Config
Trace	Fine
	Finer
	Finest
	All

The **Logger Configuration** view is the main view for the **Logger Configuration** tool, as well as for the station **DebugService**.

Figure 1. Logger Configuration view



Logs for the connected station are visible in the platform **Application Director** view. While logs for Workbench are visible in the Niagara console started with Workbench.

The logs are persisted each time they are changed, since the changes are saved to the logging.properties file.

The ROOT log category is essentially a default log level. If any log is set to severity log level default, it uses the same configured severity level selected for the Root of that particular log. For example, the following diagram setting alarm to the default log level means that the alarm log level has been set to warning (the configured level of default row).

Figure 2. logSetup

Index | logSetup | DEFAULT-alarm Sample [1]

The Log alarm has been set to default Log level

Log	Level	Log Configuration									
		OFF	SEVERE	WARNING	INFO	CONFIG	FINE	FINER	FINEST	ALL	DEFAULT
DEFAULT	WARNING	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐
alarm	DEFAULT	☐	☐	☐	☐	☐	☐	☐	☐	☐	Not Applicable
alarm.database	INFO	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐
alarm.dataRecovery	INFO	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐
backup	INFO	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐
bacnet.link.ethernet	INFO	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐
baja	FINE	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
box	FINE	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
box.ord	FINE	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
box.reg	FINE	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐

Both Local (Workbench) Spy and Remote (Station) Spy can be accessed via Workbench by right-clicking the station in the Nav tree. Only Remote (Station) Spy can be accessed via a web browser using the url :[http\(s\)://<ip address>:<port number>/ord?spy](http(s)://<ip address>:<port number>/ord?spy):

Parent topic: [About the tools menu](#)

The **Lon Xml Tool** is available from the **Tools** menu. The **Lon Xml Create** view helps you make your own Lon Xml (.Inml) file for a device, using the source .xif file and (if necessary) other resource files, as available from the device's manufacturer. The tool's window provides the necessary input fields.

Figure 1. Lon Xml Create tool view

Lon Xml Create

Working Dir

Xif File

Resource Dir

☐ Zero Based Arrays

Refer to the *Lonworks Guide* for details about using this view and about the following:

- Need for custom Lon Xml files
- Lon Xml file overview
- Lon Xml creation

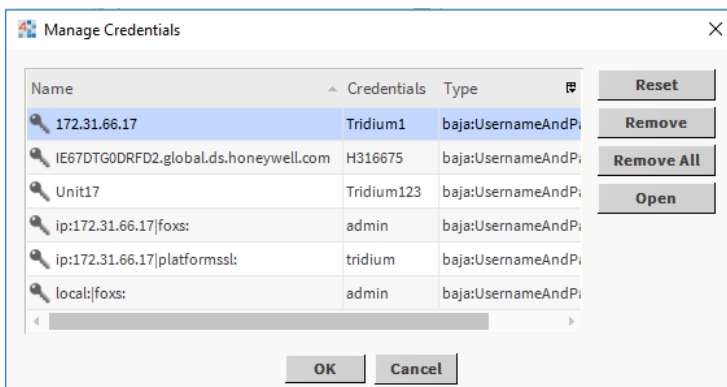
- Storing .Inml files
- Differential temperatures and Inml file edits

Parent topic: [About the tools menu](#)

This tool, or **Credentials Manager** provides a window to access and open any previously remembered (cached) connections from your Workbench, including both platform and station connections.

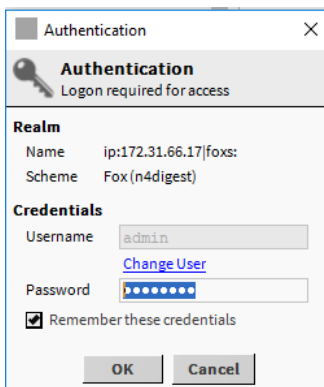
You can **Reset** or **Remove** selected Credentials or **Remove All** Credentials. You can also **Open** selected Credentials.

Figure 1. Manage Credentials window (credentials manager)



You can also remove or reset any cached credentials. Note that your credentials cache is populated whenever you have the login (Authentication) window option Remember these credentials (check box) enabled.

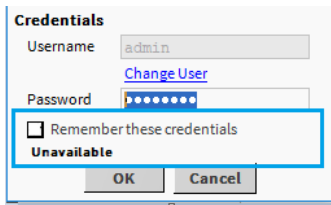
Figure 2. Authentication (Remember credentials)



The available caching of credentials is a convenience feature, such that you can simply open the platform or station later by entering only the IP address, or simply clicking on that host's dimmed platform or station in the Nav tree, or going to the credentials manager. The login authentication window has the cached credentials already entered, and you simply click **OK**.

If you want tighter security for any platform or station connection from your Workbench PC, you should clear this check box whenever opening (logging on) a platform or station. Furthermore, you should remove any related entries using the credentials manager. This way, that platform or station connection always requires full entry of both user name and password.

Note: You can globally disable user credentials caching in Workbench via **Tools > Options**, in the **General** menu. When Allow User Credentials Caching is set to false, the **Remember these credentials** checkbox remains unavailable (dimmed) in any **Authentication** window.



Parent topic: [About the tools menu](#)

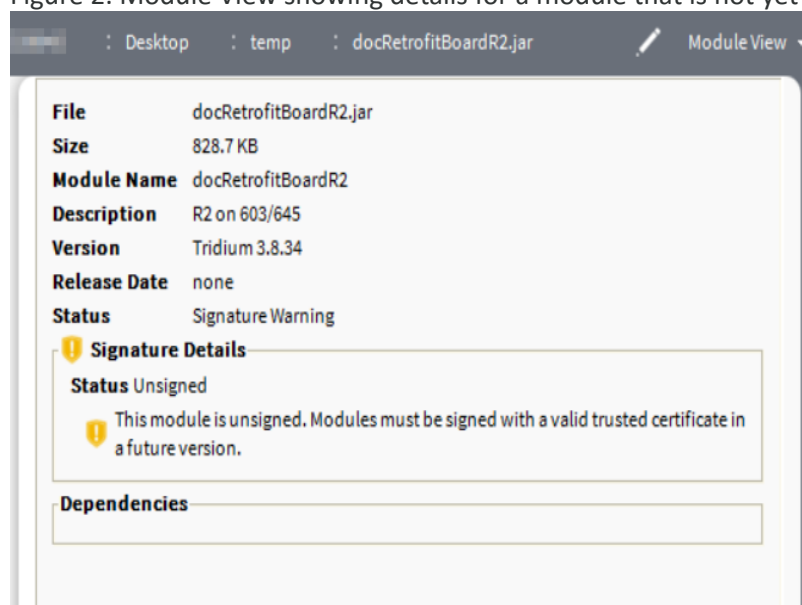
In Niagara, the **Module Info View** checks the signatures of modules installed on your local machine. The view lists all of the currently installed modules (in the !modules folder) each with a signature status icon. Like the **Software Manager** view, you can double-click (or right-click) each row to view more details.

Figure 1. Module Info View shows signature status for modules installed on local machine

Name	Version	Signature Status
aaphp-rt	Tridium 4.8.0.26	✓
aaphp-wb	Tridium 4.8.0.26	✓
aapup-rt	Tridium 4.8.0.26	✓
aapup-wb	Tridium 4.8.0.26	✓
ace-rt	Tridium 4.8.0.26	✓
ace-wb	Tridium 4.8.0.26	✓
alarm-rt	Tridium 4.8.0.26	✓
alarm-se	Tridium 4.8.0.26	✓
alarm-ux	Tridium 4.8.0.26	✓
alarm-wb	Tridium 4.8.0.26	✓
alarmOrion-rt	Tridium 4.8.0.26	✓

Note: To check the signature of a module in your file system that is not yet installed, you can navigate to the module's jar file in the Workbench Nav Tree, right-click the module and select **Views > Module View**.

Figure 2. Module View showing details for a module that is not yet installed



Parent topic: [About the tools menu](#)

In Niagara, you can use the **NDIO to NRIO Conversion Tool** to easily convert all NDIO objects to NRIO objects. This is necessary when replacing a legacy JACE model with the JACE-8000 or JACE-9000, which only support NRIO. While NRIO and NDIO objects have similar configuration properties, they are not interchangeable. So you cannot simply move objects from NDIO modules to NRIO modules, you must run the conversion tool.

The **NDIO to NRIO Conversion Tool** is available when you select **Tools > NDIO to NRIO Conversion Tool** from the menu bar.

When the conversion runs, the following actions occur:

- Each NDIO Network is converted to an NRIO Network. Any instance of NDIO in the name (case insensitive) is replaced with NRIO (preserving case) and the name is appended with "_converted"
- Each NDIO type object under an NDIO network is converted into an NRIO type object
- All dynamic properties and relevant static properties are copied to the new objects
- Any component under an NDIO network with a name that contains NDIO (case insensitive) are replaced by NRIO (preserving case)
- Any links to/from the NDIO objects are updated to reference the new NRIO objects.
- If running Niagara 4 or later, any relations to/from the NDIO objects are updated to reference the new NRIO objects.

Tool enhancements for the latest version of Niagara ensure the following:

- NDIO 16 modules will always be converted to NRIO16 modules.

- NDIO 34 modules will always be converted to the new NRIO34 modules.
- All slot path ords are preserved during the conversion. The names of any converted components have an option to replace instances of NDIO with NRIO in the display names of the NDIO networks and devices being converted.
- All handle ords are preserved during the conversion. The handle ords of the new NRIO components are set to be the same as those of the existing NDIO components .
- If not logged in as a Super User the conversion fails.
- If an updated nrio-rt module is not installed on the remote platform the conversion fails.

In Niagara, an NDIO device with more than 8 universal inputs, 4 digital outputs, or 4 analog outputs (NDIO-34) is split into multiple (1- to 3-) NRIO-16 devices, depending on the number of points. Points are split among each new device so no device exceeds the maximum amount of points, and addresses are updated so they are in the allowed range. The points in each new device retain their previous folder structure.

- Universal inputs (UI) 1-8, analog outputs (AO) 1-4, and digital outputs (DO) 1-4 are moved to device 1

Any miss-configured points with an address less than 1 are moved to device 1

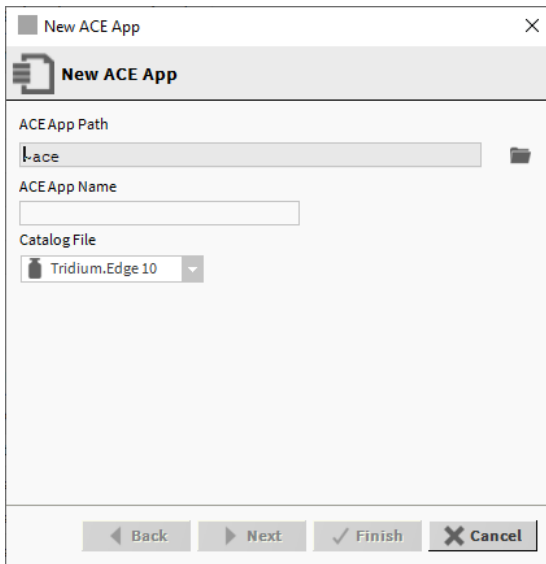
Any non-NDIO point objects are copied to device 1 only

- UI 9-16, AO 5-8, and DO 5-8 are moved to device 2
- DO 9-10 are moved to device 3

Parent topic: [About the tools menu](#)

This tool identifies an application used for ACE (Analysis, Control and Evaluation). Creating such applications expands the framework's capabilities.

Figure 1. New ACE App window



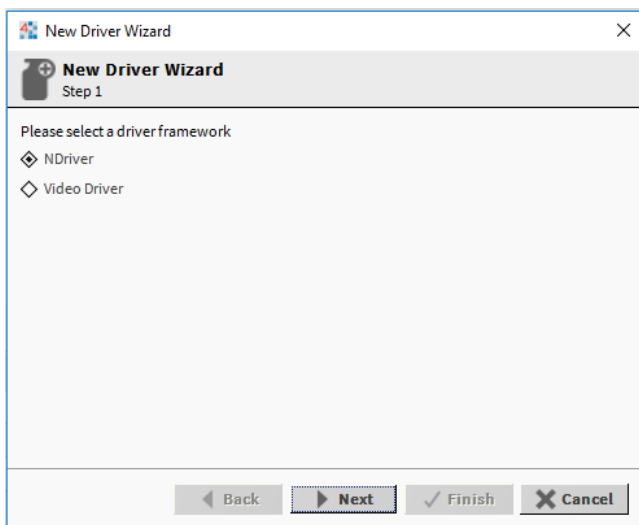
To open this window, click **Tools > New ACE App**.

Property	Value	Description
ACE App Path	file path	Locates the application in the station.
ACE App Name	text	Provides a name for the application.
Catalog File	drop-down list (defaults to Tridium.Edge 10)	Selects a source catalog.

Parent topic: [About the tools menu](#)

The **New Driver Module Wizard** is available when you select **Tools > New Driver** from the main menu. The first of four wizard window is shown below.

Figure 1. New Driver Wizard

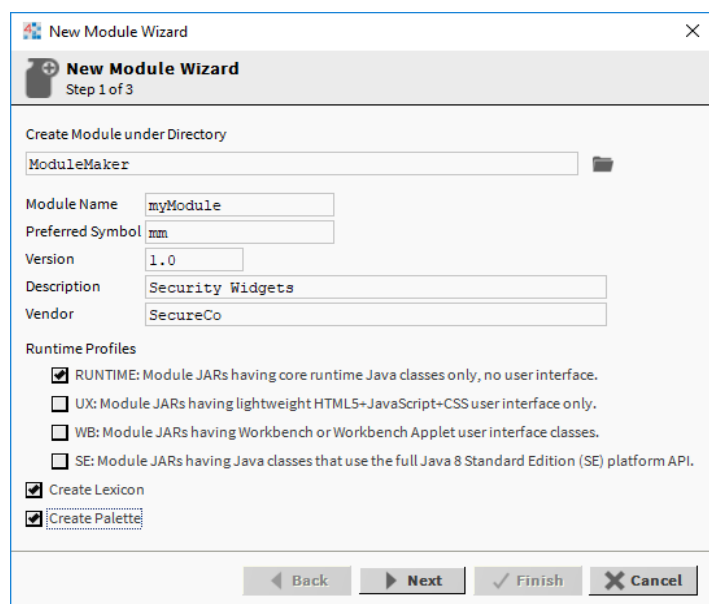


This wizard is provided as a convenient way to perform some of the steps necessary for creating a driver module. The four wizard windows allow driver developers to specify some basic driver module options and finish with the wizard creating a set of folders and files under a station directory that you assign during the wizard process. For details, see the *Niagara Developer Guide*.

Parent topic: [About the tools menu](#)

This tool presents a series of windows that help you create and save a new module. It can be selected from the main Menu by selecting **Tools > New Module**.

Figure 1. New Module Wizard creates folders under working directory



This wizard creates a set of folders and files under a station directory that you assign during the wizard process. After creating these files, you need to finish the process. For details, see the *Niagara Developer Guide*.

Parent topic: [About the tools menu](#)

This tool uses templates to define the starting contents of a new station. It can be selected from the main Menu by selecting **Tools > New Station**.

The wizard provides default station templates for creating a new controller (JACE) station or a new Supervisor station for either Linux or Windows operating systems. Additionally, the list of templates shown in the wizard window automatically includes any user-defined station templates that are available. The wizard configures the new station with services and components as determined by the selected template.

By design, the different default station templates (with the exception of names and the ports configured for Linux) create identical stations. However, the configuration of these templates may vary in future Niagara distributions and it is possible for other parties to create and distribute different station templates.

Note: In Niagara Workbench, the station templates in the **New Station Wizard** are updated so that the default Mobile Web Profile configuration of the users and user prototypes is the HTML5HxProfile. If you would like to change the Default Mobile Profile for when new users are created, go to the User Service **Default Prototype > User Prototypes** properties and set the Mobile Web Profile Type to the HTML5HxProfile.

Figure 1. New Station Wizard

Name	Vendor	Version	Description
NewControllerStation.ntpl	Tridium	1.5	
NewJACEProvisioningStation.ntpl	Tridium	1.3	
NewSupervisorStationLinux.ntpl	Tridium	1.7	
NewSupervisorStationWindows.ntpl	Tridium	1.7	

You must enter a **Station Name**. You can also choose whether this is a remote controller station or a Supervisor station. Click ➔ **Next** to proceed. Next you should enter an **Admin Password**. You can also change the **Fox Port** and **HTTP Port**. Press ✓ **Finish** to proceed. You are then presented a view of your newly created Station at local:|file:!stations/stationname/config.bog|bog:|slot:/.

When creating a new station, if you enter a station name identical to that of an existing station, the wizard alerts you that the station exists and prompts whether or not you wish to delete the existing station. The functionality gives you an opportunity to change the new station name, if you wish, or confirm that you intend to delete the existing station.

The wizard also performs entry validation on the password fields in the second dialog, alerting you to errors. By default, the system requires strong password. In order to comply, values entered in the Admin Password and Admin Password Confirm fields must be identical and meet the following password criteria:

- at least 10 characters
- at least 1 digit
- at least 1 lowercase character
- at least 1 uppercase character

Regarding the options for which action to take when you click **Finish**, the options presented are determined by whether and how you have made platform connections to localhost before. One or more of the following options are presented:

- **Open it in user home** - Selected by default, on completion the station is created in your User Home directory and a property sheet view of the station's config.bog displays.

At this point the new station exists only in your User Home directory and not in the User Home of the local platform daemon.

- **Copy it to the platform for "localhost" with Station Copier** - On completion, the station is created in your Workbench User Home directory. Then you are prompted to login to make a local platform connection. After you login, the **Station Copier** starts the transfer (copy). After the station is copied, the **Application Director** opens with the new station visible in the daemon's User Home.

At this point the new station exists in two locations: on your local host (in your Workbench User Home) and also in the platform daemon User Home.

Note: If you make any changes to the station now (while it is running on the platform daemon User Home), it is a good idea to copy the station back to your Workbench User Home so that you have a local copy of the updated station. This is useful if you plan to install it on any remote platform.

- **Close the wizard** - On completion, the station is created in your Workbench User Home directory, the wizard closes, and an alert window appears notifying you of successful station creation.

Figure 2. New station template containing configurable Foxs and HTTPS Port parameters

In cases where the selected station template contains exposed (configurable) parameters, those are included as editable fields on the second wizard window, such as the port parameters indicated in the above image. This gives you the opportunity to modify those values as needed to complete the configuration for your new station.

Note: When creating a station from the NewSupervisorStationLinux template the resulting station's Web Service is configured for HTTP on port 8080 and HTTPS on port 8443. Note, that you need to add a rule in your network firewall to allow data transmission on these ports.

The default station templates (controller, supervisor) have two port configuration parameters. However, anyone can create and distribute a custom station template that provides different parameters or none at all. Whether the wizard window displays alternative parameters, additional parameters, or no parameters is determined by the station template. If there are parameters they display in this space. When there are more parameters than the space allows, a scroll pane appears.

Parent topic: [About the tools menu](#)

This tool opens a Bind License form in your Workbench PC's default browser. By default, the only pre-filled field in this form is the host ID of your PC. You can request a license by submitting the form.

Figure 1. License request form in browser

The screenshot shows a web browser window with the address bar displaying 'axlicensing.tridium.com/license/request?params=YnJhbm...' and the Niagara Licensing logo. The form is titled 'Request/Bind License' and has two main sections. The 'License Details' section includes a 'Host Id' field with the value 'Win-5BE1-B094-FC24-3440' and an empty 'License Key' field. The 'Requester Details' section includes empty fields for 'Name', 'Company', and 'E-mail'. At the bottom of the form are 'Cancel' and 'Submit' buttons.

Typically, your Workbench PC is already licensed. Otherwise, you would not be able to successfully start Workbench, and then select **Request License** from the **Tools** menu.

However, you could use this as quick method to request a license for another PC on which you have installed the framework. In that case, you could substitute (type in) the host ID for the other PC in this form, along with other pertinent information.

Parent topic: [About the tools menu](#)

The **Resource Estimator** tool is available from the **Tools** menu. Use it as a worksheet to help you estimate the total number of station resources that you will use in a projected station implementation. The resource estimator view is, as shown below.

Figure 1. Resource Estimator view

Additional Multiplier	Medium (30%)				
Device Networks	0	x100,000 + Multiplier	0.000 KRU		
Devices	0	x5000 + Multiplier	0.000 KRU		
Proxy Points	0	x250 + Multiplier	0.000 KRU		
Program Components	0	x3000 + Multiplier	0.000 KRU		
Histories	Count * (250 + Size/10)				
Config.1	Count 0	Capacity	Record Count	Size 0	0.000 KRU
Config.2	Count 0	Capacity	Record Count	Size 0	0.000 KRU
Config.3	Count 0	Capacity	Record Count	Size 0	0.000 KRU
Config.4	Count 0	Capacity	Record Count	Size 0	0.000 KRU
Config.5	Count 0	Capacity	Record Count	Size 0	0.000 KRU
AlarmDb Capacity	0	x1	0.000 KRU		
Total	0.000 KRU				

 Calculate

- [Time Zone Database Tool](#)

This tool provides several ways to explore the local timezones.jar on the Workbench host. This jar file contains the historical time zone database.

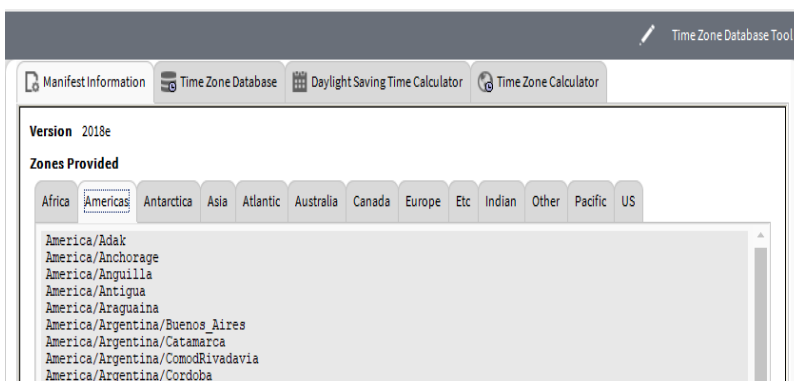
Parent topic: [About the tools menu](#)

This tool provides several ways to explore the local timezones.jar on the Workbench host. This jar file contains the historical time zone database.

If the Workbench host is running a station (for example, a Supervisor station), the station uses this time zone database. With this tool, you can see what time zones are available, see the past and current behaviors for any time zone, and, in some cases, the future behaviors as well.

Figure 1. Manifest Information tab

This tab displays the version of the time zone application and provides a list of time zones based upon the area of the world you select.

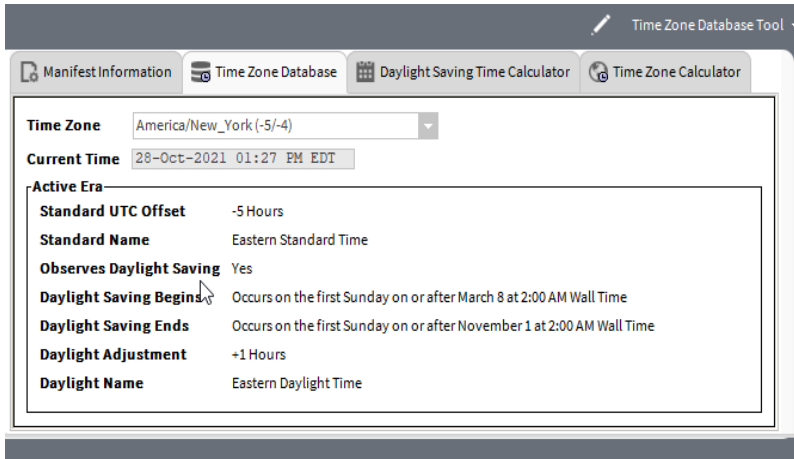


To open this tool, click **Tools** and select **Time Zone Database Tool**.

Time Zone Database

This tab summarizes the characteristics of the current time zone.

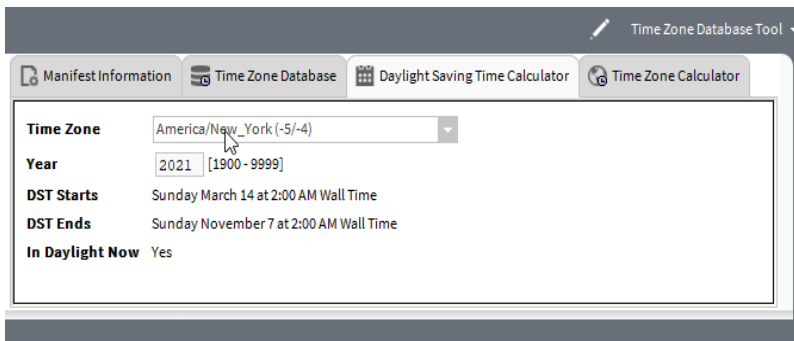
Figure 2. Time Zone Database



Daylight Saving Time Calculator

Figure 3. Daylight Savings Time Calculator tab

This tab shows when Daylight Savings Time (DST) occurs for any selected year.

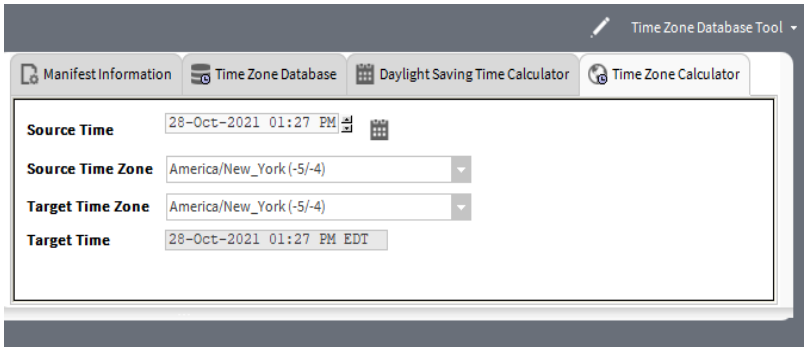


To use, click **Tools** and select **Time Zone Database Tool**, then select a time zone and type in a year in the Year property. The application updates when DST starts and ends.

Time Zone Calculator

This tab compares the time between any two time zones.

Figure 4. Time Zone Calculator tab

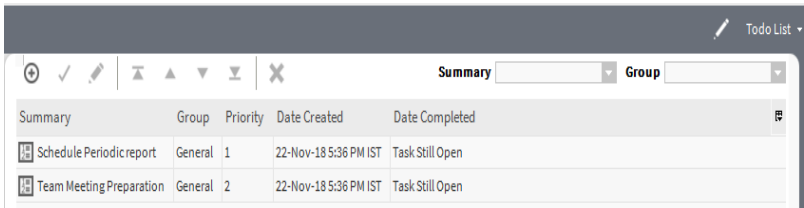


To use, click **Tools** and select **Time Zone Database Tool**, then select a source time zone and a target time zone and specify a source time (initial default is the current time). The application calculates the time in the target time zone.

Parent topic: [Resource Estimator](#)

The **Todo List** is available when you select it from the **Tools** menu. This tool is provided to help you with organizing, prioritizing and tracking tasks in Workbench. The Todo list view is shown.

Figure 1. Todo List view



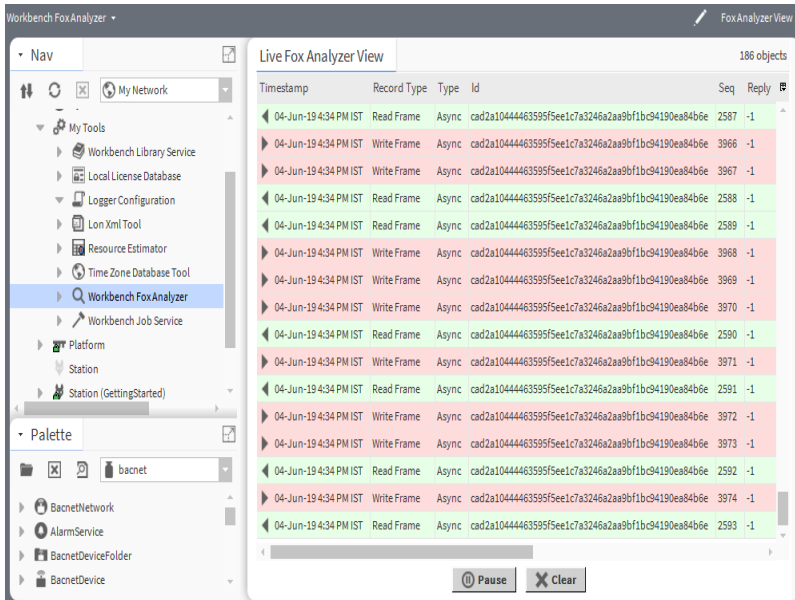
The Todo list is a tabular view with standard table controls and options. You can use this tabular list to create new lists or edit, group and rearrange existing lists and items in your lists. In addition, you can use the filter fields at the top of the display to filter what you see in the table, based on your summary description or group.

Parent topic: [About the tools menu](#)

This tool is designed to help Analyze the Fox Traffic between Workbench and any connected Stations. The live Workbench view is useful for analyzing traffic in real-time. Using the tool, a number of filters can be applied to the component to pick up only desired Fox traffic. Once any settings have been modified, you must refresh any live Fox Analyzer views for the changes to take affect.

Note: This tool is designed to only listen to Fox traffic between Workbench and any connected Station. It is not designed to listen for traffic between two stations! To view the traffic between two Stations, see the [Fox Station Analyzer](#).

Figure 1. Fox Analyzer View

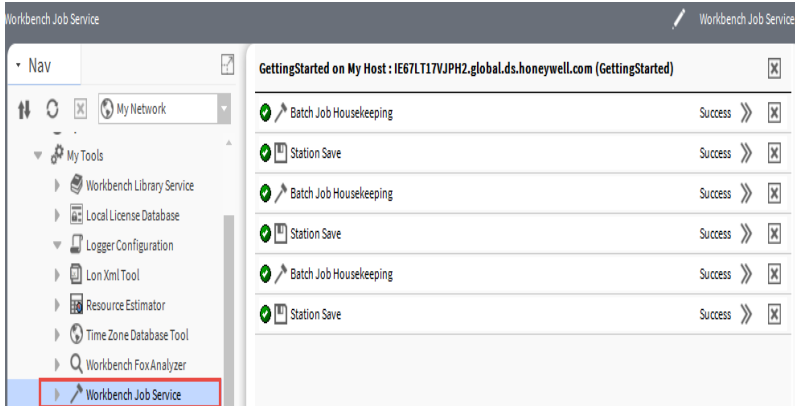


Option Name	Value	Description
View Log Limit	read-only	Displays Fox events in a color-coded table. The number indicates to the maximum number of entries displayed in the table. If the limit is exceeded, then the oldest entries are removed from the table.
Station Name Pattern	read-only	Filters for Fox Traffic between Workbench and a specific Station.
Id	read-only	Every Fox Session has a unique Id number. This Property can be used to filter for specific Fox Sessions, if the value is greater than -1.
Show Debug Frames	read-only	Fox Frames can be viewed in debug mode. This shows the frames in a readable user friendly format. By setting this Property to false, any future recorded Fox Frames will show their raw content.
Channel Pattern	read-only	Fox uses specially named Channels and Commands for different purposes. This Property provides a way to filter for specific Fox traffic.
Command Pattern	read-only	Like the Channel Pattern Property, this is used to filter for specific Fox traffic.

Parent topic: [About the tools menu](#)

The **Workbench Job Service** view, shown below, is available when you select it from the **Tools** menu. This job service tool keeps track of all Workbench jobs—these are jobs that are not initiated under a specific station, but initiated by the Workbench environment.

Figure 1. Workbench Job Service



Note: Workbench jobs are jobs that are initiated and run under Workbench - not under a station. Jobs that are run under a station are monitored and displayed in the **Station Job Service** (under the station **Services** node in the Nav tree sidebar).

Parent topic: [About the tools menu](#)

In Niagara, the **Workbench Library Service** is available in the **Tools** menu. Enabled by default, the purpose of the **Library Service** is to store versioned station artifacts (i.e. files) in Workbench.

The service allows you to view and manage objects (add and remove) in the Library, and deploy via the **Device Manager**.

The main view, the **Wb Library Service View** allows you to manually insert artifact input fields. You can invoke the view by clicking **Tools > Workbench Library Service**, or when viewing the **Workbench Service Manager**, double-click on **Workbench Library Service**.

Figure 1. Wb Library Service View shows

The screenshot shows a window titled "Wb Library ServiceView". It contains several input fields for defining an artifact:

- File Path:** C:/Users/E539945/Niagara4.4/tridium/templates/Basic-Pass.ntpl (with a "Browse..." button)
- Group:** com.tridium.template2
- Artifact:** Basic-Pass
- Type:** ntpl
- Version:** 1.0
- Metadata Name:** (empty)
- Metadata Value:** (empty)
- Search Pattern:** (empty)

Below the input fields, a text box displays the message: "Artifact inserted: com.tridium.template2:Basic-Pass:ntpl:1.0 - file:/C:/Users/E539945/Niagara4.4/tr".

At the bottom of the window, there are four buttons: "Insert", "Exists", "Resolve", and "Search".

Use **Browse** to locate and select a file to be inserted into the workbench library. Any file type can be inserted. In the view, the **File Path**, **Artifact** and **Type** values are filled in automatically. The **Group** field is a logical separator for artifacts in a Library, similar to a Java package name. The convention is to use a string with names separated by periods. Each name is considered a folder in the library for inserting and retrieving artifacts. For example, the Group `com.tridium.template` equates to a `/com/tridium/template` filepath, and artifacts will be inserted into the template folder in that path. The Group creates that path in the library and subsequent insertions will recognize that the path already exists.

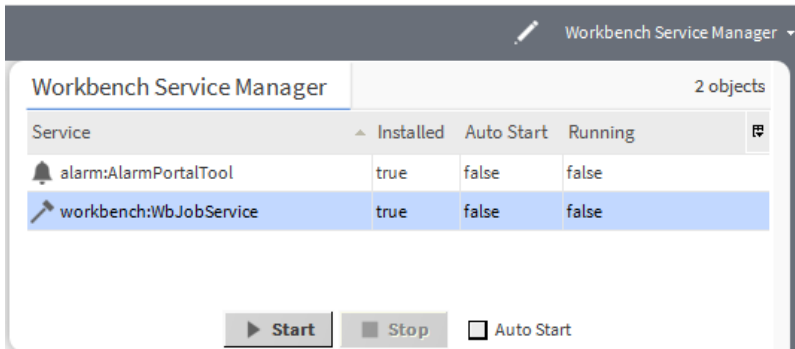
Entering a Group value activates the following buttons (at bottom):

- **Insert** — stores selected artifact(s) in the station library
- **Exists** — invokes a search for existing versioned artifacts in the station library
- **Resolve** — yields the ord for the selected artifact(s)

Parent topic: [About the tools menu](#)

The **Workbench Service Manager** view is available when you select it from the **Tools** menu. The view, shown below, is used to manage the life cycle and configuration of all services.

Figure 1. Workbench Service Manager view



The **Workbench Service Manager** is a tabular view with standard table controls and options. From this view you can Start, Stop, or configure any of the listed services to Auto-Start.

Parent topic: [About the tools menu](#)

The Tools menu in the menu bar has the following options:

Menu	Description
Viewing and Changing the Options	The Options allow you to customize the framework for the way you use it. It can be selected from the main Menu by selecting Tools > Options. It includes the following: • General • Lexicon • Text Editor • Wire sheet
ColorChooser	The ColorChooser allows you to choose Colors.
New Module Wizard	The New Module wizard allows you to build a new Module. It can be selected from the main Menu by selecting Tools > New Module .
New Station Wizard	The New Station Wizard allows you to build a new station Database. It can be selected from the main Menu by selecting Tools > New Station. You must enter a Station Name. You can also choose whether this is a remote controller station or a Supervisor station. Click ➡ Next to proceed. Next you should enter an Admin Password. You can also change the Fox Port and HTTP Port. Press ✓ Finish to proceed. You are then presented a view of your newly created Station at local: file:!stations/stationname/config.bog bog: slot:/.
Manage Credentials	Manage Credentials is available in the main Tools menu by selecting Manage Credentials . You can Reset or Remove selected Credentials or Remove All Credentials. You can also Open selected Credentials.
Request License	Request License is available in the main Tools menu by selecting Request License . You can request a license by submitting the form.

Parent topic: [Menus](#)

The Window menu in the menu bar has the following options:

Menu	Description
Side Bars	✓ Show Side Bar. You can choose whether to have Side Bars by selecting Window > Side Bars > Show Side Bar from the main menu.
PathBar Uses NavFile	You can toggle this option ON or OFF by selecting Window > Side Bars > PathBar Uses NavFile . When ON, the PathBar (located at the top of the Workbench main window) displays your current path, as defined by the NavFile (logical path). When OFF (not selected) the PathBar displays your absolute path regardless of whether it is mapped to the NavFile or not. You must refresh the view after changing this setting to see the PathBar change.
Active Plugin	The Active Plugin function gives focus to the current view. From the main menu you can select Window > Active Plugin or use the shortcut Ctrl + F4. It is very useful to use F3/Ctrl + F4 to toggle between the Console and the Text File Editor.
Hide Console	You can hide the console by selecting Window > Hide Console from the main menu or using the shortcut Ctrl + F2.
Console	The Console provides the capability to issue console commands directly. From the main menu you can select Window and Console (F3) or Hide Console (F4) to determine if the console is visible.

Parent topic: [Menus](#)

The **Px Editor** menu appears in the menu bar when Px Editor is the active view. The Px Editor has the following context-sensitive options:

Item	Description
Toggle View/Edit Mode	This command toggles the active view between Px Editor (for editing) and Px Viewer (view only). If there are unsaved changes in your Px file, you are prompted to save before switching from Px Editor to Px Viewer.
View Source Xml	Selecting this command displays the Px source file in a separate read-only window.
Go to Source Xml	Selecting this opens the Px source (xml) file directly in the text file editor. Files can be edited and saved using the editor.
Grid	This command toggles the grid display on and off.
Snap	This command toggles the snap-to-grid feature on and off.
Show Hatch	This command toggles the hatching pattern visibility on and off. When hatching is on, dim angular lines (hatching pattern) displays on objects to make them more visibly distinct.
Zoom In	The Px Editor display zooms-in on the canvas pane displaying less of the page at an enlarged size.
Zoom Out	The Px Editor display zooms-out on the canvas pane displaying more of the page at a reduced size.
Reset Zoom	Resets the canvas pane magnification to x1.0 (100%), displaying the Px page in actual size.
Set Target Media	This command is available from the PxEditor menu when you are editing a Px file directly in the Px Editor—not when you are editing the Px file as a view of a component. When selected, this command displays the Set Target Media dialog box to allow you to choose your expected media viewer: • HxPx Media • Mobile PxMedia (obsolete)

Item	Description
	• Report PxMedia • Workbench PxMedia

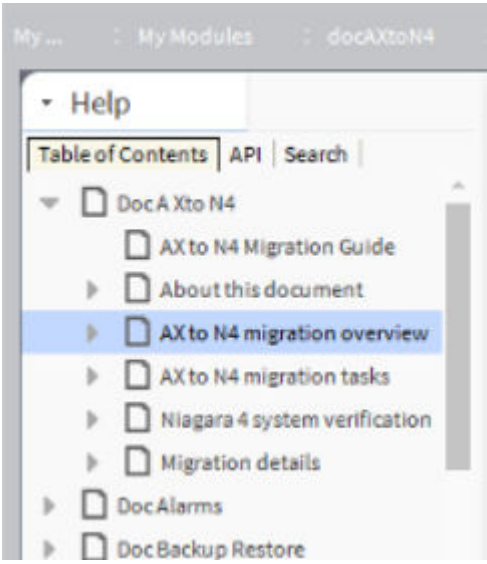
Parent topic: [Menus](#)

The **History Ext Manager** menu appears in the menu bar when History Extension Manager is the active view. The History Ext Manager menu has the following options:

Menu	Description
▶ Enable Collection	Select this menu item to enable (start the collection process) for the selected entries.
■ Disable Collection	Select this menu item to disable (stop the collection process) for the selected entries.
📄 Rename History	Select this menu item to rename the selected history. This menu item displays the Set History Name window.
📄 Edit System Tags	Select this menu item to open the Set System Tags For Selected History Extensions window. Use this window to edit system tags associated with a single history extension or perform batch edits when you have more than one history extension selected.

Parent topic: [Menus](#)

The Help menu in the menu bar has the following options:

Menu	Description
 Help Contents Index	The help contents provides a common point of access to all system documentation. It is accessed by selecting Help > Contents Index from the menu or pressing the ? Help button on the toolbar to see the Help Index.

 Help On View	This provides help for the current Plugin. It is accessed by selecting Help > On View from the menu with the Plugin in use.
 Help Guide On Target	This provides context sensitive help for Components. It is accessed by selecting Help > Guide On Target from the menu when the current view is a view of a Component. It is also available by Right-clicking a Component and choosing Views > Guide Help .
 Help Bajadoc On Target	This provides context sensitive Bajadoc help for Components. It is accessed by selecting Help > Bajadoc On Target from the menu when the current view is a view of a Component. It is also available by Right-clicking a Component and choosing Views > Bajadoc Help .
 Help Find Bajadoc	This is accessed by selecting Help > Find Bajadoc from the menu. It searches for the requested bajadoc.
 Help About	This is accessed by selecting Help > About from the menu. It provides the software release and license information.

Parent topic: [Menus](#)

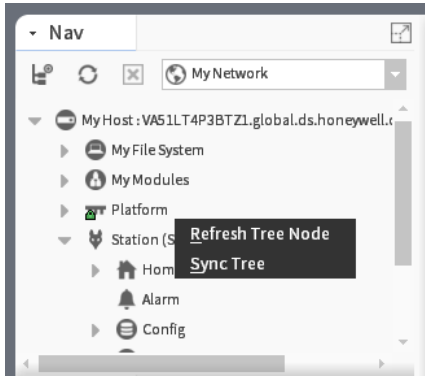
Workbench provides view-specific, or context-specific commands for editing components in many of the views. Following, is a list of the standard Workbench popup (right-click) menus.

- Nav side bar
- Wire sheet
- Property sheet
- Px Editor
- History extension manager
- Todo list
- Point Manager
- [About the Nav sidebar menu items](#)
The Nav sidebar provides a hierarchical view of the system that is analogous to an upside-down tree. The Nav sidebar menu displays command options when you right-click on a component item in the tree.
- [About the Wire Sheet menu items](#)
Most of the Wire Sheet menu commands are described in another topic in this document.
- [About the property sheet popup menu items](#)
Most of the **Property Sheet** menu commands are described below.
- [About the Px Editor popup menu items](#)
This popup menu appears when you right-click on an object in the **Px Editor** view. The menu commands are context-sensitive and are dimmed or available, depending on the type of object that you select.
- [About the history extension manager popup menu items](#)
The history extension manager popup menu has the following items.
- [About the Todo list menu items](#)
The Todo list menu has the following items.
- [About the point manager menu items](#)
The point manager popup menu has the following items.

Parent topic: [Menus](#)

The Nav sidebar provides a hierarchical view of the system that is analogous to an upside-down tree. The Nav sidebar menu displays command options when you right-click on a component item in the tree.

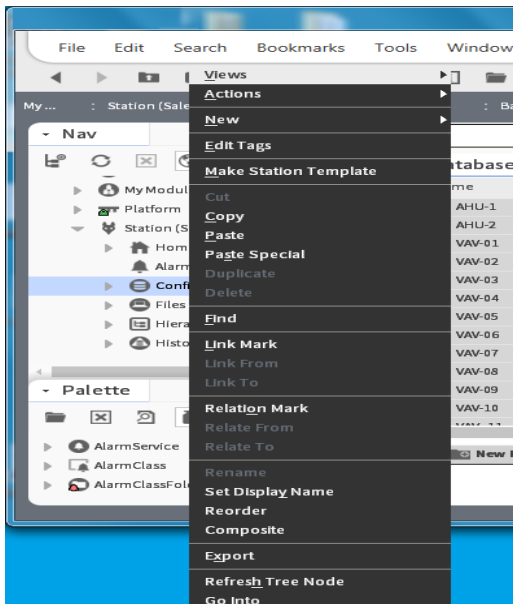
Figure 1. Nav sidebar popup menu



This popup menu has these options:

Item	Description
Refresh Tree Node	Refreshes the Nav tree.
Sync Tree	Synchronizes the Nav tree.

Figure 2. Nav sidebar popup menu



The popup menu has the following items:

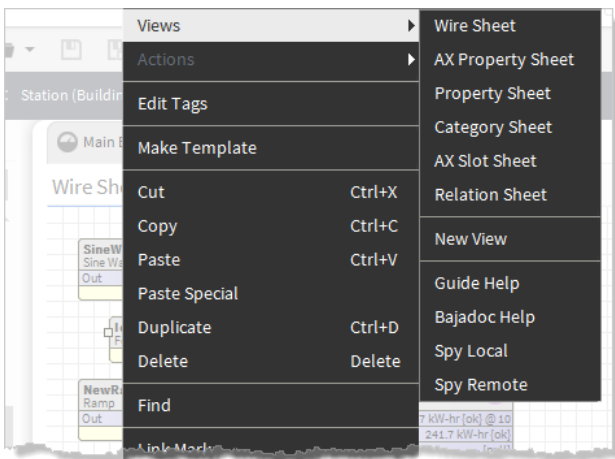
Item	Description
Views	Goes to any of the views of the selected component.
Actions	Perform any available actions on the component.
New	Creates a new component of standard types: • Folder • IconFolder • TextBlock • BooleanWritable • NumericWritable • EnumWritable • StringWritable
Edit Tags	Opens the Edit Tags window, permitting you to add or remove tags on a component.
Make Template	Creates a template of the selected root component, collects all associated Px and graphic files, and invokes the Template view, allowing you to configure the template for deployment.
Cut	Copies to the clipboard and, after pasting the copy, deletes the cut item.
Copy	Copies to the clipboard; the copied item remains in the clipboard.
Paste	Pastes the copied item from the clipboard to the cursor location.
Duplicate	Makes a copy in the same location (container) as the original item.
Delete	Removes the item from the parent container.
Find	Displays the Component Finder window.
Link Mark	Sets a selected component to your popup menu, making it temporarily available for linking from and linking to other points.
Link From	Links to a selected component from another component that has been marked using Link Mark from the popup menu.
Link To	Links from a selected component to another component that has been marked using Link Mark from the popup menu.
Relation Mark	Sets a selected component to your popup menu, making it temporarily available for relating from and relating to other points.
Relate From	Adds a relation from selected component to another that has been marked using Link Mark from the popup menu.
Relate To	Adds a relation to a selected component from another that has been marked using Link Mark from the popup menu.
Rename	Changes the name of the selected component's actual slot name as it appears in the ORD. This menu item displays the Rename window. You can only rename one component at a time.
Set Display Name	Sets a display name for the selected component as it appears in a Nav tree, and in the Wire Sheet , and Property Sheet views.

Item	Description
Reorder	Opens the Reorder Points window, which provides the following commands for reordering points within the selected parent component. • Move Up • Move Down • Sort by Name • Sort by Type • Reset
Composite	This menu item opens the Composite Editor window.
Export	Exports a selected component to the oBix .xml format.
Refresh	This menu item updates the display of the currently active view.
Go Into	Re-roots the Nav tree at any arbitrary node. Right-click the node and select the Go Into command. This makes that node the new root of the tree. A special type of Bookmark saves the nodes you have gone into. Use the pull-down to switch among them. This feature is quite handy when working with multiple stations or deep file systems and databases.
Pin Slots	Opens the Pin Slots window. Clicking to pin a slot makes that slot visible in the Wire Sheet view. Clicking to un-pin a pinned slot has the opposite effect.
Config Flags	Assigns or removes permissions flags on a component. This is available on properties in the AX Property Sheet . Right-click a property to see the Config Flags window.
More...	Indicates the presence of additional menu items. Click More... to display those items.

Parent topic: [Pop-up menus](#)

Most of the Wire Sheet menu commands are described in another topic in this document.

Figure 1. Wire sheet popup menu



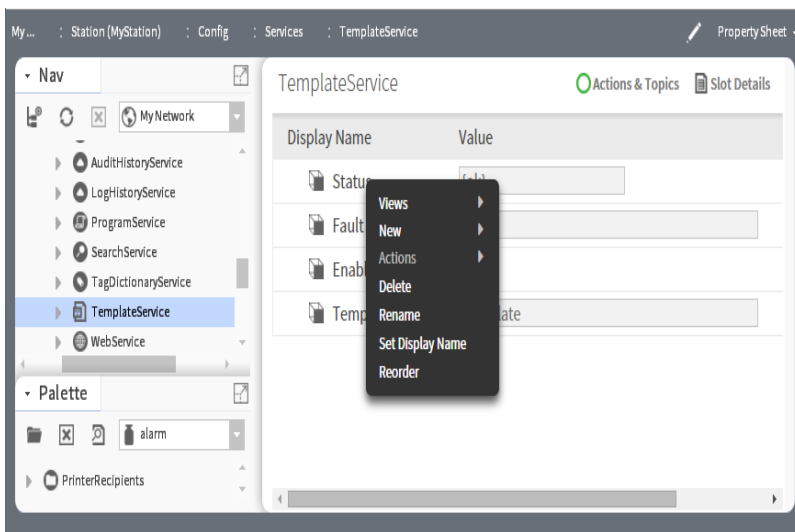
Following are wire sheet specific menu options:

Item	Description
Arrange	Provides options for aligning components on the wire sheet to make them easier to view. • Arrange All — Redistributes the layout of all components on the wire sheet. • Arrange Selection — Redistributes the layout of all selected components on the wire sheet.
Select all	Selects all components on the active wire sheet.

Parent topic: [Popup menus](#)

Most of the **Property Sheet** menu commands are described below.

Figure 1. Property sheet popup menu



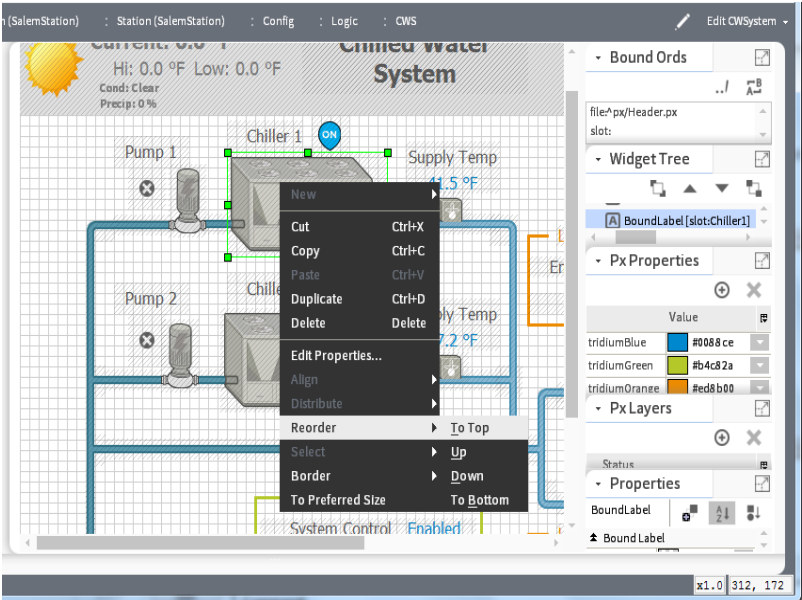
To access this menu, right-click the **Property Sheet**.

Menu item	Description
Views	Selects the different views of the component.
New	Creates a new component, folder points or text block.
Actions	Displays the actions available for the component.
Delete	Removes the component.
Rename	Changes the name of the component.
Set Display Name	Defines the display name of the component.
Reorder	Places components in a different order.
Config flags	Opens the Config window which you can use to add configuration flags on individual slots.

Parent topic: [Popup menus](#)

This popup menu appears when you right-click on an object in the **Px Editor** view. The menu commands are context-sensitive and are dimmed or available, depending on the type of object that you select.

Figure 1. Px Editor popup menu



To access this menu, right-click a **Wire Sheet**.

The following menu commands are on the Px Editor popup menu:

Item	Description
New	Creates a new item of standard types.
Cut	Copies to the clipboard and after pasting the copy, deletes the cut item.
Copy	Copies to the clipboard; the copied item remains in the clipboard.
Paste	Pastes the copied item from the clipboard.
Duplicate	Makes a copy in the same location as the original item.
Delete	Removes the item.
Edit Properties...	Opens a properties window.
Align	• Left This command is available when you select two or more objects in the Px Editor. Choose the Left command to align the left edges of two or more objects along a vertical line. • Right This command is available when you select two or more objects in the Px Editor. Choose the Right command to align the right edges of two or more objects along a vertical line.

Item	Description
	• Top This command is available when you select two or more objects in the Px Editor. Choose the Top command to align the top edges of two or more objects along a horizontal line. • Bottom This command is available when you select two or more objects in the Px Editor. Choose the Bottom command to align the bottom edges of two or more objects along a horizontal line.
Reorder	• To Top This command is available when you select one or more objects in the Px Editor. Choose the To Top command to move the selected object to the top position (inside its parent object) in the Widget Tree. This command will place the object in front (with respect to the view pane, or z-axis) of all other objects in the parent object, but will not move the object out of its parent object. • Up This command is available when you select one or more objects in the Px Editor. Choose the Up command to move the selected object one position higher in the Widget Tree and forward one position in the view pane. This command will not move the object out of its parent object. • Down This command is available when you select one or more objects in the Px Editor. Choose the Down command to move the selected object one position lower in the Widget Tree and back one position in the view pane. This command will not move the object out of its parent object. • To Bottom This command is available when you select one or more objects in the Px Editor. Choose the To Bottom command to move the selected object to the bottom position (inside its parent object) in the Widget Tree. This command will place the object behind (with respect to the view pane, or z-axis) all other objects in the parent object, but will not move the object out of its parent object.
Border	• Add Border This command is available when you select one or more objects in the Px Editor. Choose the Add Border command to wrap selected object(s) with a border pane. Each selection is wrapped in a separate border pane. • Remove Border This command is available when you select one or more border panes in the Px Editor. Choose the Remove Border command to delete selected border pane(s).

Parent topic: [Popup menus](#)

The history extension manager popup menu has the following items.

Item	Description
Views	Provides a submenu that lists all the available views of the history extension manager.
Actions > Update History Id	Refreshes the History Id after a rename. It applies the formatting property (as designated by the enclosing % signs) of the Name Format property to the Id of the History Config Id. For example, if the History Name property under a history extension is set to %parent . name%, the History Config Id is initially named based on this parent display

Item	Description
	name. However, if you rename the parent component, the History Config Id property does not automatically or immediately change. The Update History Id action invokes a renaming of the History Config Id based on the formatting property, so if the parent component (in this example case) is changed, the Update History Id action changes the Id property and, if different from the history name, changes the history name as well.
Go To Point	Displays the property sheet view of the point associated with the selected entry.
Go To History	Displays the default view of the history associated with the selected entry.
Enable Collection	Starts the collection process for the selected entries.
Disable Collection	Stops the collection process for the selected entries.
Rename History	Changes the name of the selected history. This menu item displays the Set History Name window. You can only rename one history at a time.
Edit System Tags	Opens the Set System Tags For Selected History Extensions window. Use this window to edit system tags associated with a single history extension or perform batch edits when you have more than one history extension selected.

Parent topic: [Popup menus](#)

The Todo list menu has the following items.

Item	Description
Add	Opens the Add window, when clicked. Use this menu item to add a new item to your Todo checklist and assign a Summary and Group to the item. This menu item is available even if no items are selected.
Mark Complete	Dims and lines-through the selected item(s) in the Todo list so that the item(s) appear to be crossed off the list. If an item is already marked complete and this menu item is selected, the software restores the item to its uncompleted state. With no items selected, this menu item is dimmed (unavailable).
Edit	Displays the Edit window for changing the Summary and the Group properties associated with the item.
Move to Top	Moves the selected item to the top of the list.
Move Up	Moves the selected item up in the list, one increment per click.
Move Down	Moves the selected item down in the list, one increment per click.
Move to Bottom	Moves the selected item to the bottom of the list.
Remove	Displays a "Remove selected items?" prompt, when clicked; deletes selected items when the prompt is affirmed.

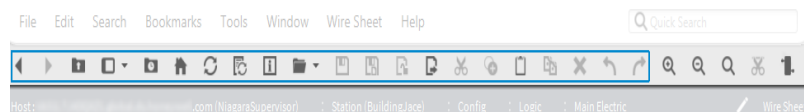
Parent topic: [Popup menus](#)

The point manager popup menu has the following items.

Item	Description
Views	Opens a submenu that lists all the available views of the point manager. These same views are available from the view selector menu when the point extension manager is the active view.
Actions	Opens a list of available actions on the component. For example, write to a writable point.
New	Adds a new component to the points manager. Valid options are presented in the submenu.
Link Mark	Includes a selected point in your menu, making it temporarily available for linking from or linking to other points.
Link From	Links from a selected point to another point that has been marked using Link Mark from the popup menu.
Link To	Links to a selected point from another point that has been marked using Link Mark from the popup menu.
Rename	Changes the name of the selected point. This menu item displays the Set Point Name window. You can only rename one point at a time.
Reorder	Displays the Reorder Points window, which provides the following commands for reordering points within the selected parent component. • Move Up • Move Down • Sort by Name • Sort by Type • Reset
Composite	Displays the Composite Editor window.
New Folder	Adds and name a new points folder.
New	Adds and name a new point.
Edit	Displays the Point Editor window.

Parent topic: [Popup menus](#)

Some icons are always visible as you navigate through the system. Additional icons may be added and removed from view when different views are active. When icons appear dimmed, their functions are unavailable.



Icons that appear on the toolbar are grouped in the following categories:

Category	Description
Standard toolbar icons	These icons may be dimmed (or unavailable in certain views) but they are always present on the toolbar.
Slot sheet toolbar icons	These icons appear only when the Slot Sheet is active.
Px Editor toolbar icons	These icons appear only when the Px Editor or Px Viewer is active.
History extension manager toolbar icons	These icons appear when the History Editor view is active.
History Editor toolbar icons	These icons appear when the History Editor view is active.
Todo list toolbar icons	These icons appear when the Todo List view is active.

- [Standard toolbar icons](#)
The following toolbar icons are available in all views.
- [Slot Sheet toolbar icons](#)
When the **Slot Sheet** view is active, the following additional icons are available.
- [Px Editor toolbar icons](#)
When the **Px Editor** view is the active the following additional icons are available.
- [About the history extension manager toolbar icons](#)
Following, are the icons that appear when the **History Extension Manager** view is active.
- [About the history editor toolbar icons](#)
Following are additional icons that appear when the **History Editor** view is active.
- [About the Todo list toolbar icons](#)
Following are the icons that appear when the **Todo List** view is active.

The following toolbar icons are available in all views.

Icon	Description
 Back	Hyperlinks back to last displayed URL.
 Forward	Hyperlinks forward to the URL selected by the Back command.

Icon	Description
 Up Level	Jumps to the parent of current page.
 Side Bars	Displays the Side Bars list.
 Recent Ords	Displays a recent ORDs list.
 Home	Jumps to the home URL (Workbench splash screen).
 Refresh	Refreshes the current page.
 Refresh Tabs	Refreshes all open tabs.
 Info	Provides information on the current connection.
 Open	Displays the Open list.
 Save(Ctrl + S)	Saves the changes to the current component.
 Save Bog	Saves changes to the bog file.
 Bog File Protection	Enters, changes, or adds a bog file passphrase.
 Export	Exports the current view or object.
 Cut (Ctrl + X)	Cuts the selected item and places it on the clipboard.
 Copy (Ctrl + C)	Copies the selected item and places it on the clipboard.
 Paste (Ctrl + V)	Pastes the items on the clipboard into the current location.
 Duplicate (Ctrl + D)	Copies and pastes to duplicate the item.
 Delete	Deletes the current selection.
 Undo (Ctrl + Z)	Reverses the previous command/action.
 Redo (Ctrl + Alt + Z)	Restores a command/action after the Undo command removed it.

Parent topic: [Toolbar icons](#)

When the **Slot Sheet** view is active, the following additional icons are available.

Icon	Description
 Add Slot	Creates a new slot (appearing as a row) on the slot sheet.
 Rename Slot	Displays the Rename Slot dialog box, when clicked. This icon is dimmed when no slot, or more than one slot is selected in the slot sheet editor view.
 Config Flags	Displays the Config Flags dialog box, when clicked. This icon is dimmed unless one or more slots are selected in the slot sheet editor view.
 Reorder	Displays the Reorder dialog box, when clicked.

Parent topic: [Toolbar icons](#)

When the **Px Editor** view is the active the following additional icons are available.

For more details, refer to the *Niagara Graphics Guide*.

Icon	Description
 Toggle View/Edit Mode	Displays, alternately, the Px Editor and the Px Viewer in the view pane. This icon appears inset when the Px Editor view is active and it appears normal when the Px Viewer is active.
 Toggle Browser Preview Mode	Visible in Px Viewer mode, displays, alternately, the Px Viewer and the Browser Preview in the view pane. This icon appears inset when the Browser Preview view is active and it appears normal when the Px Viewer is active.
 Right (Px Editor) side bar menu	Displays a drop-down list of sidebar options for the Px Editor . The following options are available: • Bound Ords shows or hides the Bound Ords sidebar. • Widget Tree shows or hides the Widget Tree sidebar. • Properties shows or hides the Properties sidebar.
 Left align	Aligns left edges of selected objects along a vertical line.
 Right align.	Aligns right edges of selected objects along a vertical line.
 Top align	Aligns top edges of selected objects along a horizontal line.
 Bottom align	Aligns bottom edges of selected objects along a horizontal line.
 To Top	Moves selected objects to the highest position (with regard to z-order) in the parent object.
 To Bottom	Moves selected objects to the lowest position (with regard to z-order) in the parent object.
 Select	Activates the pointer tool used to select objects in the Px Editor view using the mouse.

 Add Polygon	Activates the polygon tool for drawing polygons.
 Add Path	Activates the path tool that allows you to draw bezier curves in the Px Editor view.
 Add Point	Activates the Add Point tool that allows you to add a point to a path or a polygon in the Px Editor view.
 Delete Point	Activates the Delete Point tool that allows you to remove a point from a path or a polygon in the Px Editor view.

Parent topic: [Toolbar icons](#)

Following, are the icons that appear when the **History Extension Manager** view is active.

Icon	Description
 HistoryExtManager Enable Collection	Start the collection process for the selected entries.
 HistoryExtManager Disable Collection	Stops the collection process for the selected entries.
 HistoryExtManager Rename History	Renames the selected history. When clicked, the Set History Name window opens.
 HistoryExtManager Edit System Tags	Opens the Set System Tags For Selected History Extensions dialog box. Use this dialog box to edit system tags associated with a single history extension or perform batch edits when you have more than one history extension selected.

Parent topic: [Toolbar icons](#)

Following are additional icons that appear when the **History Editor** view is active.

Menu	Description
 Hide	With one or more records selected, sets the trend flag of all selected records to hidden. With no records selected, the icon is dimmed (unavailable).
 Unhide	Restores visibility to previously-hidden records. With no records selected, the icon is dimmed (unavailable).
 Filter	Opens the Configure Flags window. This icon is available even if no records are selected.
 Configure Outliers	Opens the Configure Outliers window.

Parent topic: [Toolbar icons](#)

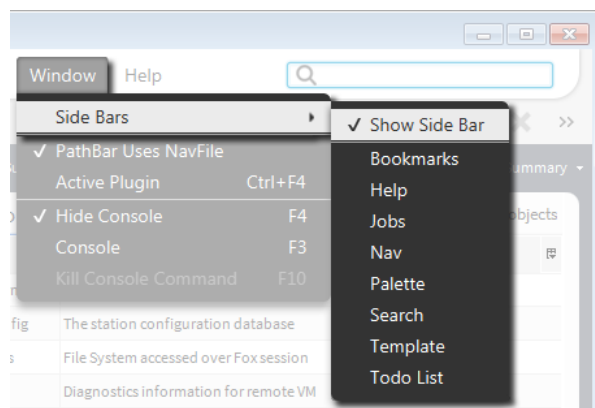
Following are the icons that appear when the **Todo List** view is active.

Icon	Description
 Add	Opens the Add window. Use this icon to add a new item to your Todo checklist and assign a Summary and Group to the item. This icon is available even if no items are selected.
 Mark Complete	Dims and lines-through the selected item(s) in the Todo list so that the item(s) appear to be crossed off the list. If the item is already marked as completed, clicking this icon restores the item to its un-crossed-off state. With no items selected, this icon is dimmed (unavailable).
 Edit	Displays the Edit window, allowing you to change the Summary and the Group fields associated with the item.
 Move to Top	Moves the selected item to the top of the list.
 Move Up	Moves the selected item up in the list, one increment per click.
 Move Down	Moves the selected item down in the list, one increment per click.
 Move to Bottom	Moves the selected item to the bottom of the list.
 Remove	Displays a “Remove selected items?” prompt and deletes selected items when you affirm the prompt.

Parent topic: [Toolbar icons](#)

The interface may be customized by adding unique sidebars that are designed to fit particular applications.

Figure 1. Default sidebar menu



To access this menu, click **Window** and select **Side Bars**.

The following sidebars may be displayed in the sidebar pane by default.

Sidebar	Description
Bookmarks	Displays a list of your bookmarks.
Help	Provides a tree view of available help documentation.
Jobs	The Jobs sidebar lists all current jobs in all of the stations with which you have a connection. The current status of each job is shown.
Nav	Provides a tree view of the system.
Palette	Provides a tree view of components that are available in specific palettes.
Search	Allows you to enter search queries and access cached results from your previous queries. Also, you can edit Search Settings (requires super user permissions).
Template	Provides access to all template files located in your Workbench User Home ~templates folder, as well as to any templates stored in modules located in the SysHome !modules folder.
Todo List	Provides a customizable list of tasks or notes.

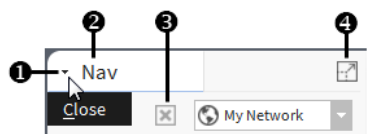
- [About the sidebar title bar](#)
All sidebars have a title bar across the top. You can click and drag on the title bar to vertically resize the sidebar or you can click on a minimized sidebar to restore it to the previous size.
- [About the Bookmarks sidebar](#)
When you open the **Bookmarks** sidebar, it appears in the sidebar pane, as shown below.
- [About the Help sidebar](#)
When you open the **Help** sidebar, it appears in the sidebar pane, as shown below.

- [About Jobs sidebar](#)
When you open the Jobs sidebar, it appears in the sidebar pane. The Jobs sidebar contains a list of jobs that have been performed or that are currently being performed.
- [About the Nav sidebar](#)
The Nav sidebar contains a hierarchical view of the whole system, referred to commonly as a tree. The metaphor is better described as an upside-down tree with the main root, My Host at the top.
- [About the Palette sidebar](#)
This sidebar provides a place to open and view sets of modules or custom palettes that you build for yourself.
- [About the Search sidebar](#)
The SearchService uses Niagara Entity Query Language (NEQL) syntax to query the system for component tags.
- [Template sidebar](#)
This sidebar provides access to template files located in the Workbench User Home ~templates folder as well as to templates stored in modules located in the SysHome ! modules folder.
- [About the Todo list sidebar](#)
The **Todo List** sidebar is a convenient way to create and access Todo List items from a palette.
- [About the Jobs sidebar](#)
The **Jobs** sidebar shows all the current jobs in all the stations with which you have a connection.
- [About the bound ords sidebar](#)
The bound ords sidebar is available when the **Px Editor** view is active. It displays a listing of all the bound ords in the current Px view.
- [About the widget tree sidebar](#)
The widget tree displays a tree hierarchy of the widgets (panes, labels, graphic elements, and so on) that are in the current Px view.
- [About the Px properties sidebar](#)
This sidebar is available when the **Px Editor** view is active. It displays a listing of all the Px properties that are defined in the currently active Px file.
- [About the Properties sidebar](#)
This sidebar is available when the **Px Editor** view is active. It displays a listing of all the properties that are in the currently selected object in the Px view.

All sidebars have a title bar across the top. You can click and drag on the title bar to vertically resize the sidebar or you can click on a minimized sidebar to restore it to the previous size.

In addition, all sidebars have the following controls:

Figure 1. Sidebar title bar



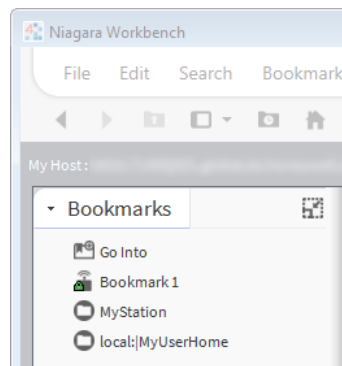
1	Close drop-down closes the sidebar.
2	Pane title identifies the type of sidebar.
3	Close is an alternative to the Close drop-down. It closes the sidebar.

- 4 Expand/Restore expands the sidebar to fill the sidebar pane and collapses other open sidebars. Clicking it again restores the normal sidebar display.

Parent topic: [Sidebars](#)

When you open the **Bookmarks** sidebar, it appears in the sidebar pane, as shown below.

Figure 1. Bookmarks side bar

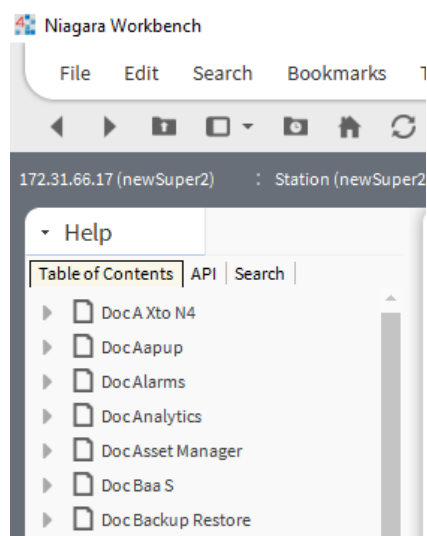


From the bookmark sidebar, you can double click on bookmark nodes or use popup menus to perform all operations that are available from the sidebar (for example, go directly to a bookmarked location, manage bookmarks, edit bookmarks, and more). The quick access provided here is very helpful for changing screens without having to go through multiple selections using other menus or submenus.

Parent topic: [Sidebars](#)

When you open the **Help** sidebar, it appears in the sidebar pane, as shown below.

Figure 1. Help sidebar



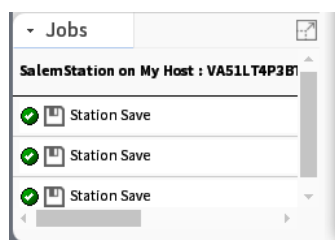
The help sidebar has three tabs that you may select by clicking on the tab. The three help tabs are listed and briefly described, as follows:

Tab	Description
Table of Contents tab	Contains a tree view of help topics, listed in alphabetical order by topic.
API tab	Contains a tree view of help topics, listed in alphabetical order by module.
Search tab	Contains a Find : text entry field and Search button.

Parent topic: [Sidebars](#)

When you open the Jobs sidebar, it appears in the sidebar pane. The Jobs sidebar contains a list of jobs that have been performed or that are currently being performed.

Figure 1. Jobs side bar

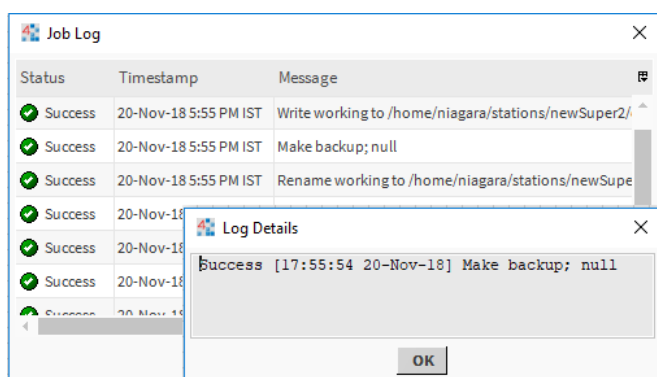


The following icons on the Jobs sidebar indicate job status:

Menu	Description
Running (🔄)	Indicates that the job is currently running.
Success (✅)	Indicates that the job has completed without error.
Failed (❌)	Indicates that the job did not complete.
Unknown (❓)	Indicates that the job status is not available.

From the Jobs sidebar, you can click on the arrow icon >> to open the **Job Log** window. The **Job Log** window displays a listing of the actions performed as part of the job. Each entry in this log contains a detailed description that you can view by double-clicking on the entry to open the **Log Details** window as shown below.

Figure 2. Log Details window

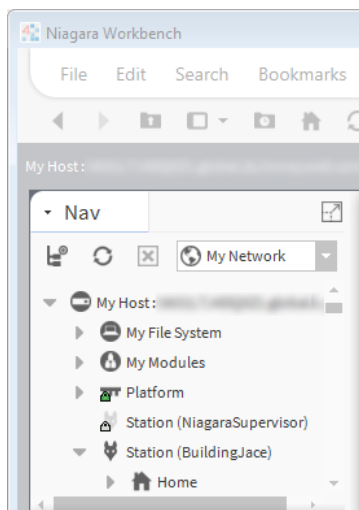


Parent topic: [Sidebars](#)

The Nav sidebar contains a hierarchical view of the whole system, referred to commonly as a tree. The metaphor is better described as an upside-down tree with the main root, My Host at the top.

The Nav tree sidebar opens by default when you start Workbench

Figure 1. Nav tree



Each node in the structure includes a symbol or icon that reflects its type. If the node is a file, the symbol represents the type of file.

At the highest level, the Nav tree may include the following (when working from a localhost, as shown):

- **My Host** identifies the hardware system that hosts the framework.
- **My File System** provides access to the system and user homes as well as the local computer's drive(s).
- **My Modules** lists the software units that comprise this instance of the framework.

- **My Tools** makes selected tools available during the current session when no station is connected. When you open Workbench, a **My Tools** node is not present in the Nav tree. It appears when you click **Tools** and select any tool. **My Tools** is provided as a convenience and may be useful in only a limited number of situations. When connected to a station, you would use the services under the **Services** folder of the specific station.
- **Platform** connects the framework to your PC or remote controller.
- **Station(s)** (connected or disconnected) represent the stations being managed by the current instance of Workbench

You can double-click on each node and use popup menus to perform all operations that are available from the Nav tree. For example, you can double-click to connect or disconnect to a station, refresh a tree node, and more). The expandable tree provided here is very useful for performing actions on nodes and for navigating among views.

- [Types of nodes in the Nav tree sidebar](#)
The **Nav** tree sidebar may include several different types of nodes and child nodes.
- [About the Nav tree sidebar toolbar](#)
In addition to the standard sidebar title bar the Nav tree sidebar has a toolbar, located just below the title bar.

Parent topic: [Sidebars](#)

The **Nav** tree sidebar may include several different types of nodes and child nodes.

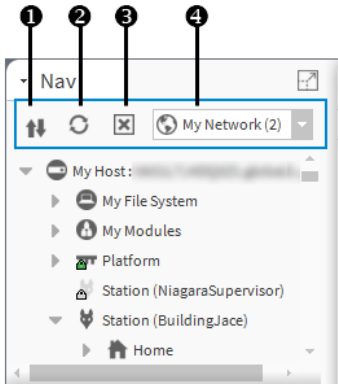
Menu	Description
My Host node	Represents a physical computer (hardware) that the rest of the nodes (subnodes) reside on
My File System node	Represents the top level of a tree view of the host file system. File system subnodes represent drives and locations on the host system. It is important to understand that the file system provides access to files that are outside of the station database.
My Modules node	When expanded, displays a tree view of available modules, listed in alphabetical order by module.
Platform node	When expanded, displays a hierarchical view of the Niagara host platform. You can double-click on the platform node and sub-nodes, or use a right-click shortcut menu to perform all operations that are available on or under this node (connecting, disconnecting, refreshing, and more). For details on the platform node and its subnodes, refer to <i>Niagara Platform Guide</i> for more details.
Station node	Represents a station (connected or disconnected). When expanded, the station node displays the station contents in a hierarchical tree. You can double-click on the station node and sub-nodes, or use a right-click shortcut menu to perform all operations that are available on or under this node (connecting, disconnecting, selecting views, and more). • Home • Alarm • Config node - When expanded, displays a tree view of the station contents or “configuration”. The config node usually contains one or more of the following types of nodes: – Services

Menu	Description
	Component for storing services, such as alarm service, history service, tagdictionary service, and more. – Drivers Provides a place to store driver modules (such as the NiagaraNetwork, BACnet drivers, Modbus, and more). When expanded, displays a tree view of loaded driver modules. For more details, see the <i>Niagara Graphics Guide</i>. – Apps – Schedules – Control or Logic Control points may be displayed directly in the root of the Config node.

Parent topic: [About the Nav sidebar](#)

In addition to the standard sidebar title bar the Nav tree sidebar has a toolbar, located just below the title bar.

Figure 1. Nav side bar tool bar



1	Recorder Tree button — opens a new Recorder Tree window to make changes in the tree nodes.
2	Sync tree button — synchronizes the tree node display with the currently selected view.
3	Close tree button — closes the currently displayed sidebar.
4	Drop-down tree selector — when multiple Nav sidebars are open, this selector allows you to choose which one to display.

- [Recorder Tree](#)
You can click the **Recorder Tree** button on the Nav side toolbar to open the **Recorder Tree** window.

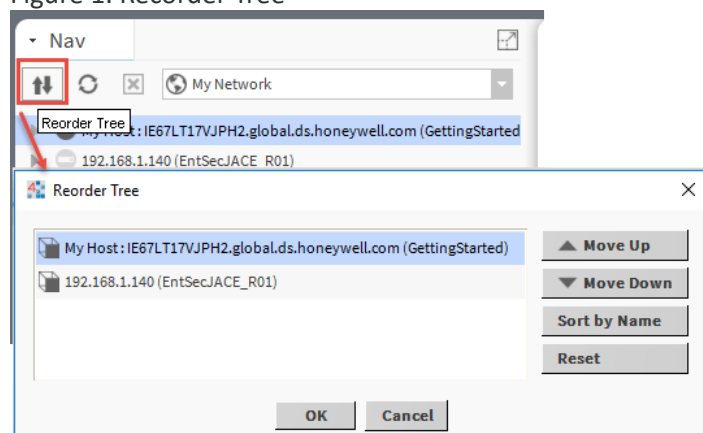
Parent topic: [About the Nav sidebar](#)

You can click the **Recorder Tree** button on the Nav side toolbar to open the **Recorder Tree** window.

There are four buttons in the **Recorder Tree** window:

- **Move Up**—When you have more than one tree node, you can move up the tree node.
- **Move Down**—When you have more than one tree node, you can move down the tree node.
- **Sort By Name**—You can sort by name.
- **Reset**—You can reset the tree node.

Figure 1. Recorder Tree

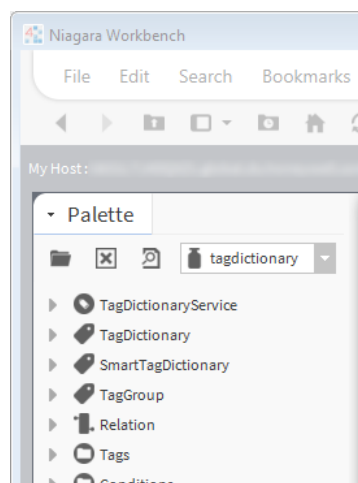


Parent topic: [About the Nav tree sidebar toolbar](#)

This sidebar provides a place to open and view sets of modules or custom palettes that you build for yourself.

When you open the **Palette** sidebar, it appears on the left side of the Workbench in the sidebar pane.

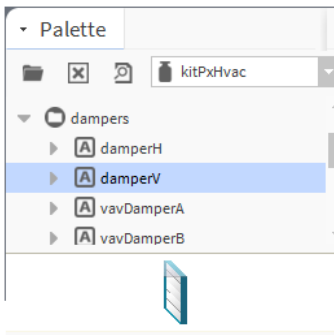
Figure 1. Palette side bar with preview pane



From the Palette sidebar, you can open multiple palettes, close palettes and view modules within palettes. You may also double-click or use popup menus to perform all operations that

are available from the Palette sidebar (for example, copy modules, select a module view, refresh the tree node, and more). The expandable tree provided in the palette allows you to perform actions on nodes within the palette and to navigate through the palette sub-directories. Items are displayed in the tree with an icon that represents an associated function or file type.

The palette sidebar also has a component preview pane (shown below) that displays an image (when available) of the selected component.
Figure 2. Palette preview pane



Palette previews display in the palette when components have images configured either as the default image property or as the image assigned to the `comPreviewWidget` property. If no preview is associated with a component, you can add a `comPreviewWidget` property to a widget to display an image in the preview pane of the palette sidebar.

- [About the Palette sidebar toolbar](#)
In addition to the standard sidebar title bar the **Palette** sidebar has a toolbar, located just below the title bar.
- [About the Open Palette window](#)
This window displays a tabular list of available palettes. If there are palettes located in locations other than the **My Modules** directory, you can use the browse button to find them.

Parent topic: [Sidebars](#)

In addition to the standard sidebar title bar the **Palette** sidebar has a toolbar, located just below the title bar.

Figure 1. Palette side bar tool bar



The palette toolbar includes the following:

1	Open palette button — opens the Open Palette window.
2	Close palette button — closes the currently displayed palette.

- 3

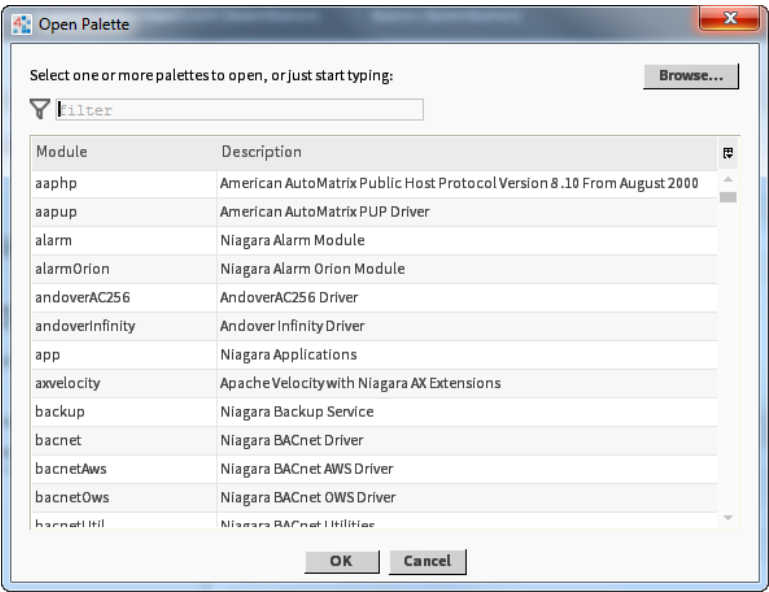
Preview button — toggles the preview pane on and off. Previews are available on some components.
- 4

Dropdown palette selector — when palettes are open in the palette sidebar, this selector allows you to choose which palette to display.

Parent topic: [About the Palette sidebar](#)

This window displays a tabular list of available palettes. If there are palettes located in locations other than the **My Modules** directory, you can use the browse button to find them.

Figure 1. Open Palette window



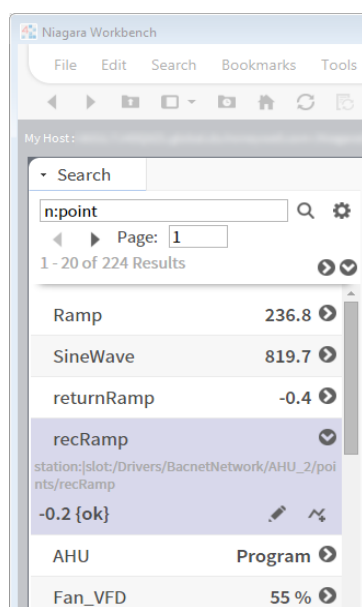
To open this window, click the Open Palette folder icon (📁) at the top of the **Palette** sidebar.

Item or column	Description
Filter	Filters the list when you type the beginning letters of the palette name . For example, typing in the letters “mo” removes palettes that do not begin with the letters “mo”. You may use the * (asterisk) character as a wild card. If you enter no filter, the table lists all palettes.
Browse button	Opens the File Chooser window for selecting palettes that are located in alternate locations.
Module	Displays the name of the palette’s parent module.
Description	Displays a short title or name for the palette’s parent module.

Parent topic: [About the Palette sidebar](#)

The SearchService uses Niagara Entity Query Language (NEQL) syntax to query the system for component tags.

Figure 1. Search side bar



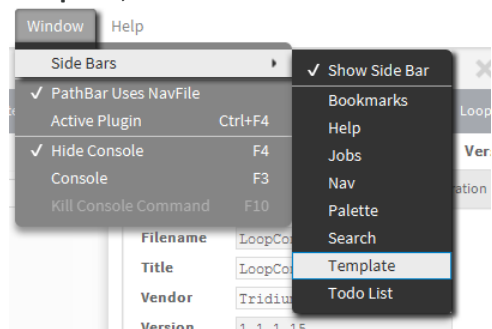
The **Search** sidebar provides a query field for entering your search criteria. For example, you might enter “n:point” (as shown) to query for all points in the station that have the Niagara tagdictionary point tag.

Click the gear icon to configure the number of search results to display per page. The “>” or “<” page through the results. The search results area provides access to additional information too. Click the “>” icon to the right of a result to expand it, showing the Ord and current status. In an expanded result, click the icons (at right) to view the live data either in a gauge or chart.

Parent topic: [Sidebars](#)

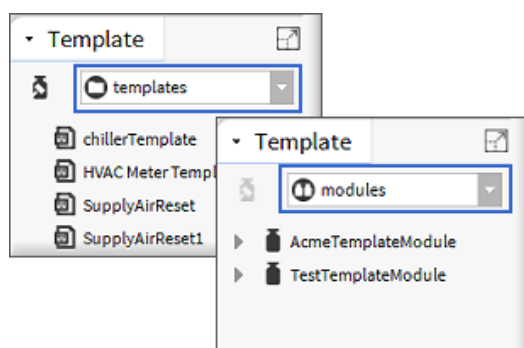
This sidebar provides access to template files located in the Workbench User Home ~templates folder as well as to templates stored in modules located in the SysHome !modules folder.

If not already visible in Workbench, display the sidebar by clicking **Window > sidebars > Template**, as shown here.



In the sidebar, the pull-down list switches the view between the ~templates and !modules folders. When the !modules folder is selected, click to expand any module to see the template files contained within.

Figure 1. Template sidebar



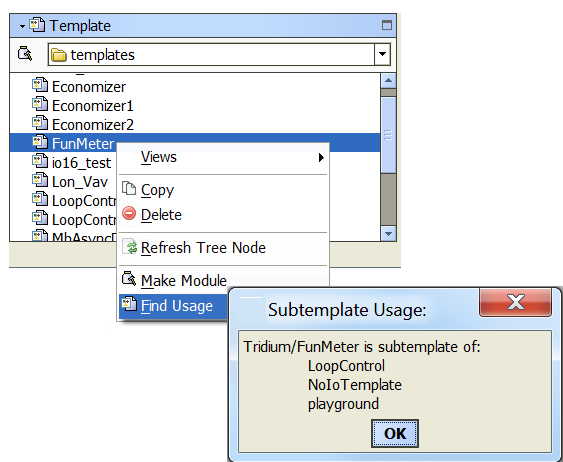
Double-click on a template file to open it in the **Template** view. When you open a file in the templates folder you can proceed to make changes and save the file. Optionally, you can create a new variation of an existing template by clicking **Save As** in the view to save it with a new name.

Note: Any template stored in a module is a read-only file which you cannot edit. When you open a template in a module, you will see “ReadOnly” in the top left corner of the **Template** view. To make changes you must first click **Save As** and save the template with a different filename in the Workbench User Home ~templates folder.

Find Usage option

In Niagara, the **Template Sidebar** has an added **Find Usage** option on the right-click menu. This menu option invokes the **Subtemplate Usage** window (as shown) which provides a list of parent templates that contain this template. Finding usage is helpful if you have modified a template by adding Inputs, Outputs, or Relations. You should review its usage in the other templates as you will likely need to resolve these new additions within any parent template.

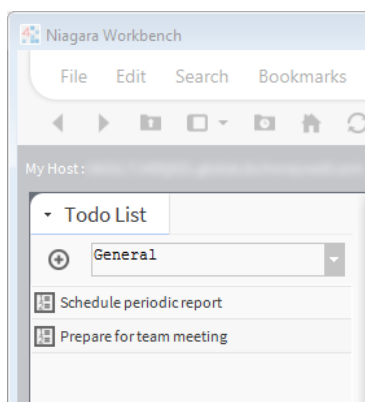
Figure 2. Find Usage menu option invokes Subtemplate Usage window



Parent topic: [Sidebars](#)

The **Todo List** sidebar is a convenient way to create and access Todo List items from a palette.

Figure 1. Todo bar

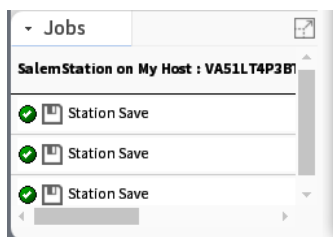


Click the **+** button on the toolbar to open the **Add** window. This window provides a text fields for adding and categorizing Todo list items.

Parent topic: [Sidebars](#)

The **Jobs** sidebar shows all the current jobs in all the stations with which you have a connection.

Figure 1. Jobs side bar



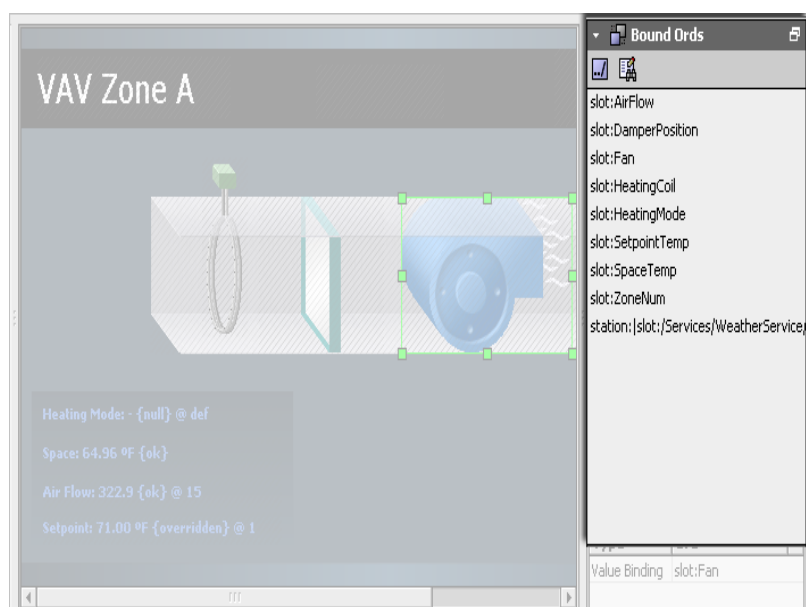
The current status of each job is shown as: running, canceling, canceled, success, or failed. If the job is running, a progress bar displays estimated progress.

You may cancel a running job by pressing the **Cancel** icon. Normally, once a job has completed you are notified via the async notification feature. You may then dismiss the job by pressing the **Close** icon. The details of the job may be accessed using the ">>" icon to display the **Job Log** dialog box.

Parent topic: [Sidebars](#)

The bound ords sidebar is available when the **Px Editor** view is active. It displays a listing of all the bound ords in the current Px view.

Figure 1. Bound ords side bar

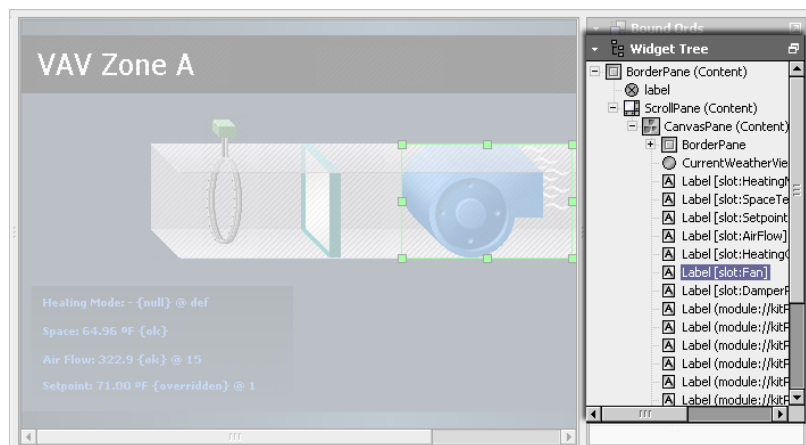


Double click on any ORD in the list to display the ORD in the **ORD editor** window.

Parent topic: [Sidebars](#)

The widget tree displays a tree hierarchy of the widgets (panes, labels, graphic elements, and so on) that are in the current Px view.

Figure 1. Widget tree sidebar

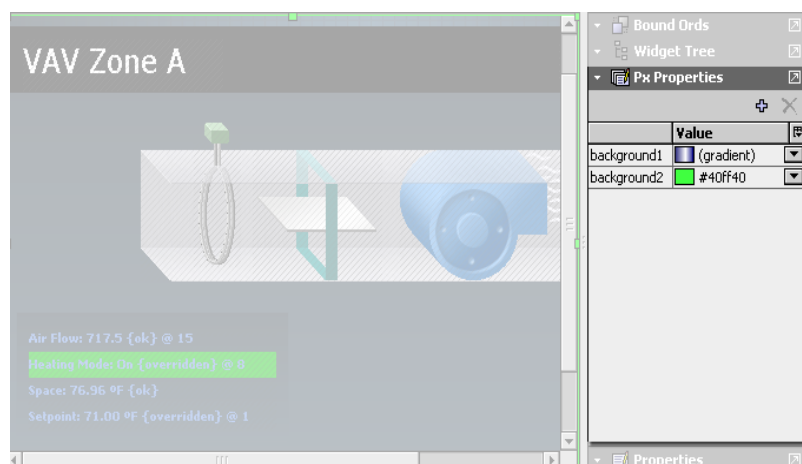


It is often easier to use the Widget Tree to select objects when you have a lot of objects on a view—especially when there are several layers of objects. When you select an object in the tree view it is selected in the Px view as well and displays the selection borders and handles.

Parent topic: [Sidebars](#)

This sidebar is available when the **Px Editor** view is active. It displays a listing of all the Px properties that are defined in the currently active Px file.

Figure 1. Px properties sidebar

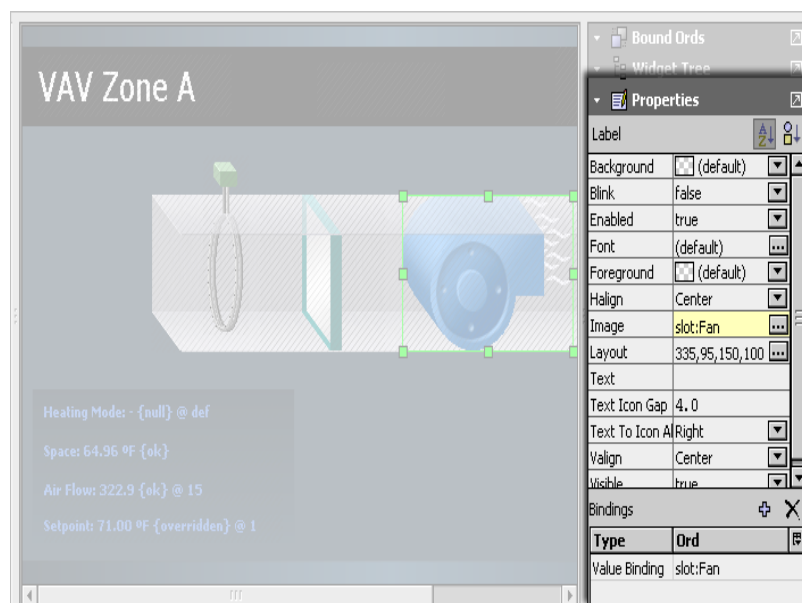


Use the menu bar icons to add, define, assign, and delete Px properties. For more information about px Properties, see the *Niagara Graphics Guide*.

Parent topic: [Sidebars](#)

This sidebar is available when the **Px Editor** view is active. It displays a listing of all the properties that are in the currently selected object in the Px view.

Figure 1. Properties sidebar

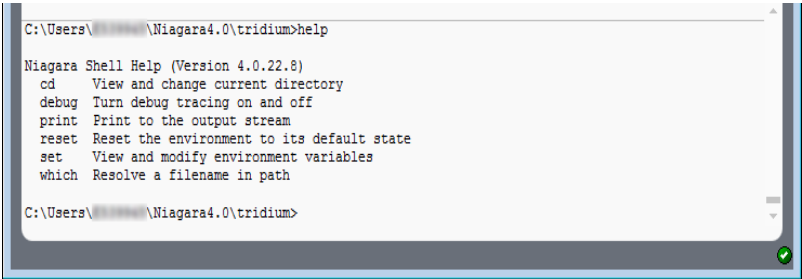


Double click on any object in the widget tree or in the in the Px viewer to display the **Properties** window (same information as the properties sidebar).

Parent topic: [Sidebars](#)

The console commands provide additional functionality. They function at the command prompt.

Figure 1. Example of a console



These commands are used at the command line.

Command	Description
Shell commands	May be typed in directly
nre commands	May be typed in using the “nre” prefix.
wb commands	May be typed in using the “wb” prefix.
plat commands	May be typed in using the “plat” prefix.

- [Shell commands](#)
These commands function at the command prompt.
- [nre \(station\) commands](#)
This command works with the station.
- [wb \(Workbench\) console commands](#)
The **wb** command starts up an instance of Workbench.
- [plat \(platform\) commands](#)
Provides platform-based commands.

These commands function at the command prompt.

Menu	Description
cd	Displays and changes the current directory. Type <code>cd <directory name></code> to change to a specific directory. Type <code>cd</code> to display the current directory.
debug	Turns debug tracing on and off. Type <code>debug on</code> to turn on debug. Type <code>debug off</code> to turn off debug.
print	Prints a message to the output stream. Type <code>print <message></code> to print the literal message to the console.
reset	Resets the environment to its default state. Type <code>reset</code> to set the environment to its default state.
set	Displays and modifies environment variables.

- Type `set` to display all the environment variables.
- Type `set <prefix>` to display all the environment variables that start with the specified prefix.
- Type `set <name>=` to remove the “named” variable.
- Type `set <name>=<value>` to set the “named” variable to the “value” specified.

`which` Resolves a filename in a path. Type `which <filename>` to find the first occurrence of the specified filename.

Parent topic: [Console commands](#)

This command works with the station.

Use the following syntax with the `nre` command.

```
nre [options] <class> [args]*
```

The following parameters may be used with the `nre` command.

Command	Description
<code>class</code>	Defines a class name or a module:classname to execute.
<code>args</code>	Provides the name of one or more arguments to pass through to main.

The following options may be used with the `nre` command.

Option	Description
<code>-version</code>	Displays the <code>nre</code> version.
<code>-modules:<x></code>	Displays the modules that match the pattern defined by “x”.
<code>-hostid</code>	Displays the Host ID for the system host.
<code>-licenses</code>	Displays a summary of the license information.
<code>-props</code>	Displays a list of the system properties.
<code>-locale:<x></code>	Sets the default locale. For example, to set the default locale to US English, type: <code>-locale:en_US</code>
<code>-@option</code>	Passes the specified option to the Java VM.
<code>-testheap</code>	Tests and displays the max heap size.
<code>-buildreg</code>	Triggers a rebuild of the registry.

Parent topic: [Console commands](#)

The `wb` command starts up an instance of Workbench.

Use the following syntax with the `wb` command:

```
wb [options] <ord>
```

The following parameter may be used with the `wb` command.

Parameter	Description
ORD	Specifies the ORD of the initial view that you want display when Workbench starts up.

The following options may be used with the `wb` command.

Option	Description
<code>-profile</code>	Specifies the Workbench profile to assign when Workbench starts up.
<code>-file:ord</code>	Specifies the initial file to display when Workbench starts up.
<code>-locale<x></code>	Sets the locale on startup.
<code>-@<option></code>	Passes the specified option to the Java VM.

Parent topic: [Console commands](#)

Provides platform-based commands.

Use the following syntax with the `plat` command:

```
plat <command> <flags> <command-flags>
```

The following commands may be used with `plat`.

Command	Description
<code>details</code>	Displays a configuration summary for a remote host.
<code>fget</code>	Retrieves one or more files from a remote host.
<code>flist</code>	Provides file details for a single file, or for all files in a directory.
<code>ipconfig</code>	Displays the TCP/IP configuration for a remote host.
<code>jacejar</code>	Creates module files that can be run on embedded hosts.
<code>liststations</code>	Lists stations that are managed by the platform daemon.
<code>moduleinstall</code>	Installs modules on a remote host.
<code>reboothost</code>	Requests that a remote platform daemon reboot its host.
<code>script</code>	Runs one or more platform commands in a script.
<code>startstation</code>	Requests that the platform daemon start a station.
<code>stopstation</code>	Requests that the platform daemon stop a station.
<code>tellstation</code>	Sends text to the console of a running station.

Command	Description
watchstation	Monitors the output of the station.
installdaemon	Installs the platform service (Win32 only).
uninstalldaemon	Removes the platform service (Win32 only).
installdialup	Installs the dialup service (Win32 only).
uninstalldialup	Removes the dialup service (Win32 only).

The following options may be used with the `plat` command.

Option	Description
-usage	plat -usage prints the help listing in the console. plat <command> -usage prints the command specific usage in the console.
-?	plat -? prints the help listing in the console.
-help	plat -help prints the help listing in the console plat <command> -help prints the command specific usage in the console
-locale:<x>	Sets the default locale (en_US).
-@<option>	Passes the option to the Java VM
-buildreg	Forces a rebuild of the registry.

Parent topic: [Console commands](#)

This chapter provides summary information on components.

Component reference summary

Note: For kitControl components, see the *Niagara KitControl Guide*.

Summary information is provided on components in the following modules:

- alarm
- backup
- baja
- chart
- control
- converters
- crypto
- email
- file
- flr
- help
- history
- net
- onCall
- program
- schedule
- sms
- web
- workbench
- [Components in backup module](#)
This module provides the BackupService component and other related components.
- [Components in baja module](#)
The baja (Building Automation Java Architecture) module contains a set of classes and APIs that facilitate the framework.

- [Components in chart module](#)
Components in chart module are:
- [Components in control module](#)
There are three folders in the **control** palette: Points, Extensions, and Trigger.
- [Components in file module](#)
- [Components in help module](#)
- [Components in net module](#)
The net module contains the components listed below.
- [Components in program module](#)
- [Components in web module](#)
- [Components in workbench module](#)

This module provides the BackupService component and other related components.

- [backup-BackupService](#)
This component provides for complete configuration backups to the Workbench PC or a browser PC (user with Wb web profile). By default, the BackupService is included when you create a new station using the **New Station** wizard. The target host (JACE, Supervisor) must have the backup module installed.
- [baja-FoxBackupJob](#)
This component appears as a child of the **JobService**.

Parent topic: [Component guides](#)

This component provides for complete configuration backups to the Workbench PC or a browser PC (user with Wb web profile). By default, the BackupService is included when you create a new station using the **New Station** wizard. The target host (JACE, Supervisor) must have the backup module installed.

The default view of a station's **BackupService** is the **BackupManager**, which provides a **Backup** button to manually initiate a backup. A backup automatically performs a local station save first, and is run as a standard station job. This means each backup provides a progress bar, and upon completion, a popup notification. Under the station's **JobService**, any backup appears as a Fox Backup.

About a backup .dist

A backup is saved as a .dist file (a zipped format) including, minimally, the station's config.bog, current station console output (.txt file), and backup.log file. If other station file types and subfolders are not excluded (in the **BackupService** configuration), the backup .dist file contains them too—for example, files of type: px, nav, html, jpg, png, and so forth.

The backup .dist contains the zipped nre-config for that host (including license and certificate files), as well as pointers to the installed nre-core, OS, and JVM, each by version. Essentially, a backup .dist provides a configuration snapshot of the entire controller platform in zipped .dist file format. This allows for a complete image restoration.

CAUTION: Be careful to keep backup .dist files in a secure location. They have always contained sensitive information, for example a station's config.bog file. They also may contain sensitive host platform information. In update releases, this includes unique key ring files used for client password encryption.

Not included in a backup is runtime information that is actively managed by the station, such as the alarm and history databases. These data should be backed up using standard alarm routing and history archiving to a Supervisor host.

The default backup destination depends on your station connection, as either:

- Workbench(Fox) — !backups

A subdirectory backups under your software release directory. If you have not previously made station backups, the software automatically creates this directory.

- Browser access (Wb Web Profile) — !backups

A niagara\wbapplet\backups folder under your Windows user profile location.

The default name for a backup file uses a format of:
backup_stationName_YYMMDD_HHMM.dist

For example, backup_demo_250412_1429.dist for a backup made of station “demo” on April 12, 2025 at 2:29 pm.

Restoring a backup

To restore a backup .dist using Workbench, you open a platform connection to the controller and use the platform **Distribution File Installer** to install the backup .dist file.

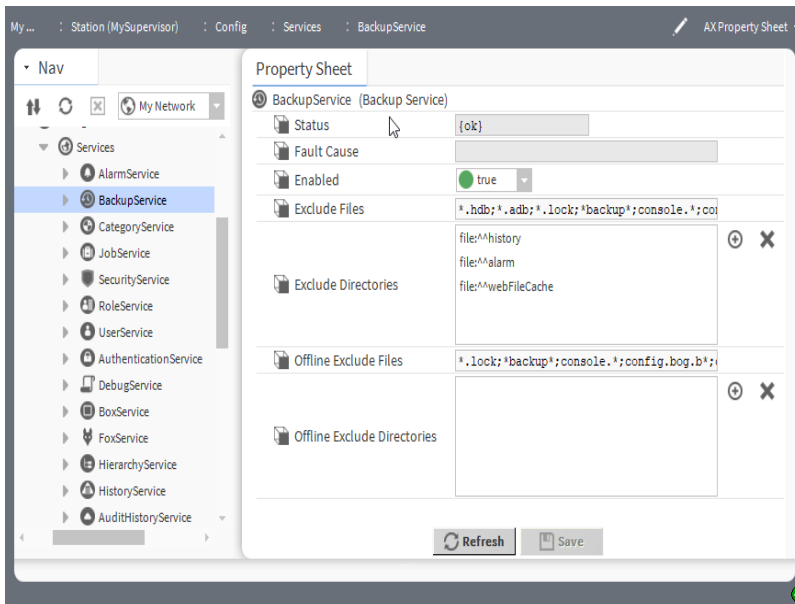
- Restoring from a backup may be necessary if the host failed in some manner to allow recovery (to the same hardware, or to replacement hardware).
- Another usage is to install the same backup .dist file on multiple hosts, such that each host (controller) has the identical configuration, including station database. When performing these replicated host installations of a backup .dist, the platform **Distribution File Installer** allows you to choose if TCP/IP settings from the backup .dist should be restored (the default is to not). Typically you do not, as TCP/IP settings must be unique on each host.

For details, see the *Niagara Platform Guide*.

BackupService configuration

Configuration lets you define the file types and/or folders not included in a station backup. The service’s property sheet provides the following properties for configuration:

Figure 1. BackupService properties

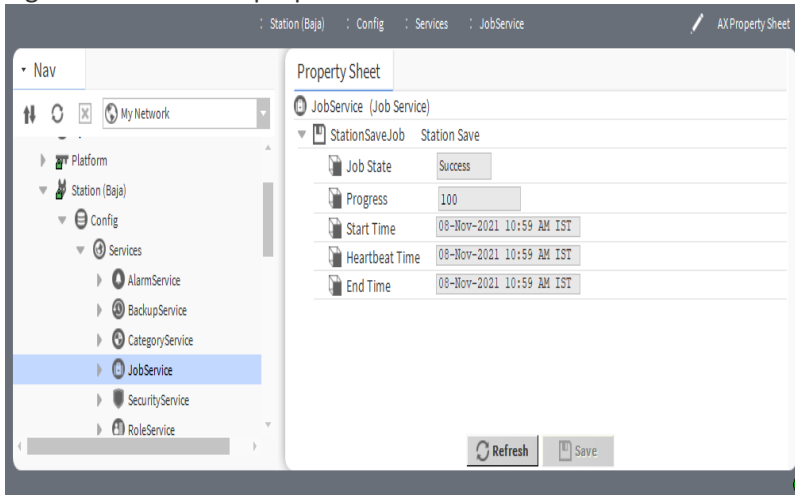


Property	Value	Description
Enabled	true or false	Activates (true) and deactivates (false) use of the object (network, device descriptor, etc.).
Exclude Files	string of names (each delimited by a semi-colon (;))	Specifies file types to exclude from the backup dist, either by name or extension. By default, the following files are excluded: *.hdb;*.adb;*.lock;*backup*;console.*;con
Exclude directories	string of Ords with Ord Chooser control	Specifies station subdirectories to exclude from the backup dist, using relative Ord syntax. An Ord chooser control provides a Directory Chooser window in which to select station subfolders. By default, the following subfolders are excluded: file:^history, file:^alarm
Offline Exclude Files	string of names (each delimited by a semi-colon (;))	Specifies file types to exclude from the backup dist, when the station is offline. By default, the following files are excluded: *.lock;*backup*;console.*;config.bog.b*;config_k Note: History (*.hdb) and alarm (*.adb) files are backed up, unlike w
Offline Exclude Directories	string of Ords with Ord Chooser control	Specifies station subdirectories to exclude from the backup dist, when the station is offline. Directories are specified using relative Ord syntax. An Ord chooser control provides a Directory Chooser window in which you can select station subfolders. By default, no directories are

Parent topic: [Components in backup module](#)

This component appears as a child of the **JobService**.

Figure 1. Job Service properties



To access, expand **Config > Services** and right-click **JobService > Views > AXProperty Sheet**.

Properties relate to the save job:

Note: A station restart clears all jobs.

Property	Value	Description
Job State	text	Displays the current or final state of the job. The first three states appear on the Device Network Job view. Unknown indicates the job is pending execution. Running indicates the job is executing. Cancelling indicates a request to cancel the job was sent, but has not been processed yet. the job is still executing. Success indicates the job finished successfully, with all steps completed for all stations. Cancelled indicates the job was canceled before it completed, and is no longer running. Failed indicates at least one step failed in one station. The job is no longer running. Each row in the table ends with a details link (>>) and a dispose button (X). Clicking this button changes the view to the Niagara Network Job view or the Batch Job Step Log File view. These views show all logged messages that are related to this single job.

Property	Value	Description
		The overall status for the job in other stations may be different.
Progress	percentage	Shows the percentage of progress toward completing the job.
Start Time	read only (YYMMDD_HHMM timezone)	Displays the start time of the job.
Heartbeat Time	read only (YYMMDD_HHMM timezone)	Displays the time of the last indication that the job is alive.
End Time	read only (YYMMDD_HHMM timezone)	Displays the end time of the job.

Parent topic: [Components in backup module](#)

The baja (Building Automation Java Architecture) module contains a set of classes and APIs that facilitate the framework.

These modules include:

- AuthenticationService
- Category
- CategoryService
- Component
- DataFile
- Directory
- FileSystem
- Folder
- Format
- IpHost
- Job
- JobService
- LocalHost
- Module
- ModuleSpace
- Permissions

- PermissionsMap
- PxView
- ServiceContainer
- Spy
- Station
- StationSaveJob
- UnrestrictedFolder
- User
- UserPasswordConfiguration
- UserPrototypes
- UserService
- UserServicePasswordConfiguration
- Vector
- VirtualComponent
- VirtualGateway
- WsTextBlock
- ZipFile
- [Admin Role \(baja-AdminRole\)](#)
This component provides a single property for configuring viewable hierarchies.
- [Authentication Service \(baja-AuthenticationService\)](#)
This component manages how users verify their identity to the station, using authentication schemes. Some schemes require password configuration, others do not. The **AuthenticationService** node is located in the **Services** container.
- [Authentication Scheme Folder \(baja-AuthenticationSchemeFolder\)](#)
This component is a special container designed to store authentication schemes used in the station **UserService**. Additional authentication schemes and authentication scheme folders may be added. This component is located in the **baja** palette.
- [Category \(baja-Category\)](#)
This component defines a user category. Categories play an integral role in station security, where you can give users permissions for some (or all) categories.
- [Category Service \(baja-CategoryService\)](#)
This service is the station container for all categories, which represent logical groupings to which you can assign components, files, and histories. It is located in a station's **Services** container.
- [Client Credentials \(baja-ClientCredentials\)](#)
This component sets up a Username and Password required to access a client station.

- **[Component \(baja-Component\)](#)**
This component is the required base class for all Baja component classes. It is available in the **baja** module.
- **[baja-DataFile](#)**
DataFile represents a data file in the file system of a session.
- **[baja-DebugService](#)**
- **[baja-Directory](#)**
Directory is used to represent directories in File space implementations.
- **[baja-FileSystem](#)**
FileSystem is a File space for the local machine's file system.
- **[baja-Folder](#)**
This component is a special container designed to store components.
- **[baja-Format](#)**
Format (or “BFormat”) is used to format objects into strings using a standardized formatting pattern language. The format string is normal text with embedded scripts denoted by the % percent character (use %% to insert a real %). A script is one or more calls chained together using the . dot operator. Calls are mapped to methods using reflections. Given call foo, the order of reflection mapping are:
- **[Digest Authentication Scheme \(baja-DigestAuthenticationScheme\)](#)**
This component is a container for properties that configure the DigestScheme. This authentication scheme uses SCRAM-SHA256 (Salted Challenge Response Authentication Mechanism). This is one of the default schemes and this component is located in the **baja** palette. When using the DigestScheme, the password is never sent across the network. Instead, the client sends proof that they know the password.
- **[Global Password Configuration \(baja-GlobalPasswordConfiguration\)](#)**
These properties configure password requirements for a particular authentication scheme. You access them by expanding **Station > Config > Services > AuthenticationService > Authentication Schemes** and double-clicking one of the schemes.
- **[HTTP Basic Authentication Scheme \(baja-HTTPBasicAuthenticationScheme\)](#)**
This authentication scheme performs HTTP-Basic authentication using standard HTTP headers. It only works via the web, and is intended for clients that cannot use cookies. In this authentication scheme, the user name and password are sent over the connection. This component is located in the **baja** palette.
- **[baja-IpHost](#)**
IpHost is used to represent a host machine that is identified with an IP address. The hostname of an IpHost is either a a name resolvable via DNS or is a raw IP address.
- **[baja-Job](#)**
A Job is used to manage a task that runs asynchronously in the background, but requires user visibility. Some example jobs include:
- **[baja-JobService](#)**
- **[Legacy Digest Authentication Scheme \(baja-LegacyDigestAuthenticationScheme\)](#)**
This default authentication scheme provides backward compatibility with stations running a previous software version. This component is located in the **baja** palette.
- **[baja-LocalHost](#)**
LocalHost represents the root of the Baja local Host namespace. The LocalHost is available in the **baja** Module.
- **[baja-Module](#)**
Module encapsulates a Baja software module which is packaged and delivered as a JAR file with a meta-inf/module.xml description. Modules are the basic unit of software deployment in the Baja architecture. Module names must be one to 25 ASCII characters in length and globally unique. Modules are available in the **Workbench ModuleSpace** (named **My Modules** under **My Host**).
- **[baja-ModuleSpace](#)**
ModuleSpace is the container for modules, which are keyed by their module name. In **Workbench** this is **My Modules** under **My Host**.

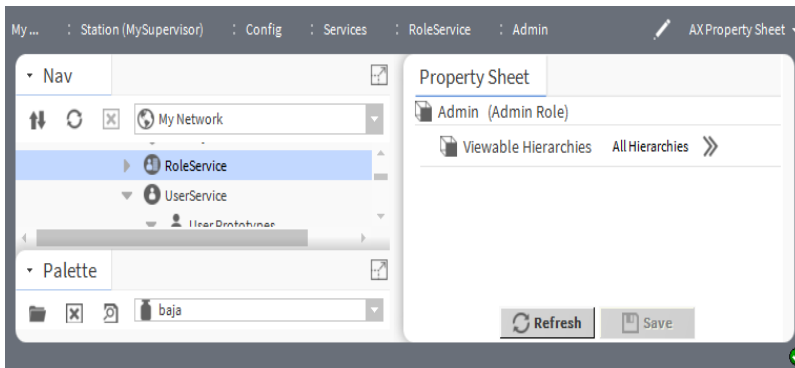
- **User Password Configuration (baja-UserPasswordConfiguration)**
This component is a child container under each User component (and **UserPrototype** component).
- **baja-Permissions**
Permissions are a property used to define the permissions given to a user's PermissionsMap.
- **baja-PermissionsMap**
This component defines the security permissions to grant to a user.
- **User Prototype (baja-UserPrototype)**
This component is an alternative to the legacy default prototype. For remote authentication schemes, it provides better control over user creation and where user properties come from.
- **User Prototype Property (baja-UserPrototypeProperty)**
This component supports the user prototype properties.
- **User Prototypes (baja-UserPrototypes)**
This frozen container on a station's **UserService** contains a single frozen **Default Prototype** user as well as any additional prototypes to support centralized users in the station's **NiagaraNetwork**.
- **Prototype Merge Policy (baja-UserPrototypeMergePolicy)**
As of Niagara 4.12, you can optionally configure SAML, LDAP and Kerberos authentication schemes to merge user prototypes that match a user's prototype attributes, instead of just using a single prototype. If you enable the **Prototype Merge Policy**, user prototypes will be merged according to the policy's configuration when you create remote users upon login. You benefit from this feature, for example, when an organization has multiple user groups for each building and it is not feasible to create additional groups for each possible combination.
- **baja-PxView**
PxView a dynamic view which may be added to components as a property or by overriding getAgents. PxViews store the view contents in an XML file with a px extension. The view itself is defined as a tree of `bajoui:Widgets`. A Px view is a custom graphical view of control logic that you define in a Px file. The view provides a visualization of information in a dynamic graphical format. The purpose of a Px view is to provide a visual representation of information in a rich, dynamic format that is easy to create and to edit.
- **RoleService (baja-RoleService)**
This service manages the **Role Manager** view, which is used to set up user roles in the system.
- **Server Port (baja-ServerPort)**
This is a frozen slot on the NiagaraStation component's **FoxService** used to configure the port number used for secure (https) and nonsecure (http) station connections. The component is available in the **fox** palette.
- **Service Container (baja-ServiceContainer)**
ServiceContainer (**Services**) is the container used, by convention, to store a station's services. The **Service Manager** is its primary view. A ServiceContainer is included in any station created using the **New Station** tool.
- **SMA Notification Settings (baja-SMANotificationSettings)**
This component configures the Software Maintenance Agreement (SMA) expiration reminder. The *Niagara Platform Guide* documents when the the expiration reminder appears.
- **baja-Spy**
This component is an object wrapper for an instance of Spy, with an available SpyViewer interface to view diagnostic information about the running station.
- **SSO Configuration (baja-SSOConfiguration)**
This component is a frozen slot on the **AuthenticationService**, used to configure Single Sign On (SSO) properties for the station. These properties allow you to enable different aspects of SSO functionality such as whether or not to automatically attempt single sign on when users log on to the station. This component is located in the **baja** palette.

- [baja-Station](#)
This component represents a station in the framework.
- [baja-StationSaveJob](#)
This component appears as a child of the **Job Service** and displays the following properties relative to the save job:
- [baja-SyntheticModuleFile](#)
SyntheticModuleFile (synthetic module) is a synthetic Java archive (.jar) that allows the creation of memory-resident modules and types programmatically at station runtime. Synthetic modules differ from standard .jar (non-synthetic) modules in a number of ways, and have a special default **Synthetic Module File View** for editing contents.
- [baja-UnrestrictedFolder](#)
UnrestrictedFolder is a special container designed to store objects for use on a palette. Normal isParentLegal checks are not applied to UnrestrictedFolders. The **UnrestrictedFolder** is available in the **baja** palette.
- [User \(baja-User\)](#)
This component represents a station connection, typically a specific type of person who needs to access the system or another system that makes a station-to-station connection (machine-to-machine). Users are children of the station's **UserService > UserPrototypes** container.
- [UserService \(baja-UserService\)](#)
This service manages all system users: human and machine. You access it by right-clicking **UserService** and clicking **Views > Property Sheet**.
- [baja-UserServicePasswordConfiguration](#)
UserServicePasswordConfiguration (Password Configuration) is a child container under the **UserService**.
- [baja-Vector](#)
Vector is a dynamic Struct which allows properties to be added at runtime.
- [baja-VirtualComponent](#)
A VirtualComponent is the Baja base class for implementations of transient (non-persisted) components that reside in the station's virtual component space, as defined by a VirtualGateway.
- [baja-VirtualGateway](#)
A VirtualGateway is the Baja base class for a component that resides under the station's component space (**Config**), and acts as a gateway to the station's virtual component space.
- [baja-WsTextBlock](#)
WsTextBlock (**TextBlock**) is a component you can drop onto a wire sheet (WireSheet) and position to add text notes. Properties include **Text** (to display), **Foreground** and **Background** colors, **Font**, **Border**, and whether the Text Block is directly **Selectable** in the wire sheet (by default, **Selectable** is true). If **Selectable** is false, you must select this component via its node in the Nav tree.
- [baja-ZipFile](#)
ZipFile represents a zip file in the file system of a session.

Parent topic: [Component guides](#)

This component provides a single property for configuring viewable hierarchies.

Figure 1. Admin Role properties



To access this property, expand **Config > Services > RoleService** and double-click **Admin**.

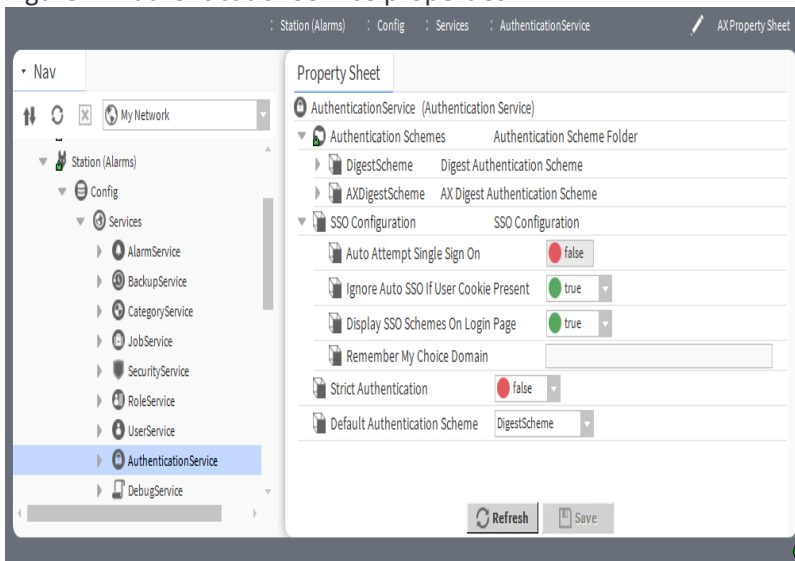
Property	Value	Description
Viewable Hierarchies	All Hierarchies	Configures viewable hierarchies.

Parent topic: [Components in baja module](#)

This component manages how users verify their identity to the station, using authentication schemes. Some schemes require password configuration, others do not. The **AuthenticationService** node is located in the **Services** container.

This component is located in the baja palette.

Figure 1. AuthenticationService properties



To access, expand **Config > Services** and double-click **AuthenticationService**.

Property	Value	Description
Authentication Schemes	additional properties	Identifies the authentication scheme provided by the baja module. The New

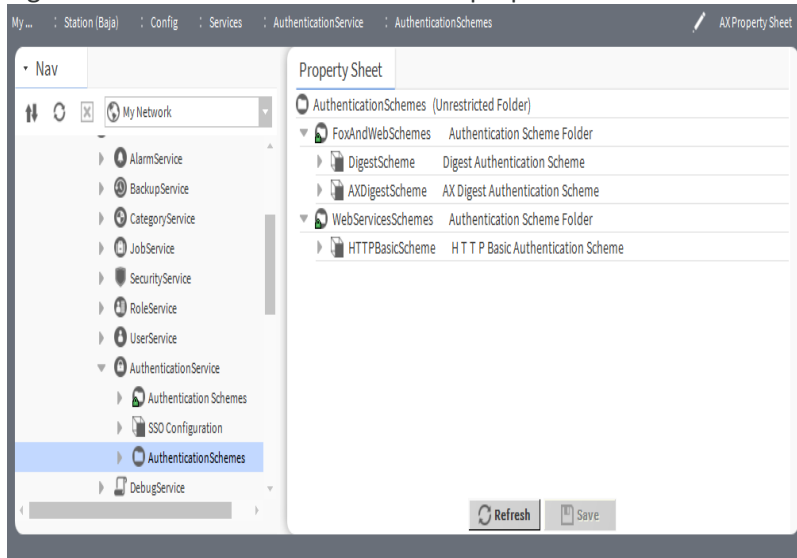
Property	Value	Description
		Station wizard installs two default authentication schemes. DigestScheme provides SCRAM-SHA256 (Salted Challenge Response Authentication Mechanism) technology for connecting framework entities. Several messages are passed back and forth to prove the client knows the password. For property descriptions, refer to “baja-DigestScheme.” AXDigestScheme provides compatibility with stations running a previous software version. For property descriptions, refer to “baja-DigestScheme.” Additional schemes may reside in other palettes. Developers may also create authentication schemes for special circumstances. You pick the one or two schemes you wish to use, drag them from the palette and drop them directly under the AuthenticationService in the Nav tree. A topic for each individual scheme documents the properties for each.
SSO Configuration	additional properties	Enables aspects of SSO (Single Sign-On) functionality, such as whether or not to automatically attempt single sign on when users log on to a station. “SSO Configuration (baja-SSOConfiguration)” documents the additional properties.
Strict Authentication	true or false (default)	Enforces authentication controls.
Default Authentication Scheme	drop-down list	Selects the default authentication scheme to use.

Parent topic: [Components in baja module](#)

This component is a special container designed to store authentication schemes used in the station **UserService**. Additional authentication schemes and authentication scheme folders may be added. This component is located in the baja palette.

The AuthenticationSchemes authentication scheme folder is a frozen slot on the **AuthenticationService**, which contains the following default authentication scheme folders and schemes.

Figure 1. AuthenticationSchemeFolder properties



To access, expand **Config > Services > AuthenticationService** and right-click **AuthenticationSchemes > Views > AX Property Sheet**.

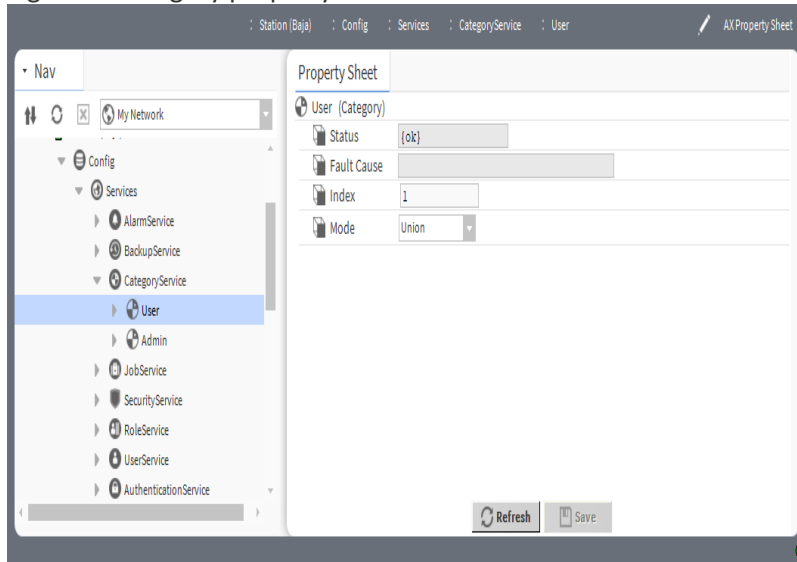
Property	Value	Description
FoxandWebSchemes	additional properties	Serves as a container for the baja core authentication schemes. A topic for each individual scheme documents the properties for each.
WebServiceSchemes	additional properties	Serves as a container for web authentication schemes. For property descriptions, refer to "baja-LegacyDigestAuthenticationScheme." Allows to configure HTTPBasicScheme which performs HTTP-Basic authentication using standard HTTP headers. It only works via the web, and is intended for clients that cannot use cookies. This authentication scheme sends the user name and password over the connection.

Parent topic: [Components in baja module](#)

This component defines a user category. Categories play an integral role in station security, where you can give users permissions for some (or all) categories.

Each Category represents a custom logical grouping and has a unique category index number. You can assign components, files, and histories to one or more Categories. All categorizable components have a **Category Sheet** view, which displays that the component's stored Category memberships (CategoryMask). Categories reside under the station's **CategoryService**, which has the **Category Browser** and **Category Manager** views.

Figure 1. Category property



To access, expand **Config > Services > CategoryService** and double-click **User** or **Admin**.

In addition to the standard properties (Status and Fault Cause), this component provides these properties.

Property	Value	Description
Index	number	A unique number for the category, as it is known to the station.
Mode	drop-down list (Union or Intersection)	Defines how a Category is mapped into user permissions for the component. Union (default): Indicates that permissions for the Category are added to the user's permissions for the component. Intersection: Indicates that missing permissions for the Category are removed from the user's permissions for the component.

Example: *Component A* is in *Category 1*, but not *Category 2*. User 'X' has access to *Category 1* and *Category 2*.

- Union mode:

User 'X' has permission to *Component A* because the OR logic between *Category 1* (where *Component A* is included) and *Category 2* (where *Component A* is not included) will cause *Component A* to be added due to its membership in *Category 1*. *Component A* is accessible by User 'X' if it is included in either *Category 1* OR *Category 2*.

- Intersection mode: User 'X' does not have permission to *Component A* because the AND logic between *Category 1* (where *Component A* is included) and *Category 2* (where *Component A* is not included) will cause *Component A* to be removed due to its exclusion

from Category 2. For User 'X' to access *Component A*, the configuration must be changed to include *Component A* in both *Category 1* AND *Category 2*.

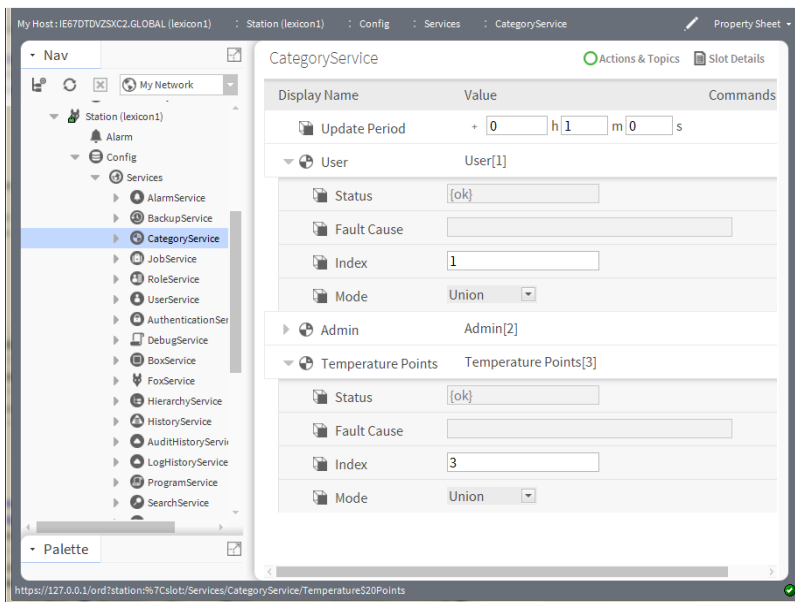
Parent topic: [Components in baja module](#)

This service is the station container for all categories, which represent logical groupings to which you can assign components, files, and histories. It is located in a station's **Services** container.

The default view of this service, the **Category Browser**, lets you centrally assign different objects to categories, using an expandable tree view of the station. The CategoryService also provides a **Category Manager** view, for you to create, edit and delete categories. Categories play an integral role in station security, where you can give users permissions for some (or all) categories. By default, the CategoryService is included when you create a new station using the **New Station** wizard.

Primary properties

Figure 1. CategoryService properties



In addition to being the container for child categories, the CategoryService has only one slot (property): Update Period.

Property	Value	Description
Update Period	hours minutes seconds	Sets the interval at which the system automatically assigns ancestor permissions. The default value is one (1) minute. If you assign a zero value, the system disables this feature.

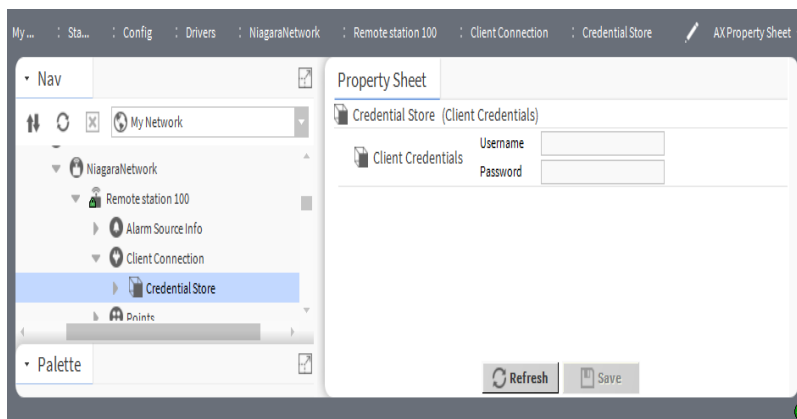
User, Admin and additional basic category properties

Property	Value	Description
Status	read-only	Reports the condition of the entity or process at last polling. {ok} indicates that the component is licensed and polling successfully. {down} indicates that the last check was unsuccessful, perhaps because of an incorrect property, or possibly loss of network connection. {disabled} indicates that the Enable property is set to false. {fault} indicates another problem. Refer to Fault Cause for more information.
Mode	drop-down list (Union or Intersection)	Defines how a category is mapped into user permissions for the component. Union (default): Indicates that permissions for the Category are added to the user's permissions for the component. Intersection: Indicates that missing permissions for the Category are removed from the user's permissions for the component.
Fault Cause	read-only	Indicates the reason why a system object (network, device, component, extension, etc.) is not working (in fault). This property is empty unless a fault exists.
Index	integer	Sequential number that identifies the property in the station.

Parent topic: [Components in baja module](#)

This component sets up a Username and Password required to access a client station.

Figure 1. Client Credentials properties



To access these properties, expand **Config > Drivers > NiagaraNetwork > NiagaraStation > ClientConnection** and double-click **Credential Store**.

Property	Value	Description
Username	text	Defines a username.
Password	text	Defines a password for username.

Parent topic: [Components in baja module](#)

This component is the required base class for all Baja component classes. It is available in the baja module.

Containers allow you to logically group components. The current container is the component that contains components in the display window. A container may be selected as the current container by one of the following methods:

- Double-click the component in the Nav tree.
- Right-click the component in the Nav tree (which brings up a menu) and select a view.
- Right-click the component in a wire sheet (which brings up a menu) and select a view.

Container components include the following:

- Component can be used as a general container for components. It allows you to place components and links in a container.
- Page is a special component used to created a map of name/Ref pairs as dynamic slots.

Parent topic: [Components in baja module](#)

DataFile represents a data file in the file system of a session.

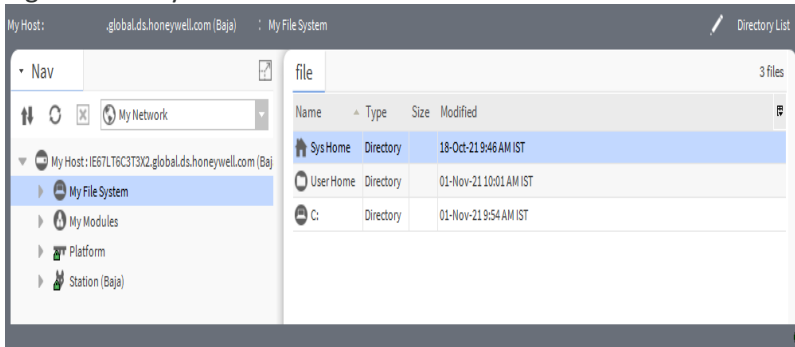
Parent topic: [Components in baja module](#)

Directory is used to represent directories in File space implementations.

Parent topic: [Components in baja module](#)

FileSystem is a File space for the local machine's file system.

Figure 1. FileSystem



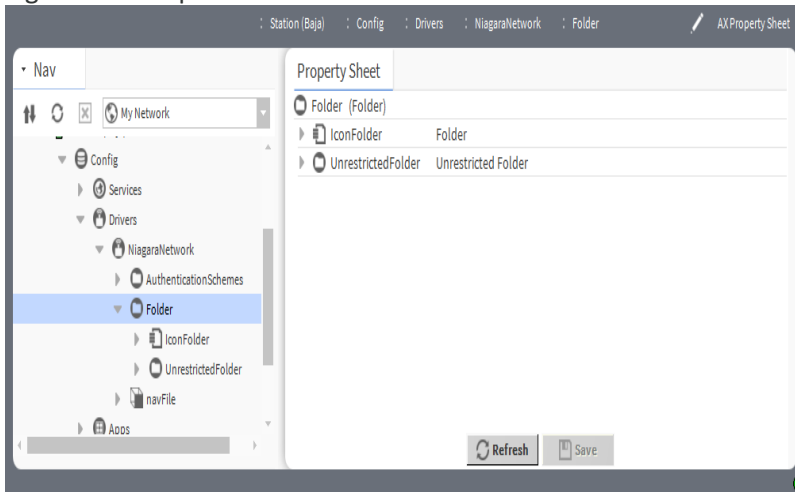
This component helps to organise the files in the local system.

Parent topic: [Components in baja module](#)

This component is a special container designed to store components.

This **Folder** is available in the baja palette. It is used in multiple places within the framework.

Figure 1. Example of this folder



To access, expand **Config > Drivers > NiagaraNetwork** and right-click **Folder > Views > AX Property Sheet**.

This component contains a folder for various components.

Parent topic: [Components in baja module](#)

Format (or “BFormat”) is used to format objects into strings using a standardized formatting pattern language. The format string is normal text with embedded scripts denoted by the % percent character (use %% to insert a real %). A script is one or more calls chained together using the . dot operator. Calls are mapped to methods using reflections. Given call foo, the order of reflection mapping are:

- special call

- `getFoo(Context)`
- `getFoo`
- `foo(Context)`
- `foo`
- `get("foo")`

The following special functions are available to use in a script:

- `time` calls `Clock.time` to get current time as an `AbsTime`
- `lexicon(module:key)` gets the specified lexicon text

Examples of formats:

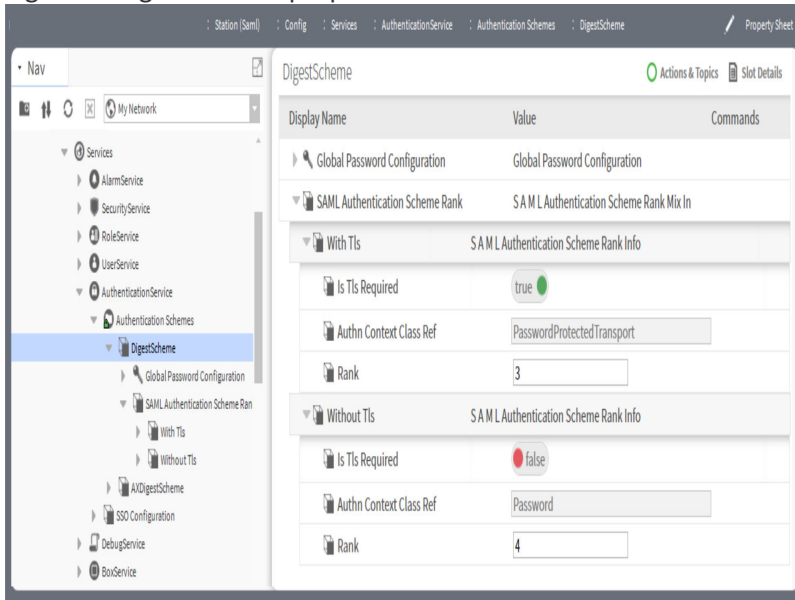
- `hello world`
- `my name is %displayName%`
- `my parent's name is %parent.displayName%`
- `%value% {%status.flagsToString%} @ %status.priority%`
- `%time.toDateString%`
- `%lexicon(bajaui:dialog.error)%`

Parent topic: [Components in baja module](#)

This component is a container for properties that configure the `DigestScheme`. This authentication scheme uses SCRAM-SHA256 (Salted Challenge Response Authentication Mechanism). This is one of the default schemes and this component is located in the `baja` palette. When using the `DigestScheme`, the password is never sent across the network. Instead, the client sends proof that they know the password.

When a supervisor has a **SAMLIdPService** installed in the **Services** container (without this, the `Mixin` and authentication scheme rank will not appear) all authentication schemes have a `Mixin` added to them, the **SAML Authentication Scheme Rank** `Mixin`. The `Mixin` has two frozen properties, **With TLS** and **Without TLS**. These properties allow the authentication schemes to have a numerical ranking relative to one another. A lower number indicates a better security ranking.

Figure 1. DigestScheme properties



To access, expand **Config > Services > AuthenticationService**. Select **AuthenticationSchemes**, right-click and choose the **Property Sheet** view.

Property	Value	Description
Global Password Configuration	additional properties	Configures password requirements for a particular authentication scheme. "Global Password Configuration (baja-GlobalPasswordConfiguration)" documents the additional properties.
SAML Authentication Scheme Rank	additional properties	Represents the Mixin, for specifying authentication scheme ranking.
SAML Authentication Scheme Rank – With Tls and Without Tls	additional properties	Represents the rank and AuthnContextClassRef of the scheme, when used over With Tls and Without Tls . - Is TLS Required: Indicates whether the rank applies to the scheme used over TLS or not. - Authn Context Class Ref: The SAML Authn Context Class Ref corresponding to this authentication type. For core Niagara authentication schemes, the property is read-only. For third party authentication schemes, by default the property is set to unspecified. To edit this property the values should be taken from the SAML specification.

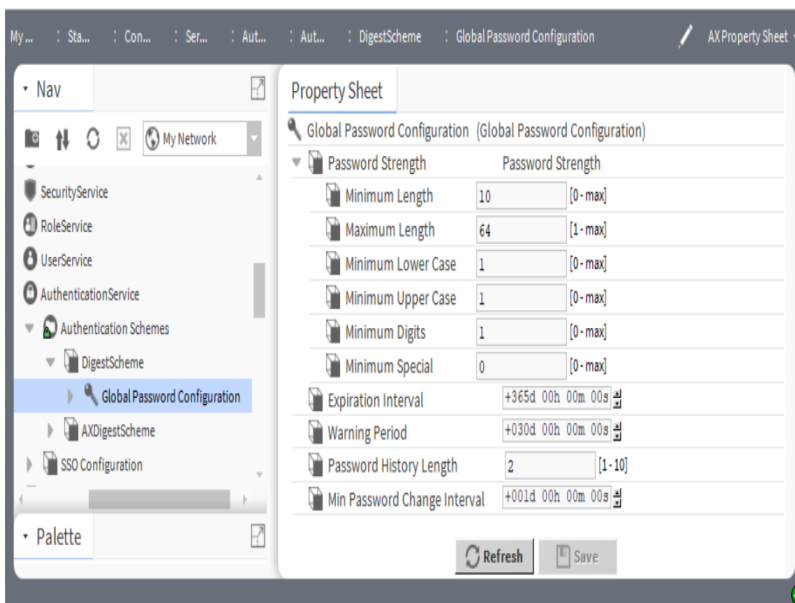
Property	Value	Description
		- Rank: Indicates a relative numerical ranking, where a lower number indicates a better security level ranking than a higher number. For core Niagara authentication schemes, a default rank is set but it can be changed. For Third party authentication schemes the defaults rank is 5 and can also be changed. The default ranking for core Niagara authentication schemes is: – 1: Google Authentication Scheme (TLS, no TLS) – 2: Kerberos (TLS, no TLS) – 3: SSL/TLS Certificate, Digest (TLS), AX Digest (TLS) – 4: Digest (no TLS), AX Digest (no TLS), LDAP (TLS) – 5: LDAP (no TLS)

Parent topic: [Components in baja module](#)

These properties configure password requirements for a particular authentication scheme. You access them by expanding **Station > Config > Services > AuthenticationService > Authentication Schemes** and double-clicking one of the schemes.

Scheme properties

Figure 1. Global Password Configuration properties

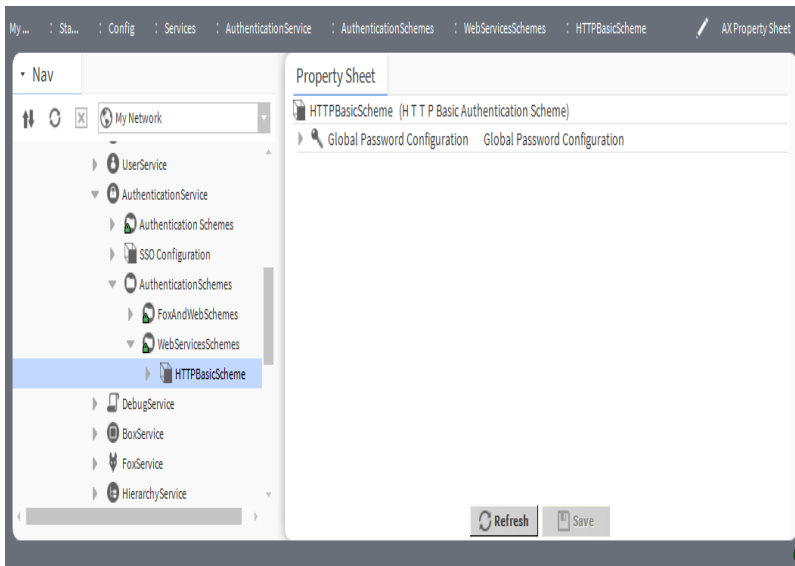


To access, expand **Config > Services > AuthenticationService > AuthenticationSchemes > DigestScheme** and double-click **Global Password Configuration**.

Property	Value	Description
Password Strength	several sub-properties	Configures password requirements for the particular authentication scheme.
Minimum Length	number (defaults to 10)	Indicates the total number of characters required.
Maximum Length	number (defaults to 64)	Indicates the maximum number of characters allowed. The default of 64 is recommended.
Minimum Lower Case	number (defaults to 1)	Indicates the minimum number of lower case letters required.
Minimum Upper Case	number (defaults to 1)	Indicates minimum number of upper case letters required.
Minimum Digits	number (defaults to 1)	Indicates the minimum number of digits.
Minimum Special	number (defaults to 0)	Indicates the number of special characters required. For example: ! @ # \$ % ^ , . ; etc.
Expiration Interval	number of days, hours, minutes and seconds	Defines the length of time from when the password is created until it is no longer valid. When this period of time expires, the system denies access.
Warning Period	number of days, hours, minutes and seconds	Defines how many days of warning a user receives prior to the expiration of the password.
Password History Length	number	Defines how many previously used passwords the system remembers.
Min Password Change Interval	number of days, hours, minutes and seconds	Defines the minimum time interval until when a user has to change the password.

Parent topic: [Components in baja module](#)

This authentication scheme performs HTTP-Basic authentication using standard HTTP headers. It only works via the web, and is intended for clients that cannot use cookies. In this authentication scheme, the user name and password are sent over the connection. This component is located in the baja palette.



To access, expand **Config > Services > AuthenticationService > AuthenticationSchemes > WebServicesSchemes** and double-click **HTTPBasicScheme**.

Property	Value	Description
Global Password Configuration	additional properties	Configures password requirements for a particular authentication scheme. "Global Password Configuration (baja-GlobalPasswordConfiguration)" documents the additional properties.

Parent topic: [Components in baja module](#)

IpHost is used to represent a host machine that is identified with an IP address. The hostname of an IpHost is either a a name resolvable via DNS or is a raw IP address.

- A blue square indicates active connection(s) from Workbench, for example, Fox (station) or platform.
- A red square indicates no active connections from Workbench.

Parent topic: [Components in baja module](#)

A Job is used to manage a task that runs asynchronously in the background, but requires user visibility. Some example jobs include:

- **StationSaveJob** — From a station save, either initiated manually or from the auto-save function (see the *Niagara Platform Guide*).
- **FoxBackupJob** — From a station backup (dist) to a remote PC.

Many drivers also have various job types too. For example, the NiagaraDriver includes a **StationDiscoveryJob** and **NiagaraScheduleLearnJob**.

Every job finishes displaying one of the following status descriptors:

- **Success** — Job completed successfully.
- **Canceled** — Job canceled by user.
- **Failure** — Job failed to complete.
- **Completed** — Job completed.

Also, if you have the station open in Workbench, you see a momentary notification window in the lower-right corner of your display.

You can monitor and cancel a job from within the particular view where you initiated it, or centrally from the **Job Service Manager** view of a station's **JobService**. You can also open a Jobs side bar to see all jobs in all opened stations.

Regardless of how you access jobs, note the following:

- To see details on a job, click the button next to its status descriptor. A popup **Job Log** window displays all the interim steps about the job, including timestamps and relevant messages.
- To dispose of a job, click the close (X) button to remove it from the station.

Note: All jobs in a station are cleared upon a station restart.

Parent topic: [Components in baja module](#)

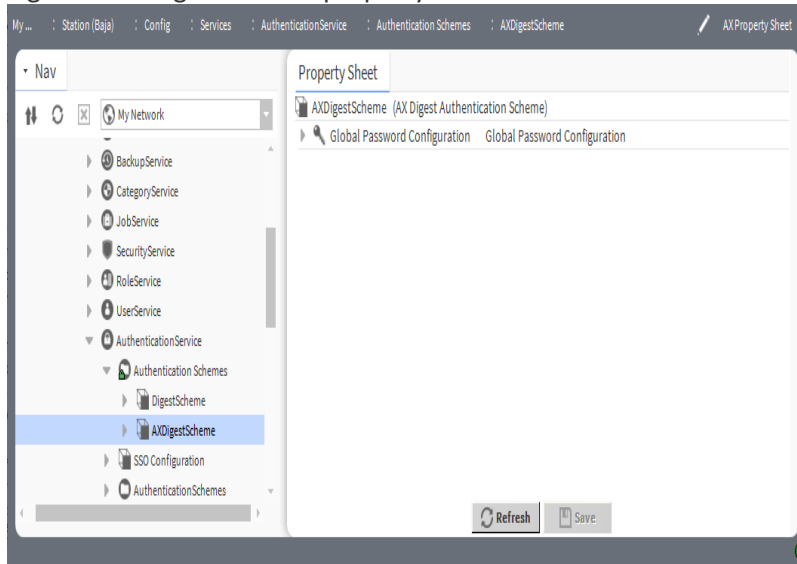
This default authentication scheme provides backward compatibility with stations running a previous software version. This component is located in the baja palette.

This authentication scheme provides compatibility with these : NiagaraAX versions:

- 3.5u4
- 3.6u4 and up
- 3.7u1 and up
- any 3.8 version

Earlier versions of NiagaraAX do not support the AXDigestScheme.

Figure 1. AXDigestScheme property



To access, expand **Config > Services > AuthenticationService > AuthenticationSchemes** and double-click **AXDigestScheme**.

Property	Value	Description
Global Password Configuration	additional properties	Configures password requirements for a particular authentication scheme. "Global Password Configuration (baja-GlobalPasswordConfiguration)" documents the additional properties.

Parent topic: [Components in baja module](#)

LocalHost represents the root of the Baja local Host namespace. The LocalHost is available in the baja Module.

Parent topic: [Components in baja module](#)

Module encapsulates a Baja software module which is packaged and delivered as a JAR file with a meta-inf/module.xml description. Modules are the basic unit of software deployment in the Baja architecture. Module names must be one to 25 ASCII characters in length and globally unique. Modules are available in the Workbench ModuleSpace (named **My Modules** under **My Host**).

Parent topic: [Components in baja module](#)

ModuleSpace is the container for modules, which are keyed by their module name. In Workbench this is **My Modules** under **My Host**.

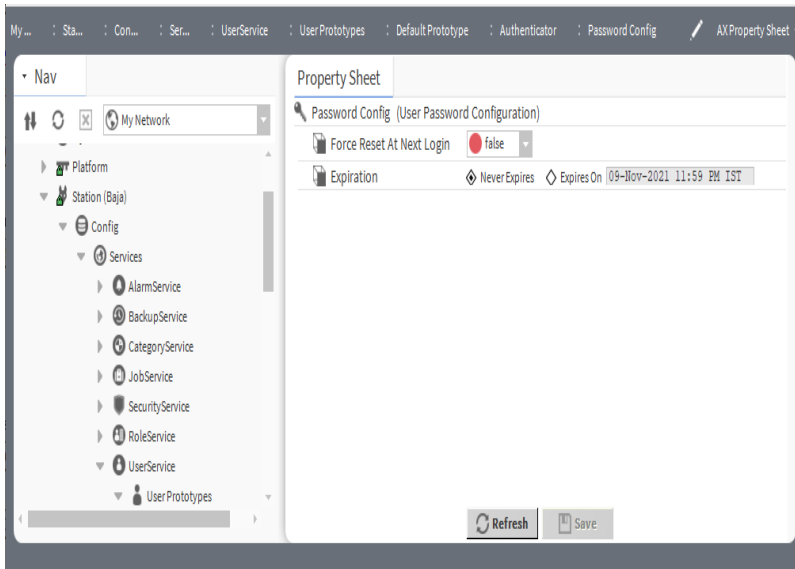
Parent topic: [Components in baja module](#)

This component is a child container under each User component (and **UserPrototype** component).

It contains properties to specify periodic password expiration for the user and to require a password change (reset) upon the user's next station login. The station's **UserService** also has a related Password Configuration child container.

Note: These components are not available in a new station until it is started and saved.

Figure 1. User Password Configuration properties

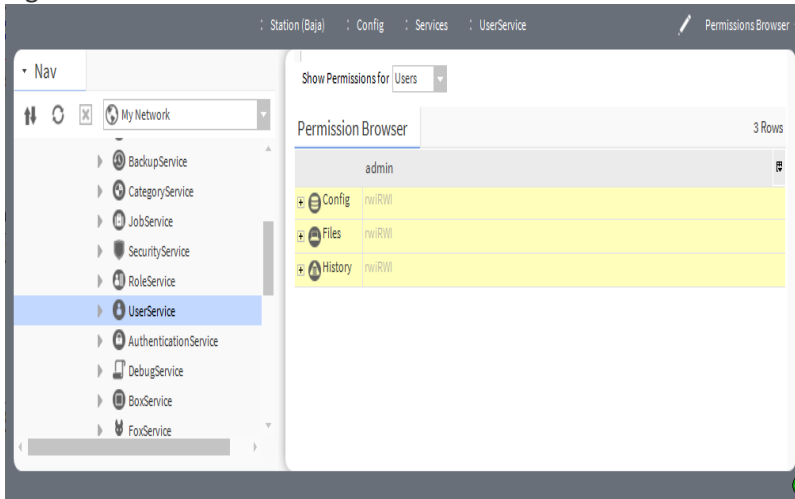


Property	Value	Description
Force Reset At Next Login	true false (default)	If set to false allow the user to continue using the same password. If set to true allows user to change the password on next login.
Expiration	check box	Expires On allows to set the password expiration date of the password. Never Expires sets the password permanently.

Parent topic: [Components in baja module](#)

Permissions are a property used to define the permissions given to a user's PermissionsMap.

Figure 1. Permissions Browser view



To access, expand **Config > Services** and right-click > **UserService > Views > Permissions Browser**.

Use the **Show Permissions for** drop-down list to switch between permissions for **Users**, and for **Roles**. When viewing permissions for **Users**, the view displays a separate column for each user as well as any prototype.

Parent topic: [Components in baja module](#)

This component defines the security permissions to grant to a user.

Permissions are added as dynamic properties with the name matching a Category and the value an instance of permissions. For details, see the *Niagara Station Security Guide*

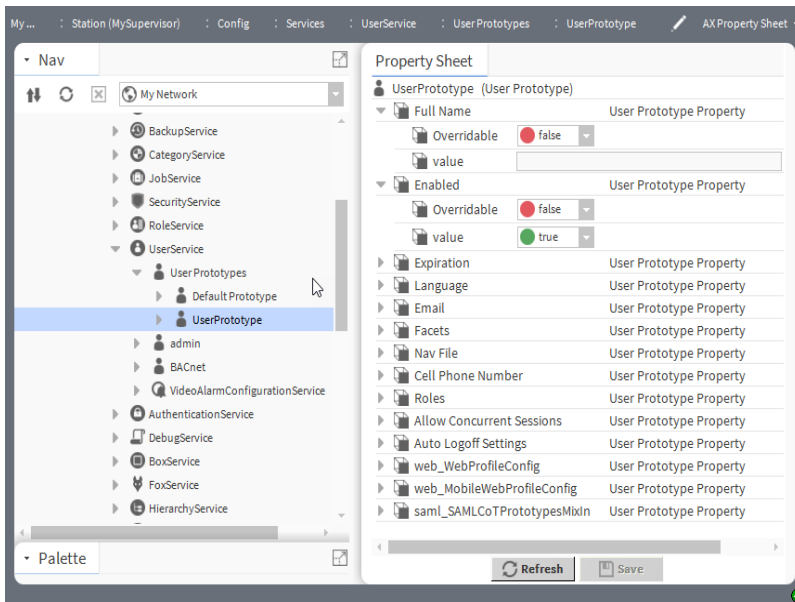
Parent topic: [Components in baja module](#)

This component is an alternative to the legacy default prototype. For remote authentication schemes, it provides better control over user creation and where user properties come from.

Note: The SAML Authentication Scheme only supports the baja-UserPrototype. While LDAP and Kerberos support this user prototype as well as the default user prototype.

Although the properties are similar to those of the default prototype, **UserPrototype** has only the properties that support SAML authentication. Also, it has an **Overridable** user property that, when set to true, prevents a value from being overwritten with a default value at next login.

Figure 1. UserPrototype properties



To use this component, expand **Config > Services > UserService > UserPrototypes**, open the baja palette and drag a **UserPrototype** component to the **UserPrototypes** folder.

To view these property groups in a station, expand **Config > Services > UserService > UserPrototypes** and double-click a **UserPrototype** component.

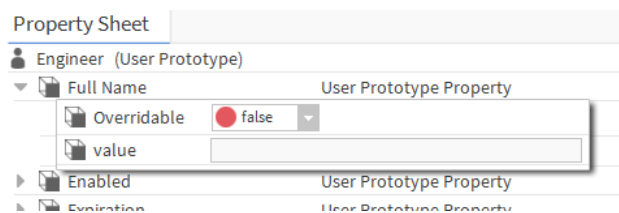
The topic “User Prototype Property (baja-UserPrototypeProperty)” documents the individual sub-properties of a user prototype.

Parent topic: [Components in baja module](#)

This component supports the user prototype properties.

Each sub-property on the **User Prototype Property Sheet** has an Overridable toggle and a value.

Figure 1. Full Name user prototype properties



To view these properties, expand **Config > Services > UserService > UserPrototypes**, double-click a **UserPrototype** component and expand the property group.

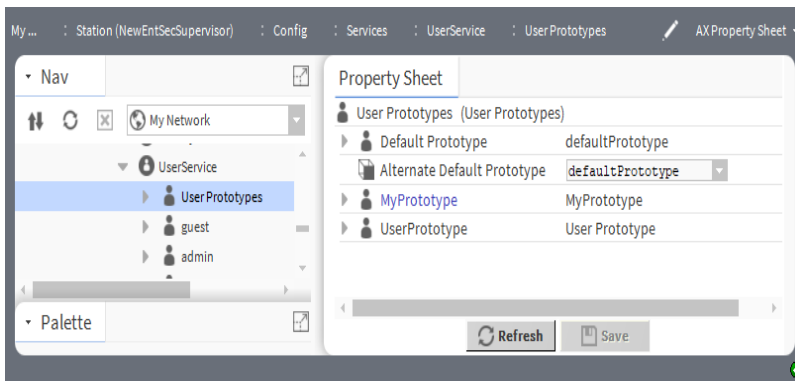
Property	Value	Description
Overridable	true, false (default)	Determines if the property can be manually overridden (true) or not (false) for a user that was created from this prototype.
Full Name value	string	Defines the user name.
Enabled value	true (default), false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Expiration value, Never Expires	radio button	Configures the prototype to never expire.
Expiration value, Expires On	date and time	Configures an expiration date and time for the prototype.
Language value	two-digit field	Selects the prototype language.
Email value	text	Defines the user's email address.
Facets value, Time Format	drop-down list (defaults to (default))	Selects the preferred time format for the prototype including the date format.
Facets value, Unit Conversion	drop-down list (defaults to None)	Selects the unit of measure for the prototype: Metric or English.
Nav File value	text (defaults to null)	Provides the path and filename that contains the navigation tree for the prototype.
Cell Phone Number value	phone number	Defines a mobile phone number to assign to the prototype.
Roles value	Edit Roles window	Selects the role to assign to this prototype.
Allow Concurrent Sessions value	true (default), false	Enables (true) and disables (false) concurrent station sessions for the prototype.
Auto Logoff Settings value, Auto Logoff Enabled	true (default), false	Enables (true) and disables (false) automatic logoff when Auto Logoff Period expires.
Auto Logoff Settings value, Use Default Auto Logoff Period	true (default), false	Enables (true) and disables (false) use of the default period (15 minutes)
Auto Logoff Settings value, Auto Logoff Period	read-only	Reports the default period.
web_WebProfileConfig, Type	drop-down list	Selects the profile to use. For details on this profile, refer to "User (baja-user)" in <i>Getting Started with Niagara</i> .
web_MobileWebProfileConfig, Mobile Nav File	string	Provides the path and filename that contains the navigation tree for the mobile web profile.
web_MobileWebProfileConfig, Type		Selects the profile to use. For details on this profile, refer to "User (baja-user)" in <i>Getting Started with Niagara</i> .

Property	Value	Description
saml_SAMLCOTPrototypesMixIn, SAML Prototypes	additional properties	Expands to another property. “SAML CoT Prototypes (saml-SAMLCOTPrototypesMixIn” documents the additional properties.

Parent topic: [Components in baja module](#)

This frozen container on a station's **UserService** contains a single frozen **Default Prototype** user as well as any additional prototypes to support centralized users in the station's **NiagaraNetwork**.

Figure 1. Example of User Prototypes (container) properties



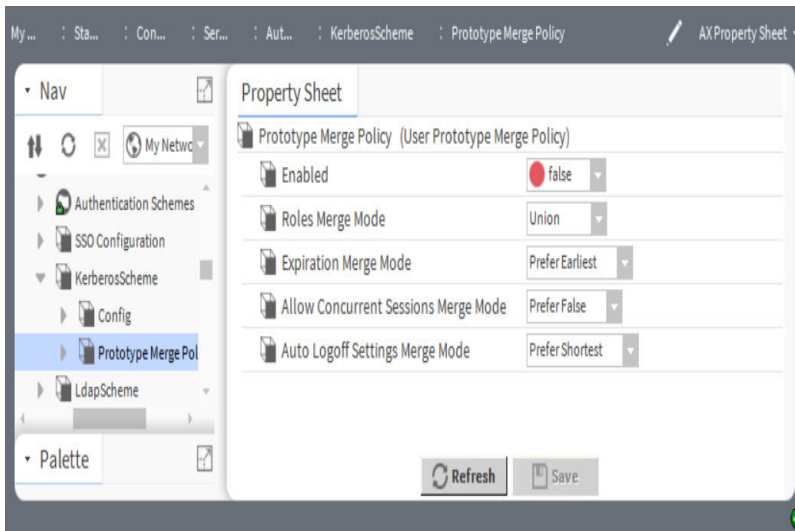
You access this view by expanding **Config > Services > UserService** followed by double-clicking **User Prototypes**.

Property	Value	Description
Default Prototype	additional properties	Defines the properties of the default prototype. “User (baja-User)” documents the additional properties.
Alternate Default Prototype	drop-down list	Provides a list of all user prototypes from which to select a default prototype.

Parent topic: [Components in baja module](#)

As of Niagara 4.12, you can optionally configure SAML, LDAP and Kerberos authentication schemes to merge user prototypes that match a user's prototype attributes, instead of just using a single prototype. If you enable the Prototype Merge Policy, user prototypes will be merged according to the policy's configuration when you create remote users upon login. You benefit from this feature, for example, when an organization has multiple user groups for each building and it is not feasible to create additional groups for each possible combination.

Figure 1. User Prototype Merge Policy



For the KerberosScheme, you access this view by expanding **Config > Services > AuthenticationService > KerberosScheme** and double-clicking **Prototype Merge Policy**.

For the LDAPScheme, you access this view by expanding **Config > Services > AuthenticationService > LdapScheme** and double-clicking **Prototype Merge Policy**.

For the SAMLAuthenticationScheme, you access this view by expanding **Config > Services > AuthenticationService > SAMLAuthenticationScheme** and double-clicking **Prototype Merge Policy**.

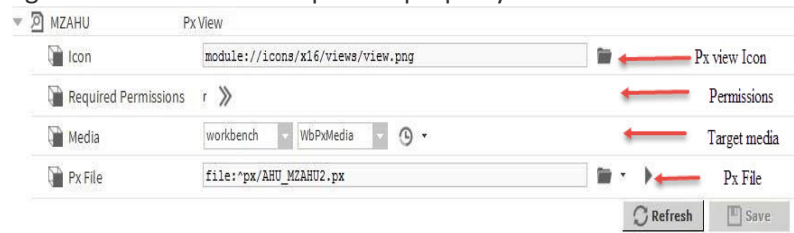
Property	Value	Description
Enabled	true, false (default)	Determines whether or not user prototypes are merged according to your policy configuration.
Roles Merge Mode	drop-down list	Provides two modes: • Union (default): the resulting user has all roles of all user prototypes. • Use first: the value from the first user prototype is used.
Expiration Merge Mode	drop-down list	Provides two modes: • Prefer Earliest (default): the earliest expiration is used. • Use first: the value from the first user prototype is used.
Allow Concurrent Sessions Merge Mode	drop-down list	Provides two modes: • Prefer False (default): if any value is false, false is used.

Property	Value	Description
		Use first: the value from the first user prototype is used.
Auto Logoff Settings Merge Mode	drop-down list	Provides two modes: - Prefer Shortest (default): the shortest auto-logoff setting is used. If there is a tie, the default system setting configured in the UserService is preferred. - Use first: the value from the first user prototype is used.

Parent topic: [Components in baja module](#)

PxView a dynamic view which may be added to components as a property or by overriding getAgents. PxViews store the view contents in an XML file with a px extension. The view itself is defined as a tree of bajaui:Widgets. A Px view is a custom graphical view of control logic that you define in a Px file. The view provides a visualization of information in a dynamic graphical format. The purpose of a Px view is to provide a visual representation of information in a rich, dynamic format that is easy to create and to edit.

Figure 1. Px view in a component property sheet



Looking at an object's Px view in the Property Sheet (where it may be edited) helps illustrate what a Px view is. A Px view is comprised of the following parts

- The Px view icon appears to the left of the Px view display name, in the view selector menu, in the Nav tree side bar and in the Property Sheet view.
- The Px file contains the view's properties.
- Media identifies the type of view.
- Required permissions determine who can open the view.

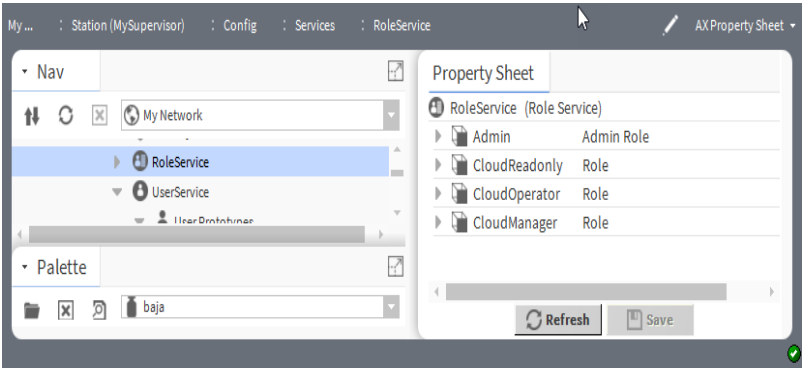
Parent topic: [Components in baja module](#)

This service manages the **Role Manager** view, which is used to set up user roles in the system.

CAUTION: There are risks involved in giving any user broad permissions on the Role Service. For example, giving a user admin write permissions on the Role Service allows that user to

create, edit, rename or delete any role. Best practices recommend that such permissions on the Role Service be limited only to appropriately authorized users.

Figure 1. RoleService properties



This screen captures shows cloud roles. To configure these roles, your station must be licensed for the nCloud driver.

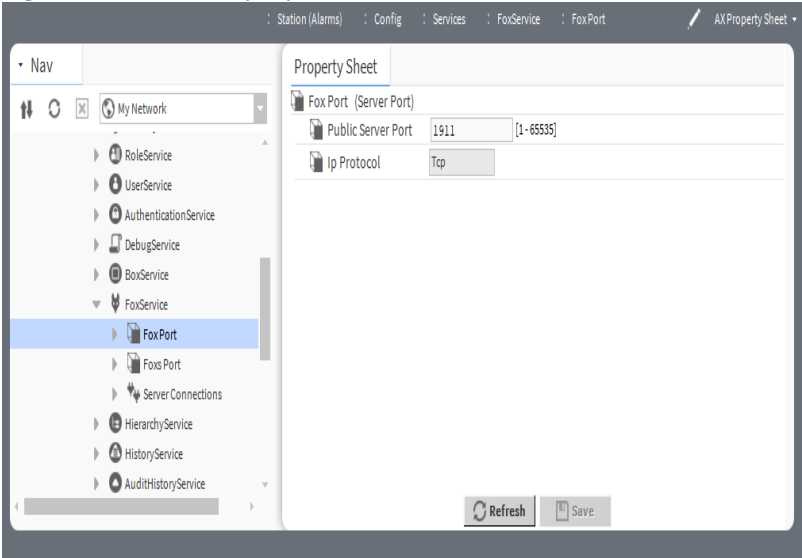
To access these properties, expand **Config > Services**, right-click **RoleService** and click **Views > AX Property Sheet**.

Property	Value	Description
Admin	additional property	Configures the viewable hierarchies for the admin role. "Admin Role (baja-AdminRole)" documents the additional property.

Parent topic: [Components in baja module](#)

This is a frozen slot on the NiagaraStation component's **FoxService** used to configure the port number used for secure (https) and nonsecure (http) station connections. The component is available in the **fox** palette.

Figure 1. ServerPort properties



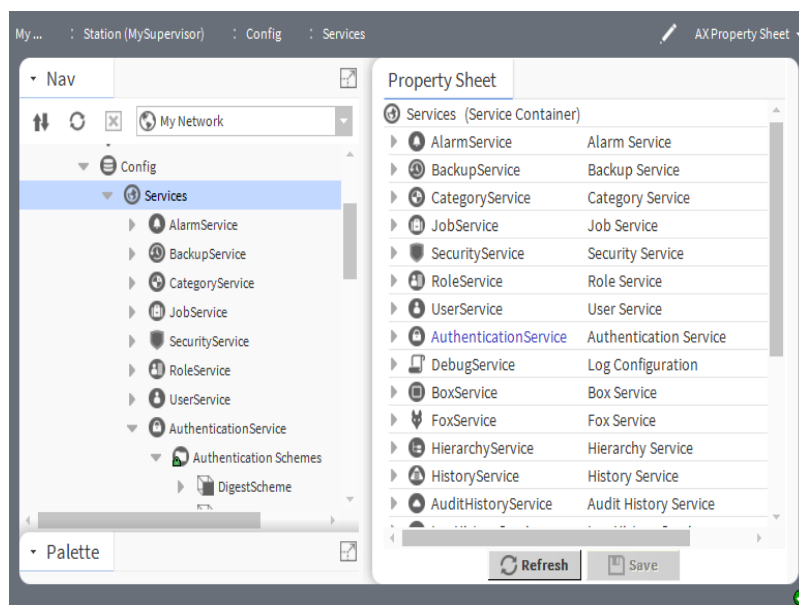
To access, expand **Config > Services > FoxService** and double-click **FoxPort** or **FoxsPort**.

Property	Value	Description
Public Server port	1911 (default)	Specifies the port number for standard Fox communication.
Ip protocol	Tcp (default)	Specifies the protocol.

Parent topic: [Components in baja module](#)

ServiceContainer (**Services**) is the container used, by convention, to store a station’s services. The **Service Manager** is its primary view. A ServiceContainer is included in any station created using the **New Station** tool.

Figure 1. Service Container Property Sheet



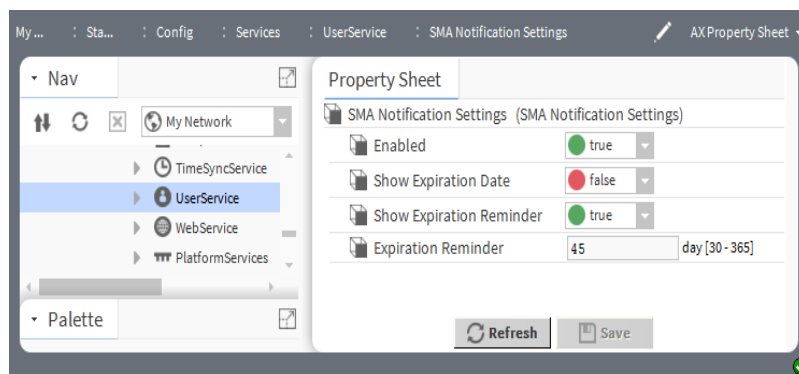
To access, expand **Config**, right-click **Services** and click **Views > AX Property Sheet**.

Each component is documented by its own topic(s).

Parent topic: [Components in baja module](#)

This component configures the Software Maintenance Agreement (SMA) expiration reminder. The *Niagara Platform Guide* documents when the the expiration reminder appears.

Figure 1. SMA expiration reminder properties



Property	Value	Description
Show Expiration Date	true (default), false	Configures SMA notification settings. true displays the initial SMA expiration reminder in the browser-connected station Login window the number days prior to expiry defined by Expiration Reminder.

Property	Value	Description
		false, hides the initial SMA expiration reminder. It does not display. Note: Once the SMA expires, the SMA expiration reminder cannot be hidden.
Expiration Reminder	30–365 days (defaults to 45 days)	Configures the number of days prior to expiry when the message displays.

Parent topic: [Components in baja module](#)

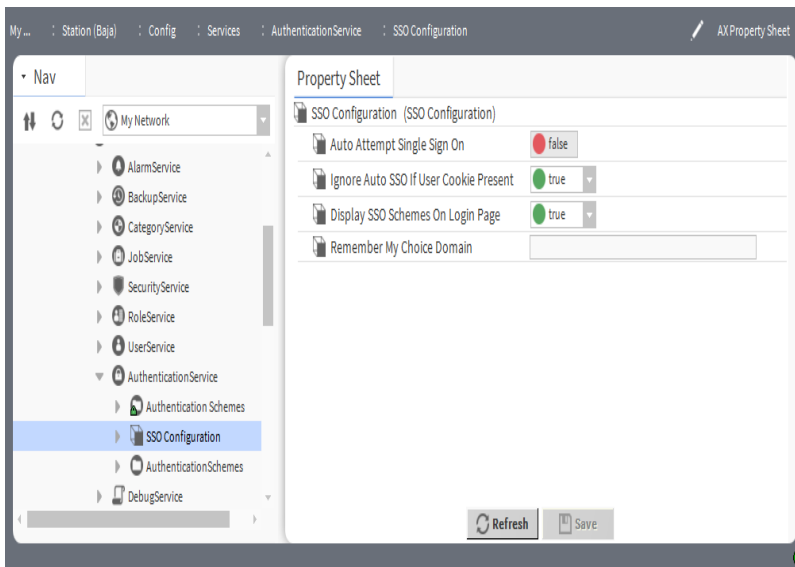
This component is an object wrapper for an instance of Spy, with an available SpyViewer interface to view diagnostic information about the running station.

Parent topic: [Components in baja module](#)

This component is a frozen slot on the **AuthenticationService**, used to configure Single Sign On (SSO) properties for the station. These properties allow you to enable different aspects of SSO functionality such as whether or not to automatically attempt single sign on when users log on to the station. This component is located in the baja palette.

Single Sign On is a method of controlling access to multiple related, but independent software systems. With SSO, a user logs in once and gains access to all networked systems without being prompted to log in again at each of them. Centrally managed credentials eliminate the opportunity for errors and using one point of authentication makes authentication less complicated and more secure.

Figure 1. SSO properties



To access, expand **Config > Services > AuthenticationService** and double-click **SSO Configuration**.

Property	Value	Description
Auto Attempt Single Sign On	true or false (default)	When set to true, SSO is automatically attempted when logging you into the station. That is unless the user specifically visits the login or prelogin pages. Typically, when there is just one SSO scheme available you would set auto-SSO to true. In order to set this to true, there must be exactly one SSO scheme available. When multiple SSO schemes are present in the station this setting is automatically false and read only.
Ignore Auto SSO If User Cookie Present	true (default) or false	When set to true, the presence of the <code>niagara_userid</code> cookie causes the user to always be redirected to the login screen, instead of automatically attempting SSO. When set to false, this has no effect. This is useful if you have certain users who need to login as station users rather than SSO users, such as admin users.
Display SSO Schemes On Login Page	true (default) or false	When set to true, a separate login button for each SSO authentication scheme in the station displays on the login page as well as on the prelogin page. Users logging in select a scheme by clicking one of those buttons. When using multiple SSO schemes, it is a good idea to configure the Login Button Text for each with a meaningful label. For example, OpenAM SSO Login.
Remember My Choice Domain	text string, null (default)	If no value in this field, logging in with SSO sets a cookie for that domain (i.e. <code>controller1.myDomain.com</code>) on that station only. If a domain name is entered in the field the effect is that a user only has to login to one station to set a cookie for that domain on all networked stations. For example, if stations all follow the pattern <code>controller1.myDomain.com</code>, <code>controller2.myDomain.com</code>, etc..., entering <code>myDomain.com</code> will cause a cookie for this domain to be set on all of the stations. This is especially useful in an environment where Auto Attempt Single Sign On is set to false.

Parent topic: [Components in baja module](#)

This component represents a station in the framework.

The Station is available in a fox connection to a host Niagara platform, along with its file space (**Files**) and histories (**History**).

See the *Niagara Developer Guide* for developer information.

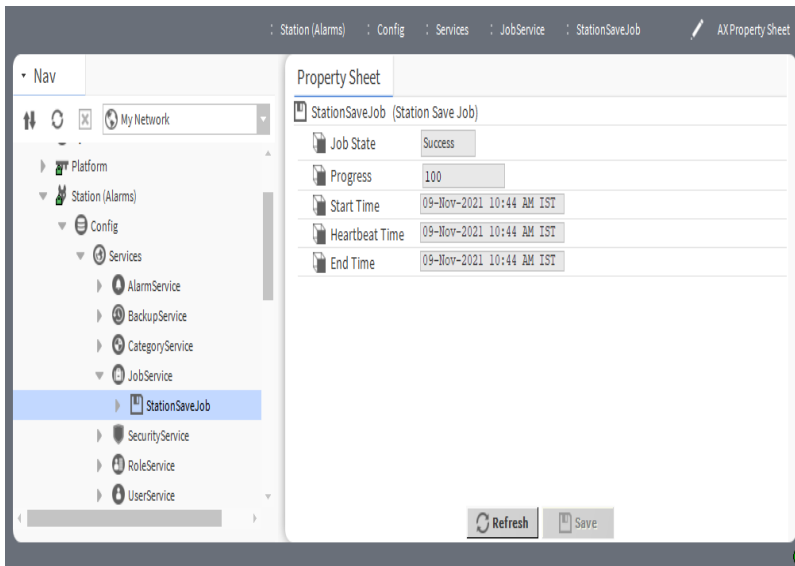
Station Save

Save the current state of the system to persistent storage. You should regularly backup your station or stations using the **Save** Action on the station.

Parent topic: [Components in baja module](#)

This component appears as a child of the **Job Service** and displays the following properties relative to the save job:

Figure 1. StationSaveJob properties



Property	Value	Description
Job State	read-only	Displays the current or final state of the job. The first three states appear on the Device Network Job view. Unknown indicates the job is pending execution. Running indicates the job is executing. Cancelling indicates a request to cancel the job was sent, but has not been processed yet. the job is still executing. Success indicates the job finished successfully, with all steps completed for all stations. Cancelled indicates the job was canceled before it completed, and is no longer running. Failed indicates at least one step failed in one station. The job is no longer running.

Property	Value	Description
		Each row in the table ends with a details link (>>) and a dispose button (X). Clicking this button changes the view to the Niagara Network Job view or the Batch Job Step Log File view. These views show all logged messages that are related to this single job. The overall status for the job in other stations may be different.
Progress	read-only	Shows the percentage of progress toward completing the job.
Start Time	read-only	Displays the time that the job started.
Heartbeat Time	read-only	Displays the time of the last indication that the job is alive.
End Time	read-only	Displays the time that the job ends.

Note: All jobs in a station are cleared upon a station restart.

Parent topic: [Components in baja module](#)

SyntheticModuleFile (synthetic module) is a synthetic Java archive (.sjar) that allows the creation of memory-resident modules and types programmatically at station runtime. Synthetic modules differ from standard .jar (non-synthetic) modules in a number of ways, and have a special default **Synthetic Module File View** for editing contents.

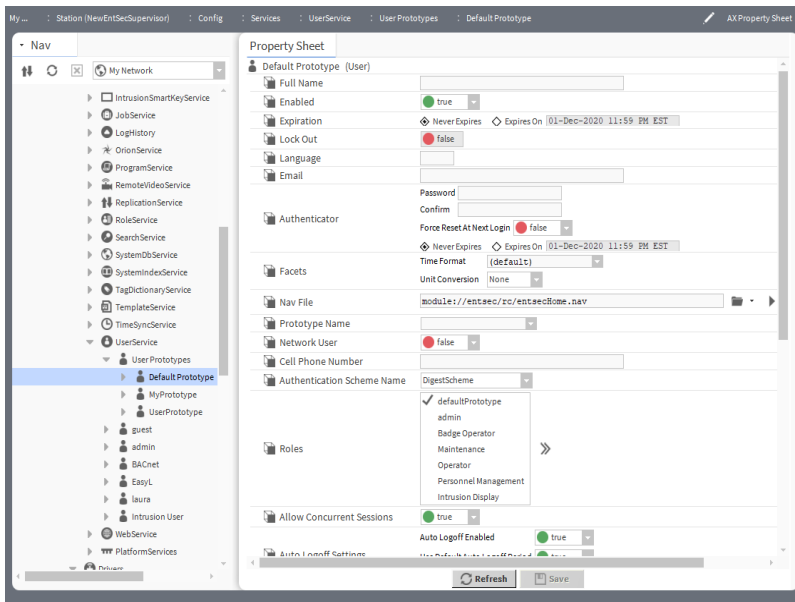
Parent topic: [Components in baja module](#)

UnrestrictedFolder is a special container designed to store objects for use on a palette. Normal isParentLegal checks are not applied to UnrestrictedFolders. The **UnrestrictedFolder** is available in the **baja** palette.

Parent topic: [Components in baja module](#)

This component represents a station connection, typically a specific type of person who needs to access the system or another system that makes a station-to-station connection (machine-to-machine). Users are children of the station's **UserService > UserPrototypes** container.

Figure 1. Some Default Prototype properties



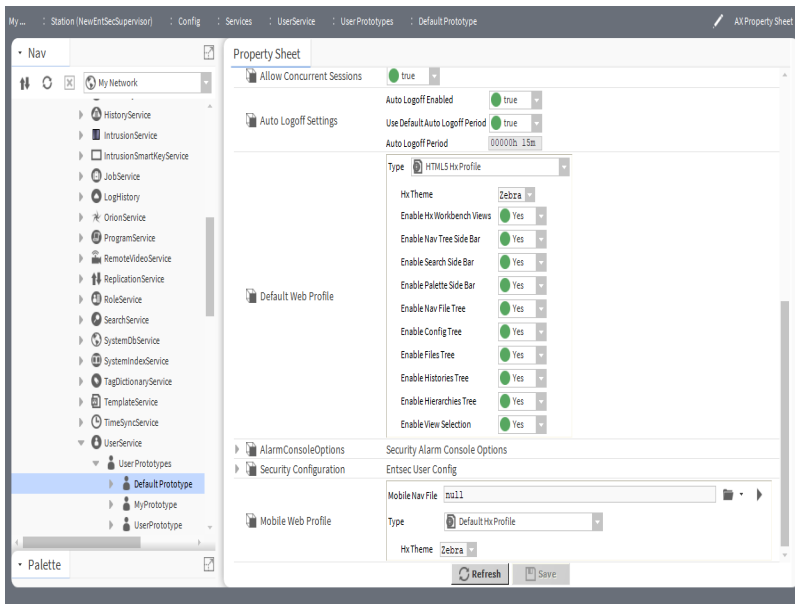
To open these properties, expand **Config > Services > UserService > User Prototypes** and double-click **Default Prototype**.

In addition to Enabled, these properties configure default users.

Property	Value	Description
Full Name	text	Provides additional information.
Expiration, Never Expires	radio button (enabled by default)	Indicates that this user is valid without restriction.
Expiration, Expires On	date and time	Limits the validity of this user. - Never Expires permits this user to always log in. - Expires On [date and time] allows this user to log in until the expiration date and time.
Lock Out	true or false (default)	Controls this user's ability to access the system. true prohibits the user from logging in. false allows the user to log in without restriction.
Language	two lower-case letters	Defines the ISO 639 language code. For a list of codes, see the following: http://www.loc.gov/standards/iso639-2/langcodes.html .
Email	text	Provides an email address for information about this user.

Property	Value	Description
Authenticator, Password and Confirm	text	Provides a password for this user.
Authenticator, Force Reset At Next Login	true or false (default for a new station)	Determines if the station resets the password when a user logs in. The next time the user logs in the system prompts for a password change. If you find yourself typically changing this each time you create a new User, change it to desired value in the Default Prototype .
Authenticator, Expires	radio button (defaults to enabled)	Saves the password for this user indefinitely. Typically, you leave this at default. However, it is possible you might change this to some far future date in the Default Prototype .
Authenticator, Expires On	radio button (defaults to disabled)	Sets a specific date and time after which the user must change their password.
Facets, Time Format		Configures the time format.
Facets, Unit Conversion		Configures the units to use.
Nav File	ORD	Identifies the file that contains the software navigation (customized navigation tree) appropriate for this user.
Prototype Name	drop-down list (defaults to Default Prototype)	Identifies the name of the prototype used to synchronize this user with the same user in other stations. The value of this property applies only when Network User is true.
Network User	true or false (default)	Configures this user to synchronize with the same user in other stations. true indicates that this user can be synchronized with other stations. false configures this user as a local user in this particular station only.
Cell Phone Number	telephone number	Provides a mobile number for information about this user.
Authentication Scheme Name	drop-down list	Selects the authentication scheme for user identity.
Roles	list	Identifies this user's role(s).
Allow Concurrent Sessions	true (default) or false	Configures the station for multiple sessions. true permits multiple sessions. false prohibits multiple sessions.

Figure 2. The rest of the Default Prototype properties



To view these properties, scroll down.

Property	Value	Description
Auto Logoff Settings, Auto Logoff Enabled	true (default) or false	Controls automatic logoff. true automatically disconnects a station connection (session) via Workbench or web a browser if a period of inactivity exceeds the amount of time specified for the Default Auto Logoff Period as defined in the UserService) false does not automatically terminate a user's session due to inactivity. Note: Separate auto logoff options exist for Workbench which functions independently of these station settings.
Auto Logoff Settings, Use Default Auto Logoff Period	true or false (default)	Configures the default setting for automatic logoff. true configures a specific Default Auto Logoff Period time. false ignores the specified Default Auto Logoff Period. Instead, you can use the Auto Logoff Period property to set a different auto logoff time period.
Auto Logoff Settings, Auto Logoff Period	read-only and hours minutes (defaults to 15 minutes)	Configures the amount of time until an automatic logoff. When Use Default Auto Logoff Period is set to false, this property specifies a different time period range from 2 minutes to 4

Property	Value	Description
		hours. Otherwise, this property is read-only, showing the value specified in the UserService 's Default Auto Logoff Period.
Default Web Profile, Type	sub properties	Configure how the framework functions in a browser. The first property, Type, selects the type of browser profile. This property defaults to HTML5 Hx Profile. Each profile provides a separate set of properties. The sections that follow in this topic document each profile.
Mobile Web Profile, Type	sub properties	Configure how the framework functions in a cellular device. Each profile provides a separate set of properties. The sections that follow in this topic document each profile.
Video Alarm Console Options	additional properties	Configures video alarms. "Video Alarm Console options (videoDriver-VideoAlarmConsoleOptions)" documents the additional properties.
Alarm Popup Settings	additional properties	Configures popup settings. "Console Recipient Monitor (videoDriver-ConsolerecipientMonitor)" documents the additional properties.
SAML Prototypes	container	Holds SAML prototypes.

HTML5 HX Profile

These properties configure the HTML5 HX Profile.

Property	Value	Description
Mobile Nav File (available for Mobile Wed Profiles only)	ORD	Selects the navigation file for a mobile device.
Hx Theme	drop-down list (defaults to Zebra)	Selects the look of the user interface: • Zebra • Lucid
Enable Hx Workbench Views	true (default) or false	Configures the interface to display all Workbench views.
Enable Nav Tree Side Bar	true (default) or false	Configures the interface to display the navigation tree (Nav tree).

Property	Value	Description
Enable Search Side Bar	true (default) or false	Configures the interface to display the Search side bar.
Enable Palette Side Bar	true (default) or false	Configures the interface to display the Palette side bar.
Enable Nav File Tree	true (default) or false	Configures the interface to display the nav file tree in general.
Enable Config Tree	true (default) or false	Configures the interface to display the Config tree.
Enable Files Tree	true (default) or false	Configures the interface to display the Files tree.
Enable Histories Tree	true (default) or false	Configures the interface to display the History tree.
Enable Hierarchies Tree	true (default) or false	Configures the interface to display the Hierarchy tree.
Enable View Selection	true (default) or false	Configures the interface to display the navigation tree (Nav tree).

Velocity Doc Web Profile

Property	Value	Description
Template File ORD	ORD (defaults to null)	

Basic Hx Profile, Default Hx Profile, Handheld Hx Profile,

Property	Value	Description
Mobile Nav File (available for Mobile Web Profiles only)	ORD	Selects the navigation file for a mobile device.
Hx Theme	drop-down list (defaults to Zebra)	Selects the look of the user interface: • Zebra • Lucid

Default Mobile Web Profile (obsolete)

Property	Value	Description
Theme, drop-down list	defaults to mobile	
Theme, second drop-down list	defaults to DefaultjQueryMobileTheme	
Show Header	Show (default) or Hide	
Show Header Back Button	Show (default) or Hide	

Property	Value	Description
Show Select Views	Show (default) or Hide	
Show Home	Show (default) or Hide	
Show Logoug	Show (default) or Hide	

Default Mobile Handheld Hx Profile

Property	Value	Description
Mobile Nav File (available for Mobile Wed Profiles only)	ORD	Selects the navigation file for a mobile device.
Hx Theme	drop-down list (defaults to Zebra)	Selects the look of the user interface: • Zebra • Lucid

Basic Wb Web Profile, Default Wb Web Profile, Handheld Wb Web Profile and Simple Admin Wb Web Profile

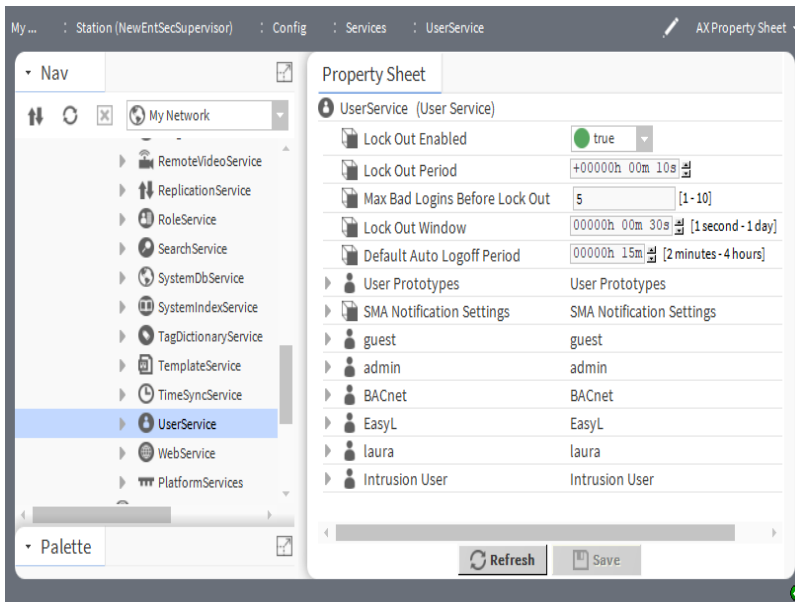
Property	Value	Description
Workbench Theme	drop-down list (defaults to Zebra)	Selects the look of the user interface: • Zebra • Lucid
Show Web Start Address Bar	true (default) or false	
Show Web Start Status Bar	true (default) or false	

Parent topic: [Components in baja module](#)

This service manages all system users: human and machine. You access it by right-clicking **UserService** and clicking **Views > Property Sheet**.

The **User Manager** is the primary view of this service. By default, creating a new station using the **New Station** wizard includes the **UserService**. The baja module makes this service available.

Figure 1. User Service property sheet view



To access these properties, expand **Config > Services**, right-click **UserService** and click **Views > AX Property Sheet**.

Property	Value	Description
Lock Out Enabled	true or false	Controls a user's ability to log in. true temporarily disables login (user account) access based on a number of consecutive authentication failures for the duration of the Lock Out Period. Using lock out makes it difficult to automate the guessing of passwords. Note: Each user has a Clear Lock Out action.
Lock Out Period	true or false	Defines how long a user is unable to log in to a user account after consecutive authentication failures. While locked out, any login attempt (even a valid one) is unsuccessful. Note: The 10 second latency (default) guards against an automated, brute-force password attack, where a computer application issues hundreds of login attempts a second. You can adjust this value to guard against human attack.
Max Bad Logins Before Lock Out	Number from 1–10 (defaults to 5)	Specifies the number of consecutive failed user login attempts that trigger a lock out when Lock Out Enabled is true and Lock Out Window expires.
Lock Out Window	hours minutes seconds (defaults to 30 seconds)	Defines how long the user has to successfully log in when Lock Out Enabled is set to true.

Property	Value	Description
		If a user fails to log in successfully before the Max Bad Logins Before Lock Out window (period) expires, the system locks the user out for the duration of the Lock Out Period. The system enforces changes to lock out properties the next time the user logs in. For example, if Max Bad Logins Before Lock Out is set to 5, user ScottF fails to log in four times within the Lock Out Window, and an admin-level user changes Max Bad Logins Before Lock Out to 3, the change does not lock ScottF out. User ScottF still has one more chance to log in before getting locked out. If ScottF's fifth attempt to log in fails, the system locks him out the next time he attempts to log in because five failed attempts is greater than or equal to the Max Bad Logins Before Lock Out of 3.
Default Auto Logoff Period	0000h 15m (default)	Specifies the amount of time that a period of inactivity may last before a station connection is automatically disconnected. The acceptable range of values is two minutes to four hours. This limit is observed only when the User's Use Default Auto Logoff Period property is set to true.
SMA Notification Settings	multiple properties	Configures the SMA (Software Maintenance Agreement) whose properties are documented in <i>baja-SMANotificationSettings</i> , which is in the <i>Getting Started with Niagara</i> .
User Prototypes	multiple properties	Serves as a container for the default and other user prototypes whose properties are documented in <i>baja-UserPrototypes</i> , which is in the <i>Getting Started with Niagara</i> .

Parent topic: [Components in baja module](#)

UserServicePasswordConfiguration (Password Configuration) is a child container under the **UserService**.

It contains global properties to define the periodic password expiration for station users as well as the unique password history setting. For more details, see the *Niagara Station Security Guide*. Also note each station User has a related Password Configuration child container as well.

Note: This component is not available in a new station until it is started and saved.

Parent topic: [Components in baja module](#)

Vector is a dynamic Struct which allows properties to be added at runtime.

Parent topic: [Components in baja module](#)

A VirtualComponent is the Baja base class for implementations of transient (non-persisted) components that reside in the station's virtual component space, as defined by a VirtualGateway.

Initial applications of virtual components are expected in various drivers for Niagara. For details, see the *Niagara Drivers Guide*.

Parent topic: [Components in baja module](#)

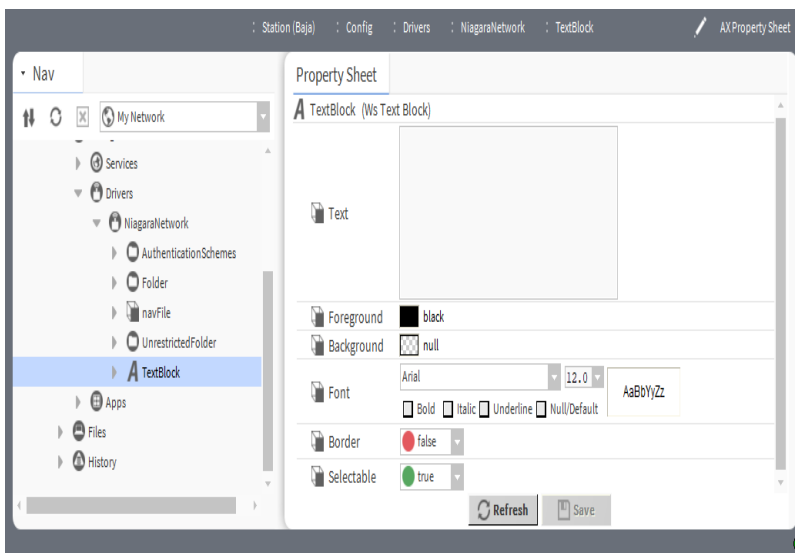
A VirtualGateway is the Baja base class for a component that resides under the station's component space (Config), and acts as a gateway to the station's virtual component space.

Other object spaces are Files and History. Initial applications of virtual gateways are expected in Niagara drivers. For details, see the *Niagara Drivers Guide*.

Parent topic: [Components in baja module](#)

WsTextBlock (**TextBlock**) is a component you can drop onto a wire sheet (WireSheet) and position to add text notes. Properties include Text (to display), Foreground and Background colors, Font, Border, and whether the Text Block is directly Selectable in the wire sheet (by default, Selectable is true). If Selectable is false, you must select this component via its node in the Nav tree.

Figure 1. WsTextBlock properties



To access, expand **Config > Drivers > NiagaraNetwork** and double-click **TextBlock**.

Property	Value	Description
Text	text	Specifies the text which is displayed in the text box.
Foreground	Solid, Gradient, Image ,Null(default)	Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window. Click the browser icon to open a file chooser and ord chooser to select the image file.

Property	Value	Description
		Null indicates no foreground fill.
Background	Null (default)	Specifies the background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white).
Font	drop-down list	Selects the font in which the text is displayed.
Border	true or false (default)	If set to true text box is displayed with a border. If set to false text box is displayed without a border.
Selectable	true or false (default)	If set to true text is selectable. If set to false text is not selectable.

Parent topic: [Components in baja module](#)

ZipFile represents a zip file in the file system of a session.

Parent topic: [Components in baja module](#)

Components in chart module are:

- BarChart
- ChartCanvas
- ChartHeader
- ChartPane
- DefaultChartLegend
- LineChart
- [Chart Pane\(chart-BarChart/LineChart\)](#)
Charts are useful to analyze the data. There are two types of charts available, BarChart and LineChart. Whereas BarChart data have set of bars and LineChart data is plotted on each series as a line.
- [Chart Canvas\(chart-ChartCanvas\)](#)
This component is the canvas widget under a ChartPane. Canvas is a editor used to draw the visual graphics, you can edit the properties of the canvas and adding widgets to the canvas pane to bind the data together in a single page.
- [Chart Header\(chart-ChartHeader\)](#)
This component is the header widget under a ChartPane. It describes about the information of a chart, you can change the properties of information present in the chart.

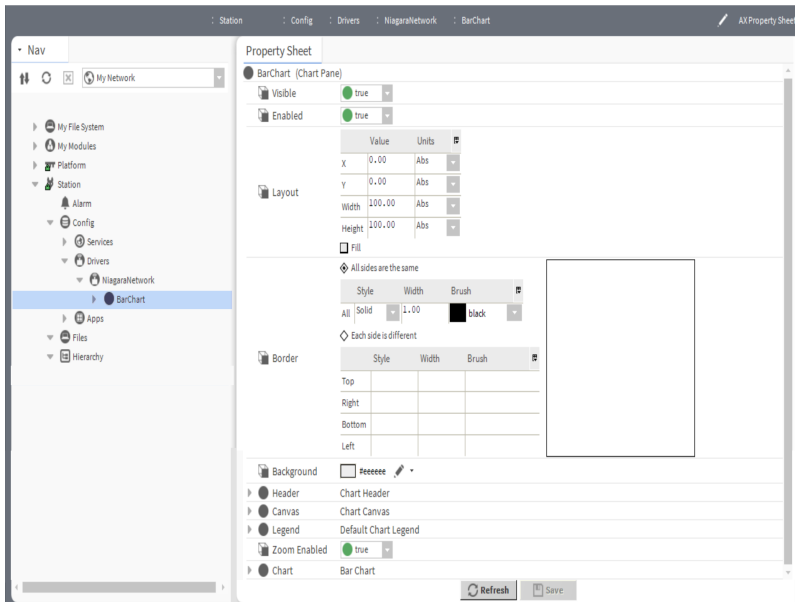
- **BarChart/LineChart(chart-BarChart/LineChart)**
This component is the container widget created when adding a Px chart.
- **Default Chart Legend(chart-DefaultChartLegend)**
This component is the legend widget under a ChartPane.
- **chart-LineChart**
This component is one of two chart types available (the other is BarChart).

Parent topic: [Component guides](#)

Charts are useful to analyze the data. There are two types of charts available, BarChart and LineChart. Whereas BarChart data have set of bars and LineChart data is plotted on each series as a line.

The **BarChart/LineChart** component is available in the Chart palette.

Figure 1. Chart Pane properties



To access these properties, expand **Station > Config > Drivers > NiagaraNetwork** and double-click on **BarChart/LineChart**

Property	Value	Description
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).
Enabled	true (default) false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).

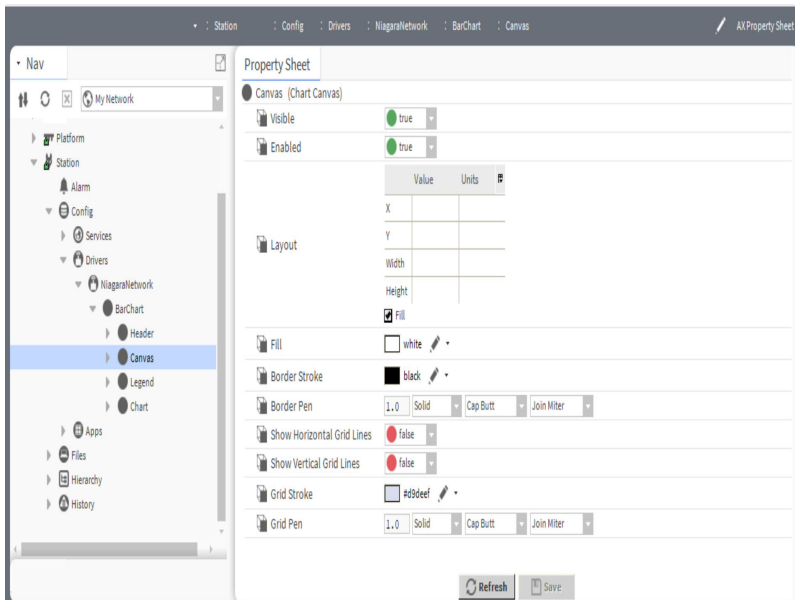
Property	Value	Description
Border	Text	Specifies the style and design of the border
Background	drop-down list (defaults to Null)	Specifies the background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white).
Header	additional properties	Configures additional parameters of a header
Canvas	additional properties	Configures additional parameters for canvas.
Legend	additional properties	Configures additional parameters for chart legend.
Zoom Enabled	true (default) false	When set to true allows the chart to zoom in. When set to false does not allow the charts to zoom in.
Chart	additional properties	Configures additional parameters for BarChart/LineChart.

Parent topic: [Components in chart module](#)

This component is the canvas widget under a ChartPane. Canvas is a editor used to draw the visual graphics, you can edit the properties of the canvas and adding widgets to the canvas pane to bind the data together in a single page.

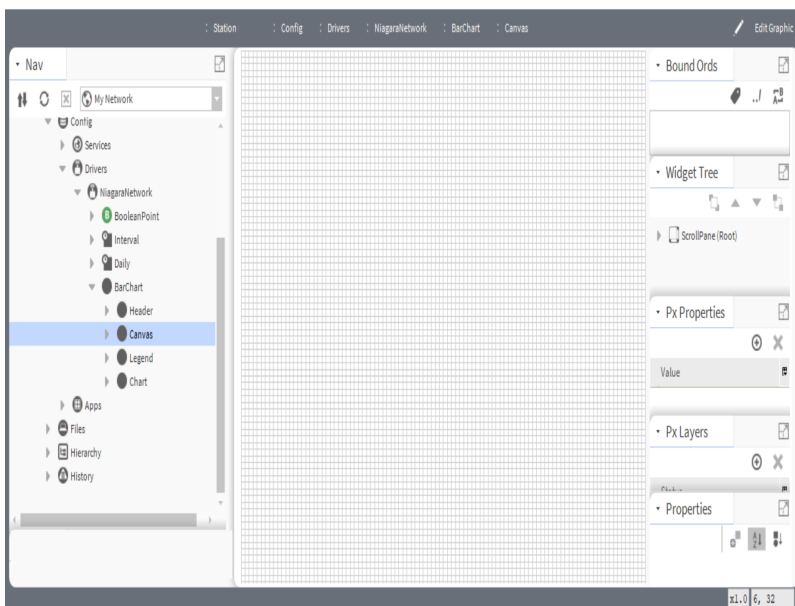
A Canvas is available in **BarChart/LineChart** component in the Chart Palette.

Figure 1. Chart Canvas properties



To access these properties, expand **Station > Config > Drivers > NiagaraNetwork > BarChart/LineChart** and double-click on **Canvas**

Figure 2. Canvas view



To access the canvas page, right-click the **Canvas > Views > New View** and create a new Px view, click **Ok**.

The New Px canvas page opens, right-click on the canvas page and click **Edit Properties** to make changes.

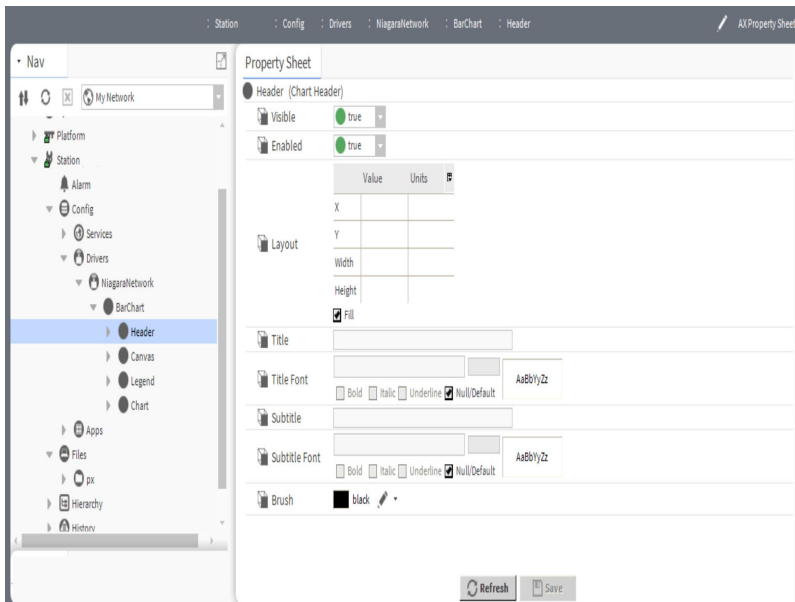
Property	Value	Description
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).
Enabled	true (default) false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).
Fill	drop-down list (defaults to Null)	Specifies the type of color to fill the pane. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window. Click the browser icon to open a file chooser and ord chooser to select the image file. Null indicates no background fill.
Border Stroke	drop-down list (defaults to Null)	Specifies color of the border. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no border fill.
Border pen	additional properties	Configures additional properties of a border.
Show Horizontal Grid Lines	true false(default)	When set to true horizontal grid lines are visible. When set to false horizontal grid lines are invisible.
Show Vertical Grid Lines	true false(default)	When set to true vertical grid lines are visible. When set to false vertical grid lines are invisible.
Grid Stroke	drop-down list (defaults to Null)	Specifies color of the grid. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no grid fill.
Grid Pen	additional properties	Configures additional parameters for grid pen.

Parent topic: [Components in chart module](#)

This component is the header widget under a ChartPane. It describes about the information of a chart, you can change the properties of information present in the chart.

The Header is available in **BarChart/LineChart** component in the Chart Palette.

Figure 1. Chart Header properties



To access these properties, expand **Station > Config > Drivers > NiagaraNetwork > BarChart/LineChart** and double-click **Header**.

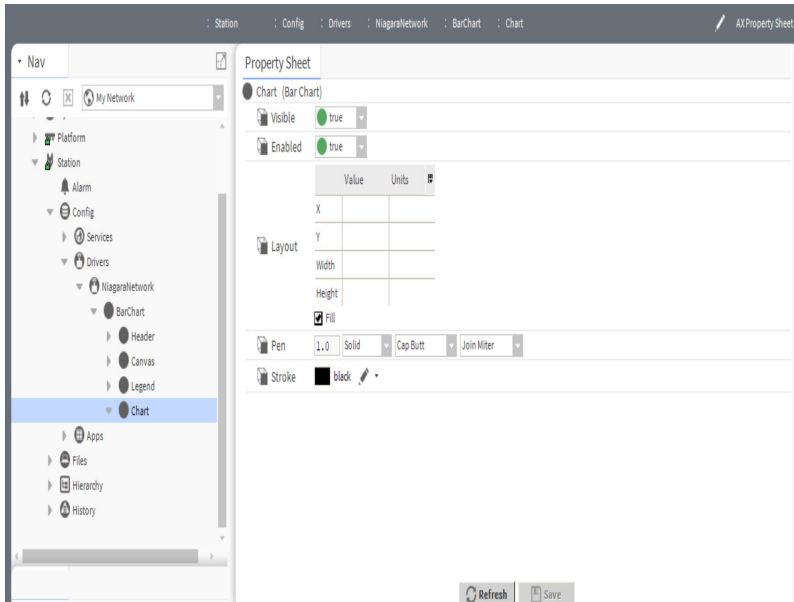
Property	Value	Description
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).
Enabled	true (default) false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).
Title	text	Specifies the title of the header.
Title Font	additional properties	Configures the font parameters for the title of the header.
Subtitle	text	Specifies the subtitle of the header.
Subtitle Font	additional properties	Configures the font parameters for the subtitle of the header.
Brush	drop-down list (defaults to Null)	Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other

Property	Value	Description
		method of selecting an image file. Null indicates no fill.

Parent topic: [Components in chart module](#)

This component is the container widget created when adding a Px chart.

Figure 1. Chart Pane Properties



To access these properties, expand **Station > Config > Drivers > NiagaraNetwork > BarChart/LineChart** and double-click on **Chart**.

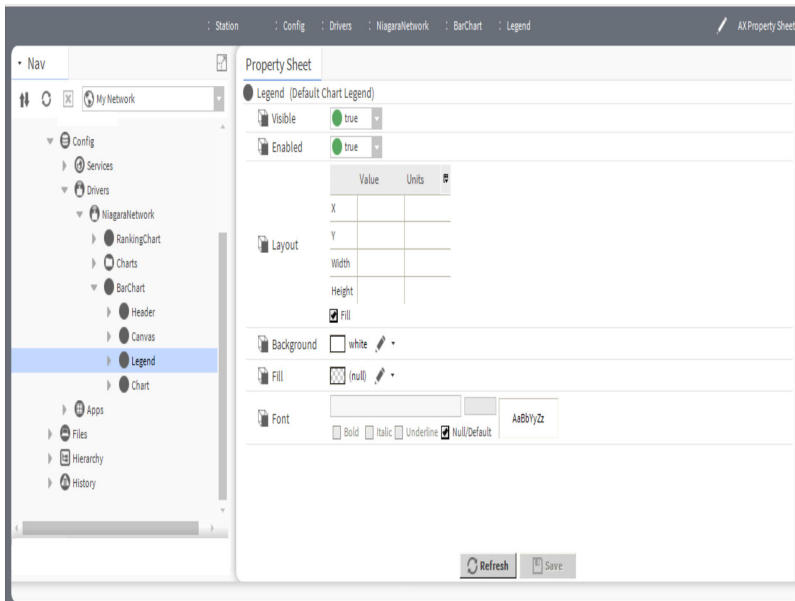
Property	Value	Description
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).
Enabled	true (default) false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).
Pen	additional properties	Configures the additional properties of a border
Stroke	drop-down list (default to Null)	Solid opens the Color Chooser window. Gradient opens the Gradient Editor

Property	Value	Description
		window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no fill.

Parent topic: [Components in chart module](#)

This component is the legend widget under a ChartPane.

Figure 1. Default Chart Legend properties



To access these properties, expand **Station > Config > Drivers > NiagaraNetwork > BarChart/LineChart** and double-click on **Legend**.

Property	Value	Description
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).
Enabled	true (default) false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).
Background	drop-down list (defaults to Null)	Specifies the background fill color. Solid opens the Color Chooser window.

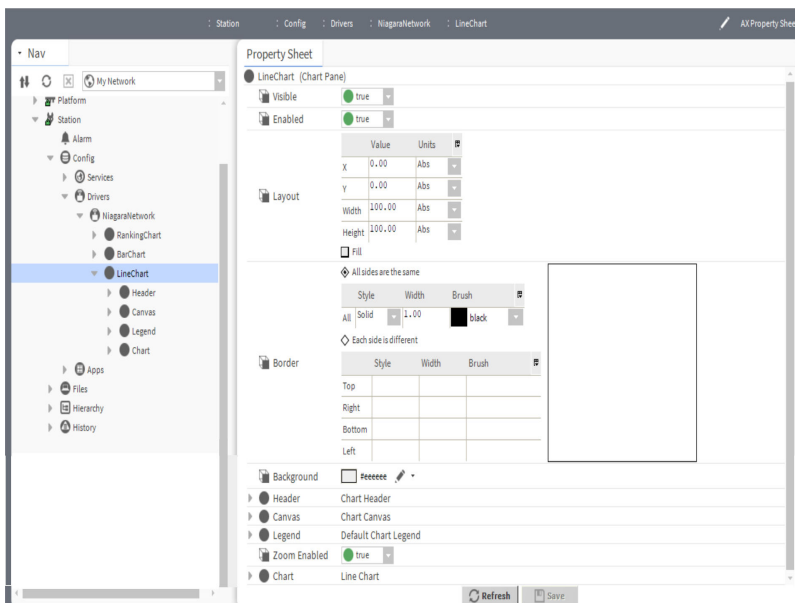
Property	Value	Description
		Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white).
Fill	drop-down list (defaults to Null)	Specifies the type of color to fill the pane. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window. Click the browser icon to open a file chooser and ord chooser to select the image file. Null indicates no background fill.
Font	additional properties	Configures the font parameters for the legend

Parent topic: [Components in chart module](#)

This component is one of two chart types available (the other is BarChart).

The **LineChart** component is available in the **Chart** palette.

Figure 1. LineChart Property Sheet



You can access the properties in the **PropertySheet** by expanding **Station > Config > Drivers** and double-click on **LineChart**

Property	Value	Description
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).
Enabled	true (default) false	When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable.
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).
Border	Text	Specifies the style and design of the border
Background	Solid, Gradient, Image ,Null(default)	Specifies the background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white).
Header	additional properties	Configures additional parameters of a header
Canvas	additional properties	Configures additional parameters for canvas
Legend	additional properties	Configures additional parameters for chart legend.
Zoom Enabled	true (default) false	When set to true allows the chart to zoom in. When set to false does not allow the charts to zoom in.
Chart	additional properties	Configures additional parameters for chart.

Parent topic: [Components in chart module](#)

There are three folders in the **control** palette: Points, Extensions, and Trigger.

Components in the folders are as follows:

- BooleanPoint
- BooleanWritable
- NumericPoint

- NumericWritable
- EnumPoint
- EnumWritable
- StringPoint
- StringWritable
- DiscreteTotalizerExt
- NumericTotalizerExt
- NullProxyExt
- TimeTrigger
- [control-BooleanPoint](#)
This component is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.
- [control-BooleanWritable](#)
This component extends **BooleanPoint** to include 16 levels of command priority control. The highest priority active command is reflected in the **Out** property. Commands at the emergency and manual levels (1 and 8) are stored persistently.
- [control-NumericPoint](#)
This point is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.
- [control-NumericWritable](#)
NumericWritable extends NumericPoint to include 16 levels of command priority control. The highest priority active command is reflected in the Out property. Commands at the Manual and manual levels (1 and 8) are stored persistently.
- [control-EnumPoint](#)
EnumPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.
- [control-EnumWritable](#)
WritableEnumPoint extends EnumPoint to include 16 levels of command priority control. The highest priority active command is reflected in the out property. Commands at the emergency and manual levels (1 and 8) are stored persistently.
- [control-StringPoint](#)
This point is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.
- [control-StringWritable](#)
StringWritable extends StringPoint to include 16 levels of command priority control. The highest priority active command is reflected in the Out property. Commands at the Manual and manual levels (1 and 8) are stored persistently.
- [control-DiscreteTotalizerExt](#)
- [control-NumericTotalizerExt](#)
NumericTotalizerExt is a control point extension used for integrating a numeric point value over time. For example, a totalizer with a minutely totalization interval can convert an instantaneous flow reading in cubic feet per minute (cfm) into a value representing total cubic feet consumed. An available resetTotal action clears (zeroes) any accumulated total.
- [control-NullProxyExt](#)
NullProxyExt is the default implement AbstractProxyExt used to indicate that a point is not a proxy point. The NullProxyExt is in the **control** palette's **Extensions** folder, but is already present by default on compatible point types. Presence indicates that you can add other types of extensions to the parent component, for example alarm or history extensions.

- [control-TimeTrigger](#)

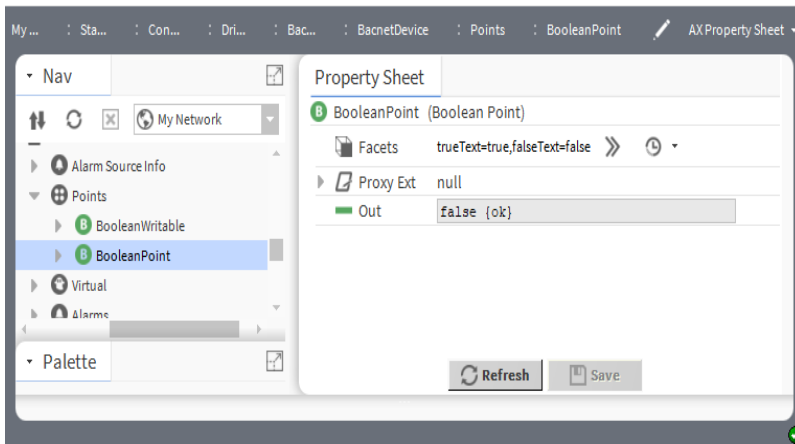
This component fires a topic at configured times. It is available in the **Trigger** folder of the **control** palette.

Parent topic: [Component guides](#)

This component is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.

A BooleanPoint is available in the **Points** folder of the control palette.

Figure 1. Boolean Writable properties



You can access these properties by double-clicking on the component wherever a Boolean control point is used.

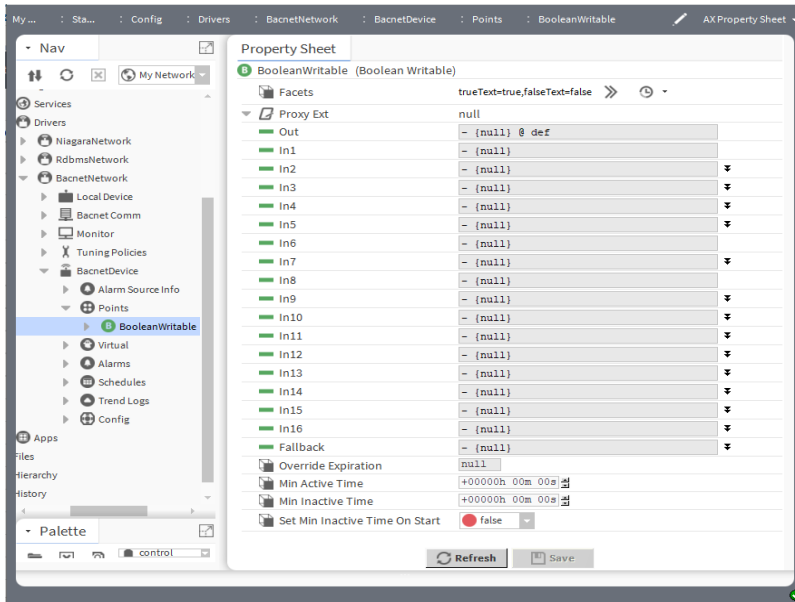
Property	Value	Description
Facets	Config Facets window	Define the text to display for the Boolean values: • trueText is the text to display when output is true • falseText is the text to display when output is false. For example, the facet trueText could display “ON” and the facet falseText “OFF.” You view Facets on the Slot Sheet and edit them from a component Property Sheet by clicking the >> icon to display the Config Facets window.
Proxy Ext, Out	read-only	Provides a current binary value, facets (text descriptors) and data status, such as fault, overridden, alarm, and so on. If no status flag is set, status is considered normal and reports {ok}.

Parent topic: [Components in control module](#)

This component extends **BooleanPoint** to include 16 levels of command priority control. The highest priority active command is reflected in the Out property. Commands at the emergency and manual levels (1 and 8) are stored persistently.

This point is available in the **Points** folder of the control palette and usually resides in a device's **Points** folder.

Figure 1. Boolean Writable properties



You can access these properties by double-clicking the component wherever a Boolean point is used. The point's characteristic green circle icon with a B in the middle identifies Boolean writable points in the Nav tree and **Wire Sheet**.

Property	Value	Description
Facets	Config Facets window	Define the text to display for the Boolean values: • trueText is the text to display when output is true • falseText is the text to display when output is false. For example, the facet trueText could display "ON" and the facet falseText "OFF." You view Facets on the Slot Sheet and edit them from a component Property Sheet by clicking the >> icon to display the Config Facets window.
Proxy Ext, Out	message	Provides a current binary value, facets (text descriptors) and data status, such as <code>fault</code> , <code>overridden</code> , <code>alarm</code> , and so on. If no

Property	Value	Description
		status flag is set, status is considered normal and reports { ok } . The station itself originates the point's default Out value.
Proxy Ext, In1–16	values	Indicate from where the point's value originates, including details specific to the parentage of the point's network and communications (driver). <code>null</code> indicates that the point is an empty placeholder.
Proxy, Fallback	value	Configures a default Out value for a point when a priority scan of inputs In2 – In7 and In9 – In16 returns a null or invalid value, and no emergency action (In1) or manual action (In8) has occurred.
Override Expiration	read-only	Reports how long a value that has been set manually (using an action) remains valid.
Min Active Time	hours minutes seconds (defaults to 00000h 00m 00s)	Defines a minimum up time. This property works independently of Min Inactive Time. You typically use this property to prevent the short-cycling of equipment controlled by the point. The default disables the timer.
Min Inactive Time	hours minutes seconds (defaults to 00000h 00m 00s)	Defines a minimum down time. This property works independently of Min Active Time. You typically use this property to prevent the short-cycling of equipment controlled by the point. The default disables the timer.
Set Min Inactive Time On Start	true or false (default)	Determines if the minimum inactive time applies at station start.

Actions

This component supports the following actions:

- **Emergency Active:** sets an active output at the emergency level.
- **Emergency Inactive:** sets an inactive output at the emergency level.
- **Emergency Auto:** clears any active or inactive output at the emergency level.
- **Active:** sets an active output at the manual level.
- **Inactive:** sets an inactive output at the manual level.
- **Auto:** clears any active or inactive output at the manual level.

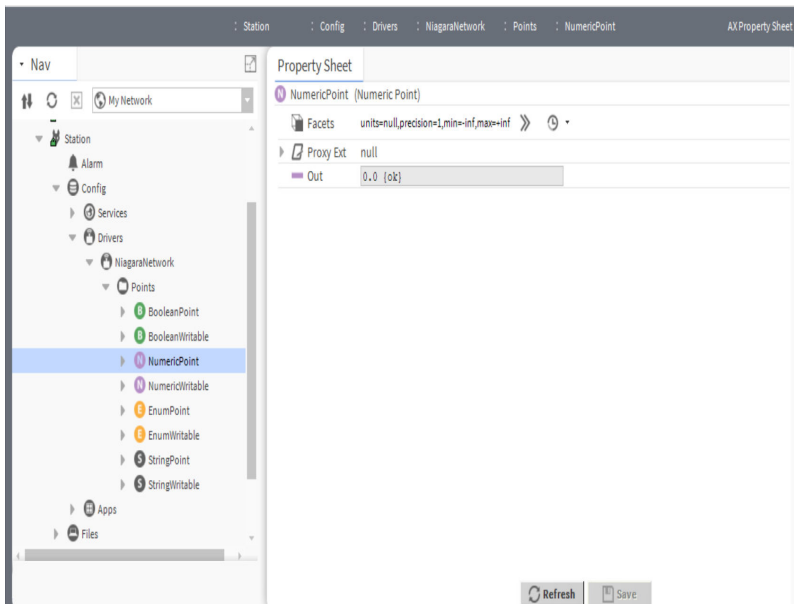
- **Set:** opens the **Set** window with true and false options, which you can use to manually configure the point's value.

Parent topic: [Components in control module](#)

This point is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.

This point is available in the **Points** folder of the control palette and usually resides in a device's **Points** folder.

Figure 1. Numeric Point Property Sheet



You can access these properties in the **Property Sheet** by expanding **Station > Config > Drivers > Points** and double-click on **NumericPoint**.

You can access these properties wherever numeric point is used. The numeric point values display in units and decimal precision.

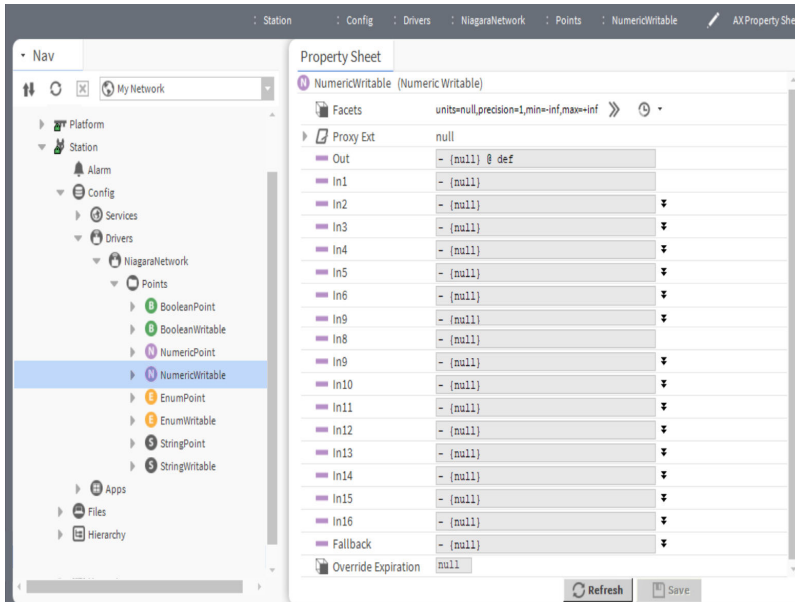
Property	Value	Description
Facets	Config Facets window	Defines the unit of measure for a numeric value: units defines the unit of measure, for example from acceleration to volumetric flow. precision defines the number of decimal places.
Proxy Ext, Out	value	Reports an analog numeric value and status, such as <code>fault</code> , <code>overridden</code> , <code>alarm</code> , and so on. If no status flag is set then status is considered normal and reports <code>{ok}</code> .

Parent topic: [Components in control module](#)

NumericWritable extends NumericPoint to include 16 levels of command priority control. The highest priority active command is reflected in the Out property. Commands at the Manual and manual levels (1 and 8) are stored persistently.

This point is available in the **Points** folder of the control palette and usually resides in a device's **Points** folder.

Figure 1. Numeric Writable property Sheet



You can access these properties in the **Property Sheet** by expanding **Station > Config > Drivers > Points** and double-click on **NumericWritable**.

Property	Value	Description
Facets	Config Facets window	Defines the unit of measure for a numeric value: units defines the unit of measure, for example from acceleration to volumetric flow. precision defines the number of decimal places.
Proxy Ext, Out	value	Reports an analog numeric value.
Proxy Ext, In1–16	values	Indicate from where the point's value originates, including details specific to the parentage of the point's network and communications (driver). null indicates that the point is an empty placeholder.
Proxy, Fallback	value	Configures a default Out value for a point when a priority scan of inputs In2 – In7 and In9 – In16 returns a null or invalid value, and no emergency action

Property	Value	Description
		(In1) or manual action (In8) has occurred.
Override Expiration	read-only	Reports how long a value that has been set manually (using an action) remains valid.

Actions

This component supports the following actions:

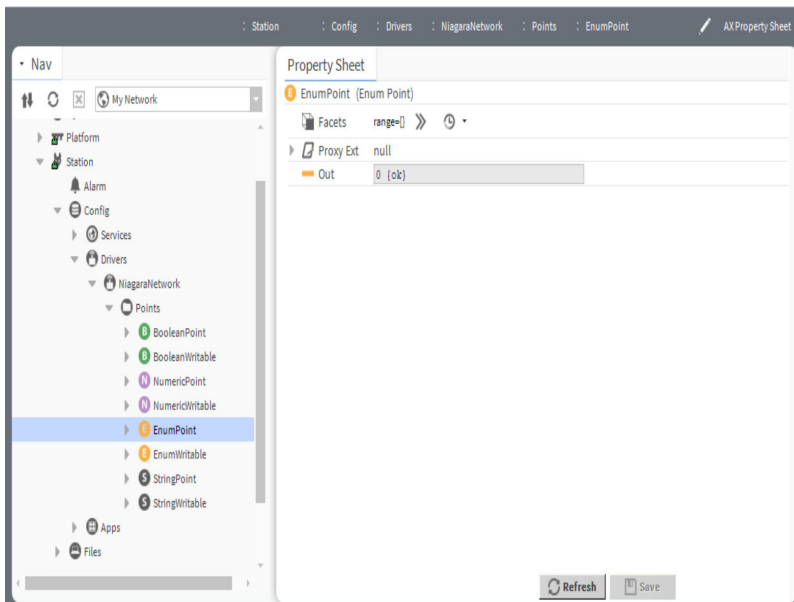
- **Emergency Override:** sets an output value at the emergency level.
- **Emergency Auto:** clears an output value set at the emergency level.
- **Override:** sets an output value at the manual level.
- **Auto:** clears an output value set at the manual level.
- **Set**
: opens the **Set** window where you can enter a numeric value with a single decimal place.

Parent topic: [Components in control module](#)

EnumPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.

This point is available in the **Points** folder of the control palette and usually resides in a device's **Points** folder.

Figure 1. Enum Point Property Sheet



You can access these properties in the **Property Sheet** by expanding **Station > Config > Drivers > Points** and double-click on **EnumPoint**.

You can access these properties wherever numeric point is used. The enum point values can be given in operating range of the component.

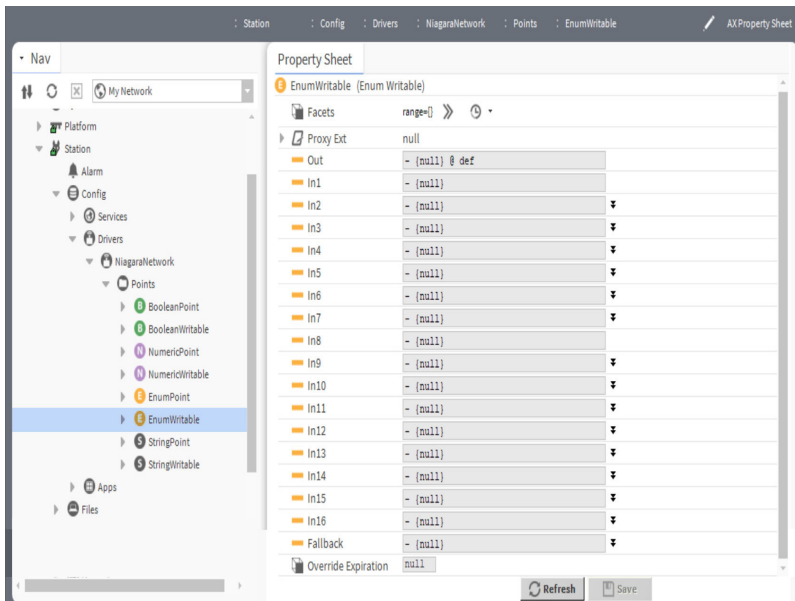
Property	Value	Description
Facets	Config Facets Window	Defines a range of enumerated states (enum values).
Proxy Ext, Out	value	Reports a value in the enum range and status, such as <code>fault</code> , <code>overridden</code> , <code>alarm</code> , and so on. If no status flag is set then status is considered normal and reports <code>{ok}</code> .

Parent topic: [Components in control module](#)

WritableEnumPoint extends EnumPoint to include 16 levels of command priority control. The highest priority active command is reflected in the out property. Commands at the emergency and manual levels (1 and 8) are stored persistently.

This point is available in the **Points** folder of the control palette and usually resides in a device's **Points** folder.

Figure 1. Enum Writable property Sheet



You can access these properties in the **Property Sheet** by expanding **Station > Config > Drivers > Points** and double-click on **EnumWritable**

Property	Value	Description
Facets	Config Facets window	Defines a range of enumerated states (enum values).
Proxy Ext, Out	value	Reports the current enum value for the component. If no range is provided, status is considered normal and reports { ok }.
Proxy Ext, In1–16	values	Indicate from where the point's value originates, including details specific to the parentage of the point's network and communications (driver). null indicates that the point is an empty placeholder.
Proxy, Fallback	value	Configures a default Out value for a point when a priority scan of inputs In2 – In7 and In9 – In16 returns a null or invalid value, and no emergency action (In1) or manual action (In8) has occurred.
Override Expiration	read-only	Reports how long a value that has been set manually (using an action) remains valid.

Actions

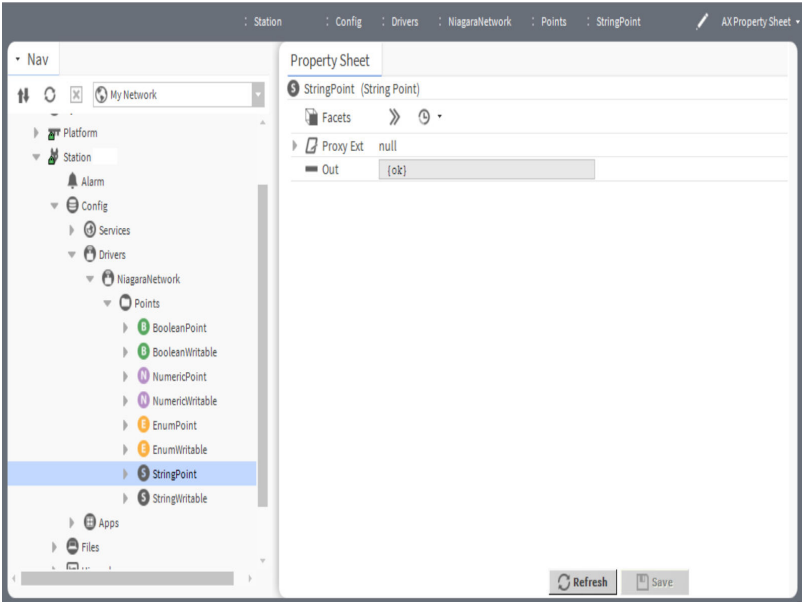
- **Emergency Override:** sets an output value at the emergency level.
- **Emergency Auto:** clears an output value set at the emergency level.
- **Override:** sets an output value at the manual level.
- **Auto:** clears an output value set at the manual level.
- **Set**
: opens the **Set** window where you can enter one of the range of values for this point.

Parent topic: [Components in control module](#)

This point is a basic read-only control point, with default slots: Facets, Out, and ProxyExt.

This point is available in the **Points** folder of the control palette and usually resides in a device's **Points** folder.

Figure 1. String Point Property Sheet



You can access these properties in the **Property Sheet** by expanding **Station > Config > Drivers > Points** and double-click on **StringPoint**.

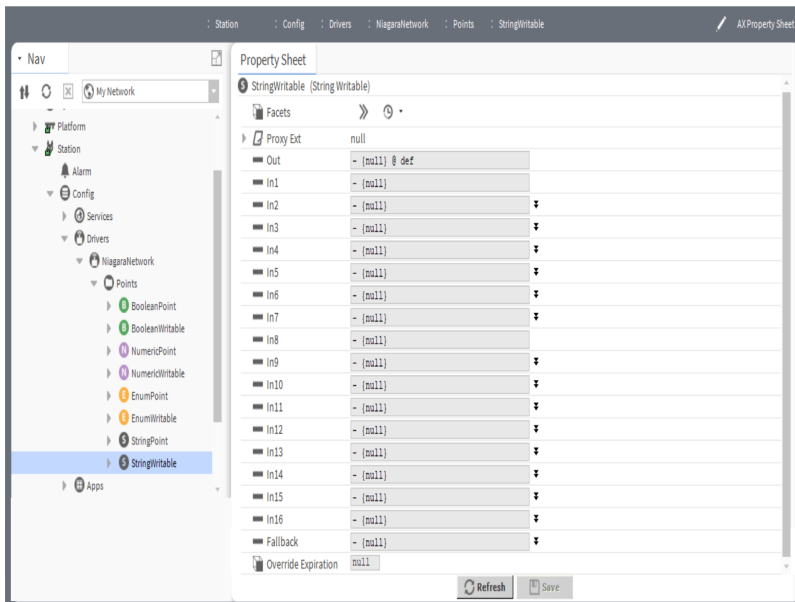
Property	Value	Description
Facets	Config Facets window	Defines a range of enumerated states (enum values).
Proxy Ext, Out	value	Reports both numeric and alpha values, facets and data status such as <code>fault</code> , <code>overridden</code> , <code>alarm</code> , and so on. If no status flag is set then status is considered normal and reports <code>{ok}</code>

Parent topic: [Components in control module](#)

StringWritable extends StringPoint to include 16 levels of command priority control. The highest priority active command is reflected in the Out property. Commands at the Manual and manual levels (1 and 8) are stored persistently.

This point is available in the **Points** folder of the control palette and usually resides in a device's **Points** folder.

Figure 1. String Writable Property Sheet



You can access these properties in the **Property Sheet** by expanding **Station > Config > Drivers > Points** and double-click on **StringWritable**

Property	Value	Description
Facets	Config Facets window	Associates a key with one or more phrases.
Proxy Ext, Out	value	Reports both numeric and alpha values, facets and data status such as <code>fault</code> , <code>overridden</code> , <code>alarm</code> , and so on. If no status flag is set then status is considered normal and reports <code>{ok}</code>
Proxy Ext, In1–16	values	Indicate from where the point's value originates, including details specific to the parentage of the point's network and communications (driver). <code>null</code> indicates that the point is an empty placeholder.
Proxy, Fallback	value	Configures a default Out value for a point when a priority scan of inputs In2 – In7 and In9 – In16 returns a null or invalid value, and no emergency action (In1) or manual action (In8) has occurred.
Override Expiration	read-only	Reports how long a value that has been set manually (using an action) remains valid.

Actions

- **Emergency Override:** sets an output value at the emergency level.

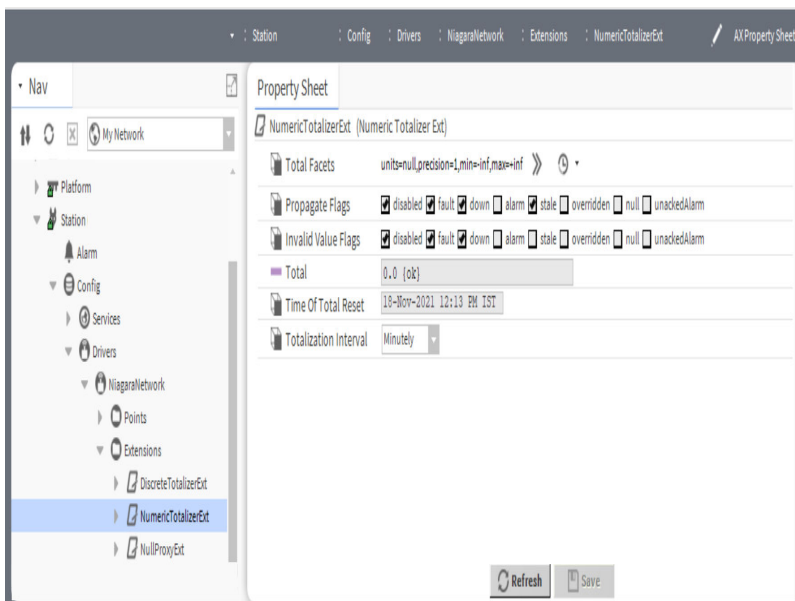
- **Emergency Auto:** clears an output value set at the emergency level.
- **Override:** sets an output value at the manual level.
- **Auto:** clears an output value set at the manual level.
- **Set**
: opens the **Set** window where you can enter a text string.

Parent topic: [Components in control module](#)

NumericTotalizerExt is a control point extension used for integrating a numeric point value over time. For example, a totalizer with a minutely totalization interval can convert an instantaneous flow reading in cubic feet per minute (cfm) into a value representing total cubic feet consumed. An available resetTotal action clears (zeroes) any accumulated total.

The NumericTotalizerExt is available in the Extensions folder of the **control** palette.

Figure 1. NumericTotalizerExt Property Sheet



You can access these properties in the **Property Sheet** by expanding **Station > Config > Drivers > NiagaraNetwork > Extensions** and double-click on **NumericTotalizerExt**.

Property	Value	Description
Total Facets	Config Facets window	Defines a range of enumerated states (enum values).
Propagate Flags	Check boxes	Defines which input status flags will be propagated from input to output.

Property	Value	Description
Invalid Value Flags	Check boxes	Defines which input status flags will denote invalid input values that should not be included in the total
Total	Value	Defines the total accumulated value. If no status flag is set then status is considered normal and reports {ok}
Time of Total Reset	read-only	Shows a date/timestamp of the last reset.
Totalization Interval	Minutely and Hourly	Describes the total interval of the accumulated values. When set to Minutely value occurs at every minute and when set to Hourly value occurs at every hour.

Actions

- **Timer Expired**
: The period of the duration has expired.
- **Reset Total**: Clears the total accumulated value to zero.

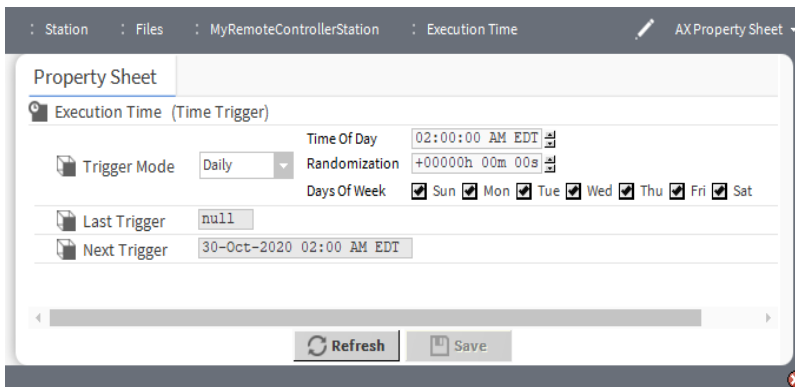
Parent topic: [Components in control module](#)

NullProxyExt is the default implement AbstractProxyExt used to indicate that a point is not a proxy point. The NullProxyExt is in the control palette's **Extensions** folder, but is already present by default on compatible point types. Presence indicates that you can add other types of extensions to the parent component, for example alarm or history extensions.

Parent topic: [Components in control module](#)

This component fires a topic at configured times. It is available in the Trigger folder of the **control** palette.

Figure 1. Example of time trigger properties from the niagaradriver module



Property	Value	Description
Trigger Mode	drop-down list	Determines when a TimeTrigger fires. Daily fires the trigger at a specific time on selected days of the week, and includes a randomized interval so that the trigger does not fire at exactly the same time every day. Randomized firing is useful when handling large volumes of data. Interval fires a trigger each time the specified interval elapses. You would use it to fire the trigger several times per day (for example, every 5 minutes). Manual requires a human to fire a trigger. When this trigger fires for a station database, a system indexer (LocalSystemIndexer or GlobalNiagaraNetworkIndexer) executes its index operation. CAUTION: Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.
Trigger Mode, Time of Day	hour	Configures either a specific time during the day or a time range with Start Time and End Time properties.
Trigger Mode, Randomization	hour minute second	Configures an event at random times based on the defined interval. Note: Some control time trigger components do not provide this property.
Trigger Mode, Days of the Week	check boxes (defaults to daily)	Specifies the days of the week to include.
Last Trigger	read-only	Reports when (by displaying a timestamp) the last trigger fired.
Next Trigger	read-only	Reports when the trigger is scheduled to fire again.

Actions

- **Fire Trigger:** Sets the time to fire the trigger on **Daily/Interval** basis.

Parent topic: [Components in control module](#)

- ApplicationFile

- `AudioFile`
- `BajadocFile`
- `BogFile`
- `BogScheme`
- `BogSpace`
- `CFile`
- `CssFile`
- `ExcelFile`
- `GifFile`
- `HtmlFile`
- `ImageFile`
- `JavaFile`
- `JpegFile`
- `NavFile`
- `PaletteFile`
- `PdfFile`
- `PngFile`
- `PowerPointFile`
- `PrintFile`
- `PxFile`
- `TextFile`
- `VideoFile`
- `VisioFile`
- `WordFile`
- `XmlFile`
- [**file-ApplicationFile**](#)
ApplicationFile stores an application file.
- [**file-AudioFile**](#)
AudioFile stores an audio file.

- **[file-BajadocFile](#)**
BajadocFile represents Bajadoc documentation. Bajadoc is a special file that can describe components in a database.
- **[file-BogFile](#)**
BogFile represents a BogFile in the file system of a session. A Bog File is a special file that can describe Components in a Database.
- **[file-BogScheme](#)**
BogScheme represents a BogScheme in the file system of a session. A Bog File is a special file that can describe Components in a Database.
- **[file-BogSpace](#)**
BogSpace represents a BogSpace in the file system of a session. A Bog File is a special file that can describe Components in a Database.
- **[file-CFile](#)**
CFile stores a c source file.
- **[file-CssFile](#)**
CssFile stores a CSS cascading style sheet.
- **[file-ExcelFile](#)**
ExcelFile stores a Microsoft Excel file.
- **[file-GifFile](#)**
GifFile stores a GIF image.
- **[file-HtmlFile](#)**
HtmlFile stores HTML markup.
- **[file-ImageFile](#)**
ImageFile stores an image.
- **[file-JavaFile](#)**
JavaFile stores a java source file.
- **[file-JpegFile](#)**
JpegFile stores a JPEG image.
- **[file-NavFile](#)**
NavFile stores XML nav markup.
- **[file-PaletteFile](#)**
This file is a Bog file with a different extension and icon. Many modules include a palette.
- **[file-PdfFile](#)**
PdfFile stores a Adobe PDF file.
- **[file-PngFile](#)**
PngFile stores a PNG image.
- **[file-PowerPointFile](#)**
PowerPointFile stores a Microsoft PowerPoint file.
- **[file-PrintFile](#)**
PrintFile stores XML print markup.
- **[file-PxFile](#)**
PxFile is just a Bog File file with a different extension and icon.
- **[file-TextFile](#)**
TextFile stores plain text.
- **[file-VideoFile](#)**
VideoFile stores an video file.
- **[file-VisioFile](#)**
VisioFile stores a Microsoft Visio file.
- **[file-WordFile](#)**
WordFile stores a Microsoft Word file.
- **[file-XmlFile](#)**
XmlFile stores an xml file.

Parent topic: [Component guides](#)

ApplicationFile stores an application file.

Parent topic: [Components in file module](#)

AudioFile stores an audio file.

Parent topic: [Components in file module](#)

BajadocFile represents Bajadoc documentation. Bajadoc is a special file that can describe components in a database.

Parent topic: [Components in file module](#)

BogFile represents a BogFile in the file system of a session. A Bog File is a special file that can describe Components in a Database.

Parent topic: [Components in file module](#)

BogScheme represents a BogScheme in the file system of a session. A Bog File is a special file that can describe Components in a Database.

Parent topic: [Components in file module](#)

BogSpace represents a BogSpace in the file system of a session. A Bog File is a special file that can describe Components in a Database.

Parent topic: [Components in file module](#)

CFile stores a c source file.

Parent topic: [Components in file module](#)

CssFile stores a CSS cascading style sheet.

Parent topic: [Components in file module](#)

ExcelFile stores a Microsoft Excel file.

Parent topic: [Components in file module](#)

GifFile stores a GIF image.

Parent topic: [Components in file module](#)

HtmlFile stores HTML markup.

Parent topic: [Components in file module](#)

ImageFile stores an image.

Parent topic: [Components in file module](#)

JavaFile stores a java source file.

Parent topic: [Components in file module](#)

JpegFile stores a JPEG image.

Parent topic: [Components in file module](#)

NavFile stores XML nav markup.

Parent topic: [Components in file module](#)

This file is a Bog file with a different extension and icon. Many modules include a palette.

Parent topic: [Components in file module](#)

PdfFile stores a Adobe PDF file.

Parent topic: [Components in file module](#)

PngFile stores a PNG image.

Parent topic: [Components in file module](#)

PowerPointFile stores a Microsoft PowerPoint file.

Parent topic: [Components in file module](#)

PrintFile stores XML print markup.

Parent topic: [Components in file module](#)

PxFile is just a Bog File file with a different extension and icon.

Parent topic: [Components in file module](#)

TextFile stores plain text.

Parent topic: [Components in file module](#)

VideoFile stores an video file.

Parent topic: [Components in file module](#)

VisioFile stores a Microsoft Visio file.

Parent topic: [Components in file module](#)

WordFile stores a Microsoft Word file.

Parent topic: [Components in file module](#)

XmlFile stores an xml file.

Parent topic: [Components in file module](#)

- BajadocOptions

- [help-BajadocOptions](#)

The BajadocOptions stores the options used by the BajadocViewer. The Bajadoc options allow you to change the following:

Parent topic: [Component guides](#)

The BajadocOptions stores the options used by the BajadocViewer. The Bajadoc options allow you to change the following:

- **Show Baja Only**
- **Flatten Inheritance**

These are stored under /user/{user}/bajadoc.options.

Parent topic: [Components in help module](#)

The net module contains the components listed below.

- **InternetAddress**
- **HttpProxyServer**
- [net-InternetAddress](#)
InternetAddress models an Internet address which is a composite of a hostname (or raw IP address) and a port number.
- [net-HttpProxyServer](#)
HttpProxyService is used to support connections to the Internet through a non-transparent proxy.

Parent topic: [Component guides](#)

InternetAddress models an Internet address which is a composite of a hostname (or raw IP address) and a port number.

Parent topic: [Components in net module](#)

HttpProxyService is used to support connections to the Internet through a non-transparent proxy.

All services that make use of the `bajax.baja.net.HttpConnection` class will automatically roll over to using the proxy server once the HttpProxyService has been configured and enabled. The Weather Service is one of the services that is affected by this feature.

To use this component you must:

- Open the net palette and drag **HttpProxyServer** to your station's **Services** folder.
- Configure the following properties in the **Property Sheet** view:

Property	Value	Description
Status	read-only	This display-only property displays the status of the Http Proxy Service.

Property	Value	Description
Fault Cause	read-only	This display-only property provides an error message that indicates the reason for a fault.
Enabled	true or false (default)	Enables and disabled the server.
Server	text	This property provides a text field for entering the address of the proxy server you are connecting to.
Port	number	Specifies the port number for communicating with the proxy server.
Exclusions	text	This text field allows you to designate addresses that are allowed to be contacted directly, therefore excluding them from having to be contacted through the proxy server. The default addresses in the field are typical addresses followed by a slash(/) and the subnet mask designation.
Authentication Scheme	drop-down list	This property provides the following two options: • None: no authentication is required at the proxy server. • Basic: basic authentication is required at the proxy server.
User	text	This is the login name to be used when authentication is set to Basic.
Password	text	This is the password text that is to be used when authentication is set to Basic.

Parent topic: [Components in net module](#)

- Program
- ProgramService
- ProgramModule
- Templates
- [program-Program](#)
Program uses an instance based class file to implement user defined component logic. The Program can be viewed and edited using the ProgramEditor.
- [program-ProgramService](#)
The **ProgramService** provides a (default) **Batch Editor** view to launch a batch operation on multiple selected component slots.

- [program-ProgramModule](#)
ProgramModule provides a **Program Module Builder** view used to build standard Niagara modules from Program components. This view provides a mechanism to version and provision Program modules just like any other Niagara modules.
- [Templates](#)
Templates folder is available in the program palette.

Parent topic: [Component guides](#)

Program uses an instance based class file to implement user defined component logic. The Program can be viewed and edited using the ProgramEditor.

Actions

Program is available in the program palette.

Execute — this function takes some input, does whatever the action should (including possibly calling sub action) and returns the action's output.

Parent topic: [Components in program module](#)

The **ProgramService** provides a (default) **Batch Editor** view to launch a batch operation on multiple selected component slots.

The **ProgramService** also provides a secondary **Robot Editor** view to create Robots, which are similar to Program components (written using Java code), but are not persisted in the station database.

The **ProgramService** is available in the program palette, but typically exists in most station's **Services** container, as it is included among default services when using the **New Station** tool (wizard) in Workbench.

Actions

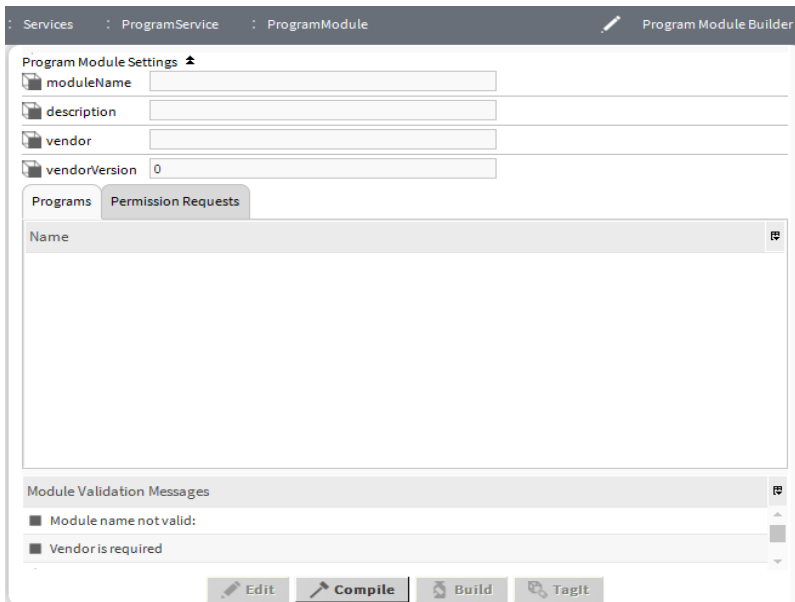
Run Robot — It is used to create robots.

Parent topic: [Components in program module](#)

ProgramModule provides a **Program Module Builder** view used to build standard Niagara modules from Program components. This view provides a mechanism to version and provision Program modules just like any other Niagara modules.

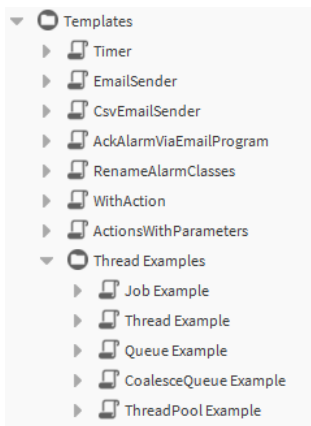
The ProgramModule is available in the program palette.

Figure 1. Program Module Builder view



Parent topic: [Components in program module](#)

Templates folder is available in the program palette.



Parent topic: [Components in program module](#)

- **WebService**
- **ClientEnvironment**
- **[WebService \(web-WebService\)](#)**
This service configures access to the HTTP server as well as the servlet infrastructure used to expose custom applications over HTTP. It is available in the **web** palette and is one of the default services in a station created by using the **New Station** tool. A station supports only one **WebService**.
- **[web-MobileClientEnvironment](#)**
MobileClientEnvironment (mobile) is a child of the **ClientEnvironments** container of a station's **WebService**, and present only if the host is licensed with the **mobile** feature. It

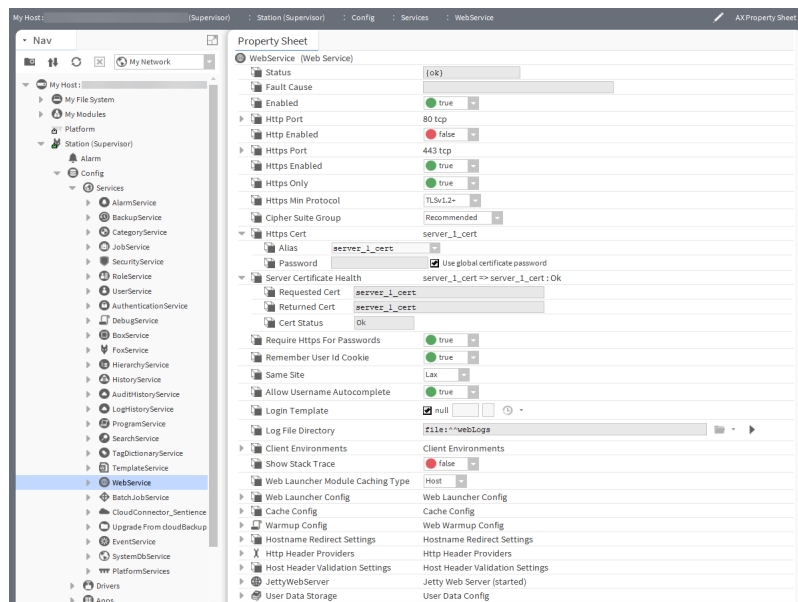
is used in the automatic selection of the appropriate webProfile type for a user, based on the detection of the incoming browser client type (e.g. desktop or mobile).

Parent topic: [Component guides](#)

This service configures access to the HTTP server as well as the servlet infrastructure used to expose custom applications over HTTP. It is available in the web palette and is one of the default services in a station created by using the **New Station** tool. A station supports only one **WebService**.

Note: Starting in Niagara 4.11, the **WebService** provides a TLSv1.3 option for the Https Min Protocol property. If you choose the TLSv1.3 setting, it is possible that **UxMedia** pages will not display correctly in a **Wb UxMedia** view. To work around this potential problem, you can choose to view the **UxMedia** page with a web browser or revert to the native JavaFx view in Workbench.

Figure 1. WebService properties



To access these properties, expand **Config > Services > WebService**.

In addition to the common properties (Status, Fault Cause and Enabled), these properties are unique to this component.

Property	Value	Description
Http Port	additional properties (port number defaults to 80)	Specifies the HTTP client's TCP port. The service listens on on this port for connections. The "Server Port (baja-ServerPort)" topic documents the additional properties.
Http Enabled	true (default) or false	Turns the processing of HTTP requests on (true) and off (false).

Property	Value	Description
		true turns on a standard Http connection (no communication security). When enabled, Fox Enabled in the FoxService must also be set to true (for Web Launcher use). false turns off the standard Http connection causing the system to ignore any attempts to connect using Http port 80. If Https Only is enabled, this setting (false for Http Enabled) is irrelevant.
Https Port	additional properties (port number defaults to 443)	Specifies the HTTPS client's TCP port. The service listens on this port for connections. The "Server Port (baja-ServerPort)" topic documents the additional properties.
Https Enabled	true (default) or false	Turns the processing of HTTPS requests on (true) and off (false). true turns on secure Http communication using port 443. When enabled, Foxs Enabled in the FoxService must also be set to true (for Web Launcher use). false turns off the secure Https connection causing the system to ignore any attempts to connect using Https port 443.
Https only	true (default) or false	Configures the security of the connection. true redirects any attempt to connect using a connection that is not secure (Http alone) to a secure Https connection. false permits an Http connection.
Https Min Protocol	drop-down list (defaults to TLSv1.2+)	Configures the minimum level of the TLS (Transport Layer Security) protocol to which the server accepts negotiations. Options include versions TLSv1.0+ TLSv1.1+, TLSv1.2+, and TLSv1.3. Note: TLSv1.0 and TLSv1.1 are still supported for backwards compatibility, but is recommended to use TLSv1.2 and higher. During the handshake, the server and client agree on which protocol to use. Change Protocol from the default if your network requires a specific version or if a future vulnerability is found in one of the versions.
Https Cert	text (read-only)	Displays the host platform's server certificate that is currently used.
Alias	drop-down list (defaults to default)	Specifies the alias of the host platform's server certificate, which the client uses to validate server authenticity. The default identifies a self-signed certificate that is automatically created when you initially log on to the server. It cannot be deleted and should be used for recovery purposes. The default certificate is protected by the global certificate password. If other certificates are in the host platform's key store, you can select them from the drop-down list.
Password	text and check box	Defines a the user-defined password or the global certificate password associated with the server certificate. This unique password is required as of Niagara 4.13.

Property	Value	Description
Server Certificate Health	text (read-only)	Displays the alias of the used server certificate and its status (OK, Bad Password)
Requested Cert	text (read-only)	Displays the certificate's alias that was requested.
Returned Cert	text (read-only)	Displays the actual certificate that is currently used.
Cert Status	text (read-only)	Specifies the status of the requested certificate (OK, Bad Password)
Require Https For Passwords	true (default) or false	Controls the protocol used when a user creates a new password. true prevents the creation of a password for a new user across a connection that is not secure. HTTPs Enabled also must be set to true or the system disables the New button (for creating a new user in the UserService). false leaves the New button (for creating a new user in the UserService) enabled even if HTTPs Enabled is false. This combination of settings creates a security vulnerability when creating passwords for new users and is not recommended.
Remember User Id Cookie	true (default) or false	Controls if the system sets the cookie in the user's browser. true sets the niagara_userid cookie with the user's username in the user browser. false, the niagara_userid cookie is not used at all.
Same Site	None, Lax (default), or Strict	Configures browser behavior for HTTP cookies. Lax allows cookies to be sent automatically only in a first-party context and with HTTP GET requests. Cookies will be withheld on cross-site sub-requests. None allows cookies to be sent with HTTP requests to third party sites, without restriction. Strict allows cookies to be sent only for first-party context requests (requests originating from the site that set the cookie). Note: The Strict setting is not supported when using SAML Authentication.
Allow Username Autocomplete	true (default) or false	Enables (true) and disables (false) autocomplete browser behavior on the username field of the prelogin page.
Login Template	check box and chooser (defaults to null)	Determines if a login template is used. Any selects no custom login template. When Any is not selected, the option list shows available custom login templates that you can select for a station login page.
Log File directory	filepath (defaults to file:^^webLogs)	Defines the folder in the station's file space in which log files are stored. Log file names use a YYMMDD.log (date) convention, such as 230501.log for a file created May 1, 2023.
Client Environments	additional properties	Serves as a container for Mobile Client Environment (mobile) entries, which are available if the station's host is licensed with the mobile feature. It is used in detection of a user's browser type (for example,

Property	Value	Description
		desktop or mobile) and the selection of the appropriate webProfile for the user. The topic “Client Environments (web-ClientEnviroments” documents this container. The <i>Niagara Mobile Guide</i> documents the properties for the mobile environment.
Show Stack Trace	true or false (default)	Controls if exception stack traces, when available, appear in error responses. true shows exception stack traces in error responses when they are available. false disables exception stack traces in error responses.
Load JxBrowser from Cloud	drop-down list	Loads the JxBrowser from the cloud.
Web Launcher Module Caching Type	Host (default) or User	Determines how a client using the Web Launcher caches modules. Host results in a folder and the downloading of installation modules to the module folder (n4applet for N4, and wbapplet for AX). This results in the creation of multiple folders of downloaded modules, which negatively affects platform memory usage. User results in the creation of a .sharedModuleCache folder (one cache per host visited; one shared cache per user). The system then downloads to a sub-folder at this location (n4applet for N4, and wbapplet for AX). This option minimizes the memory required when running in a controller.
Web Launcher Config, Web Launcher Enabled	true (default) or false	Provides a container for several sub-properties used to configure aspects of Web Launcher, which provides an applet-like Workbench environment that runs completely outside of a web browser.
Cache Config, Enabled	true (default) or false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Cache Config, Cached File Extensions	text (defaults to png,jpg,gif,svg)	Sets the desired file type(s) to configure cache memory, which caches all station home image files in the web browser. png, jpg, gif, svg caches only files with these extensions. * caches all file types without re-validation.
WarmupConfig	additional properties	Speeds the loading of HxPx graphics. This property was new in Niagara. The first time an HxPx loads, it takes time because Java code also runs for the first time. It does not have complete compiled methods, which it needs to download fast. This component overcomes this problem. The topic, “Warmup Config (Web Warmup Config)” documents the additional properties.
Hostname Redirect Settings, Redirect to Hostname	true or false	Controls the URL used to connect to a station. true redirects to the mentioned host station in the host name field, when browsed. false denies the redirection to the mentioned host station in the host name property, when browsed. When you open the browser and enter the local station name or IP address of the local host in the URL, the host redirects to the station mentioned in the host name field and displays the station name in the URL.

Property	Value	Description
Hostname Redirect Settings, Hostname	text string	Defines the host name.
Http Header Providers	additional properties	Configure HTTP Header provider properties in Niagara. The topic “Http Header Providers (Http Header Providers)” documents the additional properties.
JettyWebServer	additional properties	Configures the Jetty web server. “Jetty Web Server (jetty-JettyWebServer)” documents the additional properties.
User Data Storage	additional property	Provides an option to store user data. The topic, “User Data Config (web-UserDataConfig)” documents the single property.

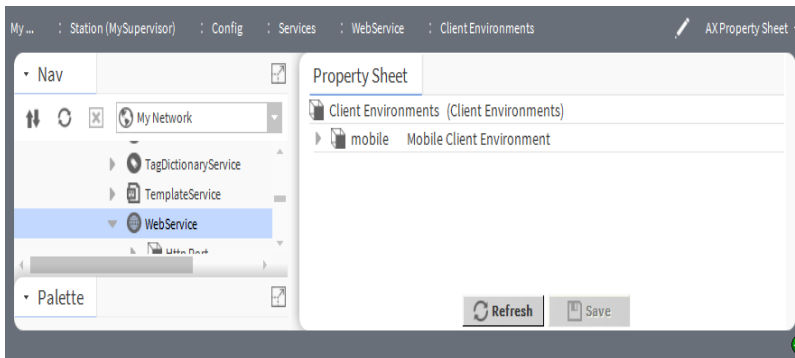
- **[Client Environments \(web-ClientEnvironments\)](#)**
This component serves as a container for environments, such as mobile.
- **[Warmup Config \(web-WebWarmupConfig\)](#)**
These properties configure what happens when a controller restarts. You can use them to speed up the loading time of the first page for Hx profiles. Once you have run an HxPx profile in a station, it records that you are an HxPx user, which enables Warmup Config by default the next time you restart the controller.
- **[Http Header Providers \(web-HttpHeaderProviders\)](#)**
These headers pass additional information with an HTTP request or response between client and server. This information ensures the authenticity of the messages providing security against click-jacking and other threats. This component contains four headers that you may customize as needed. To ensure the most robust security, leave all headers enabled. To turn off a header, if necessary, set its Enabled property to false.
- **[Host Header Validation Settings \(web-HostHeaderValidationSettings\)](#)**
This component controls host header validation.
- **[Csp Header Provider \(web-CspHeaderProvider\)](#)**
this header configures a typical HTML5 HX profile. You may add additional sources to these directives, but removing any of the default sources may cause your views to stop working.
- **[X-Content-Type-Options \(web-XContentTypeOptionsHeaderProvider\)](#)**
This component configures options for X-Content.
- **[X Frame Options Header Provider \(web-XFrameOptionsHeaderProvider\)](#)**
This component supports the X Frame Options.
- **[X Xss Protection Header Provider \(web-XXssProtectionHeaderProvider\)](#)**
This component configures the protection header provider.
- **[Jetty Web Server \(jetty-JettyWebServer\)](#)**
These properties configure the Jetty Web Server.
- **[User Data Config \(web-UserDataConfig\)](#)**
This component provides an option to store user data.

Parent topic: [Components in web module](#)

This component serves as a container for environments, such as mobile.

This component has no unique properties of its own. For information about the specific environments, refer to the topic for each environment. For example, “web-MobileClientEnvironment” documents the mobile properties.

Figure 1. Client Environments



To access this **Property Sheet**, expand **Config > Services**, double-click **WebService** and click **Client Environments**.

Parent topic: [WebService \(web-WebService\)](#)

These properties configure what happens when a controller restarts. You can use them to speed up the loading time of the first page for Hx profiles. Once you have run an HxPx profile in a station, it records that you are an HxPx user, which enables Warmup Config by default the next time you restart the controller.

If you are not using Warmup Config or HxPx, turn it off by setting Enabled to No.

Note: Warmup Config does not run until and unless you set Enabled to Yes.

Figure 1. Warmup Config properties

Property Sheet	
Warmup Config (Web Warmup Config)	
Status	{disabled}
Fault Cause	
Enabled	No
Running	☐ false
Interrupted	☐ false
User Activity Found	☐ false
Recent User Activity Found	☐ false
Last Warmup Time	00000h 00m 00.000s
Hx App Warmup	Hx App Warmup
Hx Px Warmup	Hx Px Warmup

To access this component, double-click **Services > WebServices > Warmup Config**.

Refer to separate topic.

In addition to the common properties (Status and Fault Cause), these properties are unique to this component.

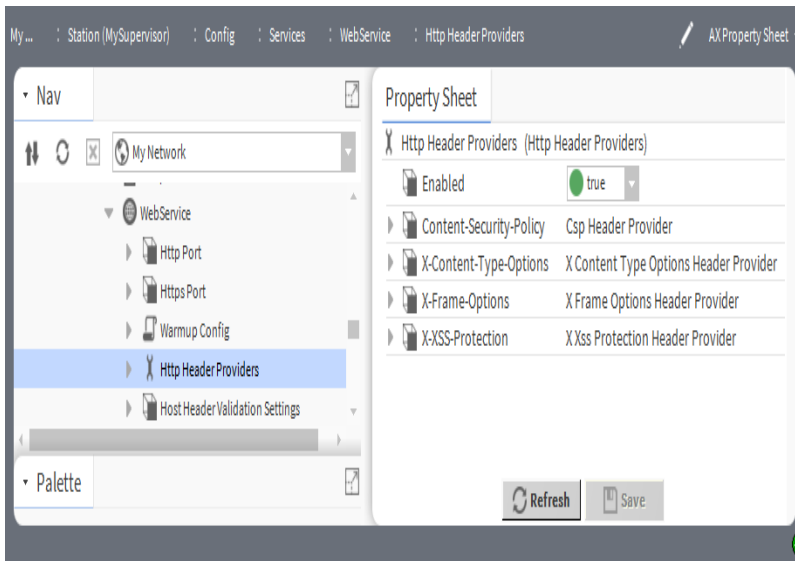
Property	Value	Description
Enabled	Yes, No and When User Activity Found (default)	Controls when a user starts to use the HxPx feature.
Running	true or false (default)	Configures whether or not the WebWarmup is currently running. This is usually only true just after a station starts unless it ran manually with the Warmup action. When a station runs by default it is true unless the station runs manually with the Warmup action.
Interrupted	true or false (default)	Controls a station interrupt. This is usually false and only true in two cases: • If WebWarmup is running and a user logs in when it is running. • If WebWarmup is running and the WebWarmup config action called Cancel Warmup runs.
User Activity Found	true or false (default)	Controls if a warmup runs after a station reboot. For the Warmup process, this property is true when a user uses a framework feature. For example for the HxPxWarmup this property is true when a user loads an HxPx file from a browser. Once it is true, it stays true even after a restart. This allows the Warmup to run after a reboot. For HxPxAppWarmup, this property is true for any WebActivity.
Recent User Activity Found	true or false (default)	Functions the same as User Activity Found except after a restart when it defaults to false. It is a flagged transient.
Last Warmup Time	read-only	Displays the time of the last warmup.

Parent topic: [WebService \(web-WebService\)](#)

These headers pass additional information with an HTTP request or response between client and server. This information ensures the authenticity of the messages providing security against click-jacking and other threats. This component contains four headers that you may customize as needed. To ensure the most robust security, leave all headers enabled. To turn off a header, if necessary, set its Enabled property to false.

For complete details on headers, see the MDN web docs site (<https://developer.mozilla.org/en-US/docs/Web/API/Headers>).

Figure 1. Http Header Providers properties



To access these properties, expand **Services > WebServices** and double-click **Http Header Providers**.

In addition to the standard property, Enabled, these properties configure header providers.

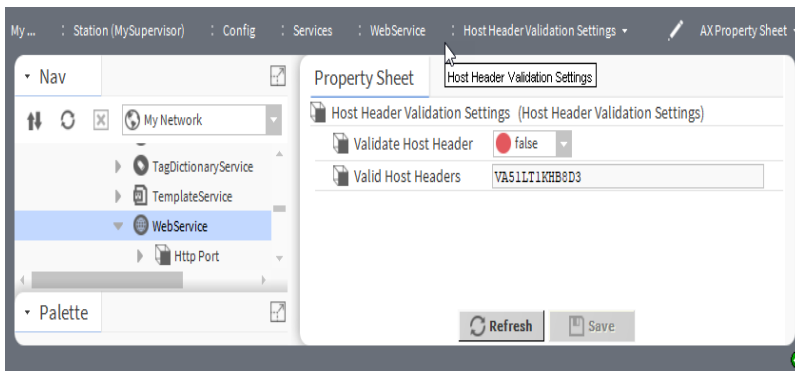
Property	Value	Description
Content-Security-Policy	additional properties	Notifies the browser what restrictions should be put on images, JavaScript, or CSS, in response to a request for resources. "Csp Header Provider (web-CspHeaderProvider)" documents the additional properties.
X-Content-Type-Options	drop-down list (defaults to nosniff)	Indicates to browsers that they should apply additional restrictions to auto-detect content types in downloaded files. For best security, nosniff is the recommended value.
X Frame Options	drop-down list (defaults to Sameorigin)	Indicates if a browser should be allowed to render pages served by your station in a <frame> or <iframe> of another site. Use it to avoid click-jacking attacks. Sameorigin allows the browser to embed other pages from within the same station. This is considered a safe practice and is necessary for the correct functioning of the HTML5 Hx Profile. Deny prevents the browser from loading the page in a frame.

Property	Value	Description
		Note: Deny inhibits the display of some typical HTML5 Hx Profile views. Any may cause a Cross-Frame Scripting (XFS) or click-jacking vulnerability and is not recommended. If an external site needs to embed your station's web interface, configure a "frame-ancestors" directive under Content-Security-Policy.
X-XSS-Protection	text (defaults to 1; mode=block)	Ensures that, if an XSS attack is detected, the browser prevents the page from loading. 1; mode=block is the recommended value.

Parent topic: [WebService \(web-WebService\)](#)

This component controls host header validation.

Figure 1. Host Header Validation Settings properties



To access this **Property Sheet**, expand **Config > Services**, double-click **WebService** and click **Host Header Validation Settings**.

Property	Value	Description
Validate Host Header	true or false (default)	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Valid Host Headers	text	Defines the host headers.

Parent topic: [WebService \(web-WebService\)](#)

this header configures a typical HTML5 HX profile. You may add additional sources to these directives, but removing any of the default sources may cause your views to stop working.

<https://content-security-policy.com/> provides additional information on the Content-Security-Policy HTTP response header.

The Security Dashboard provides information about the HTTP Header configuration and whether there is any performance degradation. It provides notification for any non-secure headers and explains why the settings are not secure. To secure the header's settings, set the values as described in the properties table of the “WebService (web-WebService)” topic.

Figure 1. Content-Security-Policy properties

Property Sheet

Content-Security-Policy (Csp Header Provider)

Enabled	true
Status	{ok}
Violation Text	
child-src	
connect-src	'self' workbench ws://%hostname%:%port%
default-src	'self' workbench
frame-src	
font-src	
img-src	'self' workbench data:
manifest-src	
media-src	
object-src	
report-uri	/csp-reports
script-src	'self' workbench 'unsafe-inline' 'unsafe-
style-src	'self' workbench 'unsafe-inline'

Additional Directives

Refresh **Save**

To access this **Property Sheet**, expand **Config > Services > WebService > Http Header Providers** and double-click **Content-Security-Policy**.

The term 'self' in the various -src (source) properties instructs the software to load resources from the same origin, that is, the same scheme, host and part.

Note: The host workbench in the properties above allows HTML views, such as Web Chart to correctly function in Workbench and should not be removed under normal circumstances.

In addition to the standard properties (Enabled and Status), these properties are unique to this component.

Property	Default value	Description
Violation Text	text	Creates the text to display when a browser reports a Content-Security-Policy violation to a station, which logs it in the web.reporting.csp log. The station logs the first violation with SEVERE priority, and subsequent violations as FINE. Note: A Content-Security-Policy violation should not typically occur during normal usage of the system. If you receive one, consider whether your Content-Security-Policy configuration should be changed to

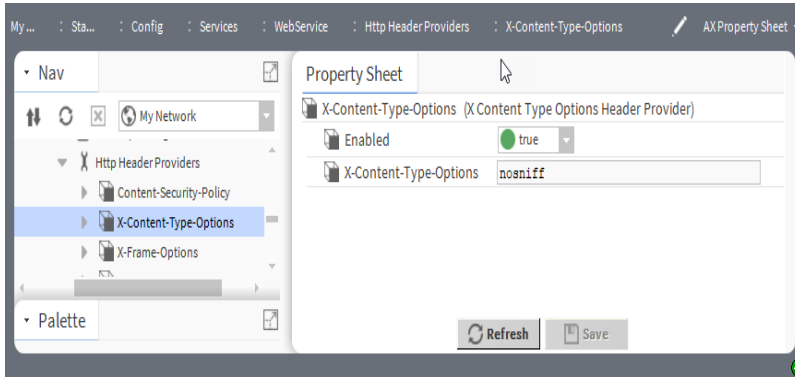
Property	Default value	Description
		match browser behavior or if the violation represents an attempted XSS attack.
child-src	text	Defines the valid sources for web workers and nested browsing contexts loaded using elements, such as <frame> or <iframe>.
connect-src	text (defaults to 'self' workbench ws://%hostname%:%port% wss://%hostname%:%port%)	Restricts the URLs that can be loaded using script interfaces. You can set up a template so that all Content-Security-Policy directives reference the %scheme%, %hostname%, and %port% from the originating HTTP request. Note: When viewing HTML views in Workbench, this request is made to Workbench. Content-Security-Policy headers include this by default. Removing it may cause HTML views to stop working in Workbench.
default-src	text (defaults to 'self' workbench)	Serves as a fallback for the other fetch directives.
frame-src	text	Specifies valid sources for nested browsing contexts loading using elements such as <frame> or <iframe>.
font-src	text	Specifies valid sources for fonts loaded using @font face.
img-src	text (defaults to 'self' workbench data:)	Specifies valid sources of images and favicons.
manifest-src	text	Specifies valid sources of application manifest files.
media-src	text	Specifies valid sources for loading media using the <audio>, <video> and <track> elements.
object-src	text	Specifies valid sources for the <object>, <embed>, and <applet> elements.
report-uri	text (defaults to /csp-reports)	Instructs the user agent to report attempts to violate the Content Security Policy. These violation reports consist of JSON documents sent via an HTTP POST request to the specified URI.
script-src	text (defaults to 'self' workbench 'unsafe-inline' 'unsafe-eval')	Specifies valid sources for JavaScript.
style-src	text (defaults to 'self' workbench 'unsafe-inline')	Specifies valid sources for stylesheets.

Property	Default value	Description
Additional Directives	text	Provides a location to enter any Content-Security-Policy directives not covered by the other properties on this component.

Parent topic: [WebService \(web-WebService\)](#)

This component configures options for X-Content.

Figure 1. X-Content-Type-Options properties



To access these properties, expand **Config > Services > WebService > Http Header Providers** and double-click **X-Content-Type-Options**.

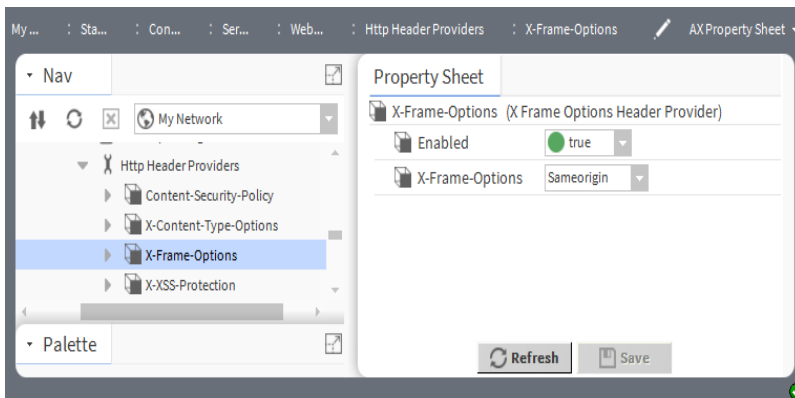
In addition to the standard property, Enabled, this property supports this component.

Property	Value	Description
X-Content-Type-Options	text	Defines the type options.

Parent topic: [WebService \(web-WebService\)](#)

This component supports the X Frame Options.

Figure 1. X Frame Options Header Provider properties



To access these properties, expand **Config > Services > WebService > Http Header Providers** and double-click **X-Frame-Options**.

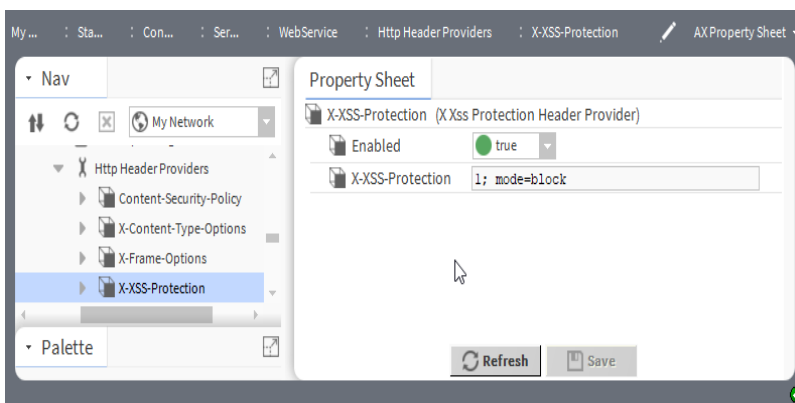
In addition to the standard property, Enabled, this property supports this component.

Property	Value	Description
X-Frame-Options	drop-down list (defaults to Sameorigin)	Selects the options. Deny prevents the browser from loading the page in a frame. Note: Deny inhibits the display of some typical HTML5 Hx Profile views. Sameorigin allows the browser to embed other pages from within the same station. This is considered a safe practice and is necessary for the correct functioning of the HTML5 Hx Profile. Any may cause a Cross-Frame Scripting (XFS) or click-jacking vulnerability and is not recommended. If an external site needs to embed your station's web interface, configure a "frame-ancestors" directive under Content-Security-Policy.

Parent topic: [WebService \(web-WebService\)](#)

This component configures the protection header provider.

Figure 1. X Xss Protection Header Provider property



To access these properties, expand **Config > Services > WebService > Http Header Providers** and double-click **X-XSS-Protection**.

In addition to the standard property, Enabled, this property supports this component.

Property	Value	Description
X-XSS-Protection	text	Defines the protection code. 1; mode=block is the recommended value.

Parent topic: [WebService \(web-WebService\)](#)

These properties configure the Jetty Web Server.

Figure 1. Jetty Web Server properties

The screenshot shows the 'Property Sheet' for 'JettyWebServer (Jetty Web Server)'. It lists several properties with their current values and ranges:

- Server State:** started
- Min Threads:** 8 [8-30]
- Max Threads:** 30 [14-max]
- Thread Idle Timeout:** 00000h 05m 00s [1 second - +inf]
- Denial Of Service Settings:** Jetty Do S Filter
- Quality Of Service Settings:** Jetty Qo S Filter
- Connection Limit:** Jetty Connection Limit
- Accept Rate Limit:** Jetty Accept Rate Limit
- NCSA Log:** N C S A Request Log
- Diagnostics Enabled:** true

To access these properties, expand **Config > Services > WebService** and double-click **JettyWebServer**.

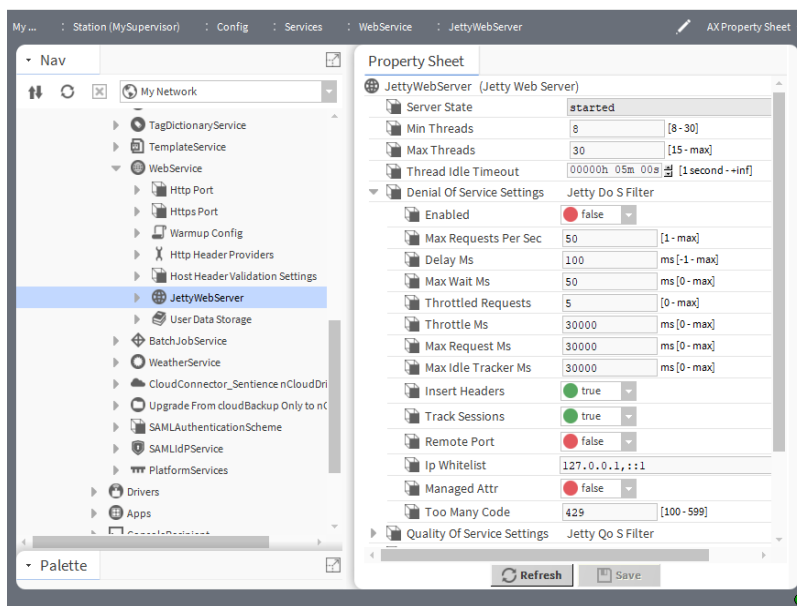
Property	Value	Description
Server State	read-only	Displays the state of the server.
Min Threads	number (defaults to 4)	Specifies the minimum number, concurrent connections (threads) that the station makes with the server.
Max Threads	number (defaults to 30)	Specifies the maximum number of multiple, concurrent connections (threads) that the station makes with the server.
Denial Of Service Settings	additional properties	Keeps track of the number of requests from a connection per second. If a limit is exceeded, the request is either rejected, delayed, or throttled. For property descriptions, refer to Denial of Service Settings .

Property	Value	Description
Quality Of Service Settings	additional properties	Limits the number of active requests to the number set by the "maxRequests" in it parameter (default 10). If more requests receives, it suspends and placed on priority queues. For property descriptions, refer to Quality of Service Settings
Connection Limit	additional properties	Defines a connection limit. For property descriptions, refer to Connection Limit
Accept Rate Limit	additional properties	Defines an accept-rate limit. For property descriptions, refer to Accept Rate Limit
N C S A Log	additional properties	For property descriptions, refer to N C S A Log .

Denial of Service Settings

These properties configure how the server responds to denial of service.

Figure 2. Denial of Service Settings properties



To access these properties, expand **Config > Services > WebService**, double-click **JettyWebServer** and expand **Denial of Service Settings**.

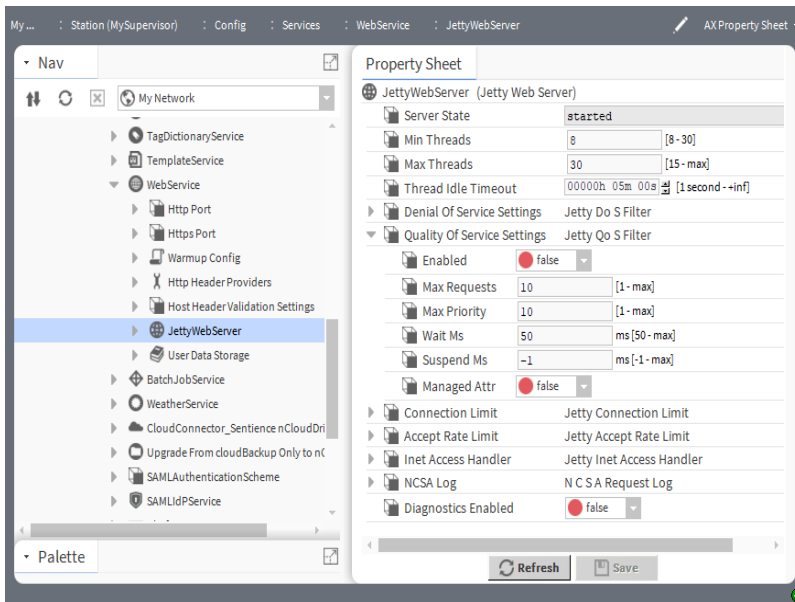
In addition to the standard property, Enabled, these properties support denial of service settings.

Property	Value	Description
Max Requests Per Sec	number (defaults to 50)	Defines the maximum numbers of requests from the server per second.
Delay Ms	milliseconds (defaults to 100)	Defines the number of milliseconds of delay to allow.
Max Wait Ms	milliseconds (defaults to 50)	Defines the maximum number of milliseconds to wait.
Throttled Requests	number	
Throttle Ms	milliseconds (defaults to 30000)	
Max Request Ms	milliseconds (defaults to 30000)	
Max Idle Tracker Ms	milliseconds (defaults to 30000)	
Insert Headers	true (default) or false	Enables (true) and disables (false) insert headers.
Track Sessions	true (default) or false	Enables (true) and disables (false) the tracking of sessions.
Remote Port	true or false (default)	Enables (true) and disables (false) the use of a remote port.
Ip Whitelist	numbers	Identifies the IP address of the whitelist.
Managed Attr	true or false (default)	Enables (true) and disables (false) managed attributes.
Too Many Code	number (defaults to 429)	

Quality of Service Settings

These properties configure how the server provides service.

Figure 3. Quality of Service Settings properties



To access these properties, expand **Config > Services > WebService**, double-click **JettyWebServer** and expand **Quality of Service Settings**.

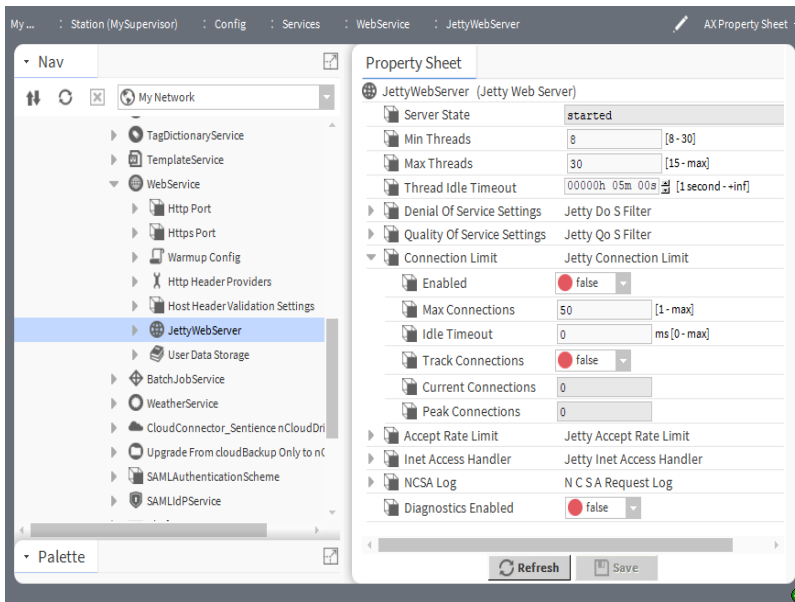
In addition to the standard property, Enabled, these properties support quality of service settings.

Property	Value	Description
Max Requests	number (defaults to 10)	
Max Priority	number (defaults to 10)	
Wait Ms	milliseconds (defaults to 50)	
Suspend Ms	milliseconds (defaults to -1)	
Managed Attr	true or false (default)	

Connection Limit

These properties configure connection limits.

Figure 4. Connection Limit properties



To access these properties, expand **Config > Services > WebService**, double-click **JettyWebServer** and expand **Connection Limit**.

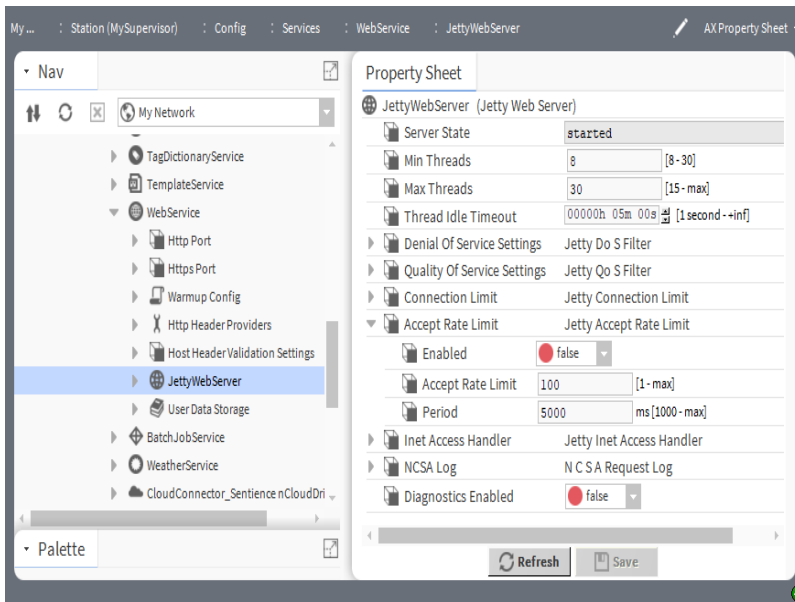
In addition to the standard property, Enabled, these properties support connection limit properties.

Property	Value	Description
Max Connections	number (defaults to 50)	
Idle Timeout	milliseconds (defaults to 0)	
Track Connections	true or false (default)	
Current Connections	read-only	Reports the number of current connections.
Peak Connections	read-only	Reports the maximum number of connections.

Accept Rate Limit

These properties configure the accept rate limit.

Figure 5. Accept Rate Limit properties



To access these properties, expand **Config > Services > WebService**, double-click **JettyWebServer** and expand **Accept Rate Limit**.

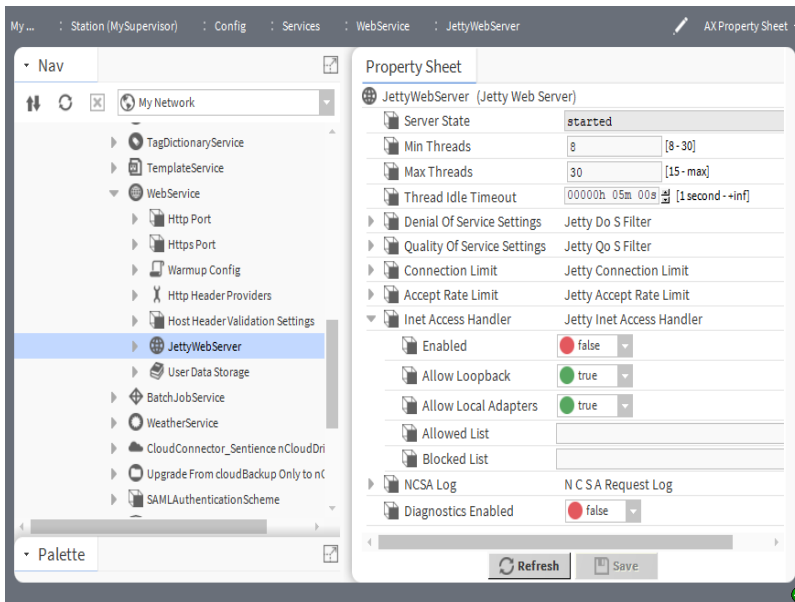
In addition to the standard property, Enabled, these properties support accept rate limit properties.

Property	Value	Description
Accept Rate Limit	number (defaults to 100)	
Period	milliseconds (defaults to 5000)	

Inet Access Handler

These properties configure the inet access handler.

Figure 6. Inet Access Handler properties



To access these properties, expand **Config > Services > WebService**, double-click **JettyWebServer** and expand **Inet Access Handler**.

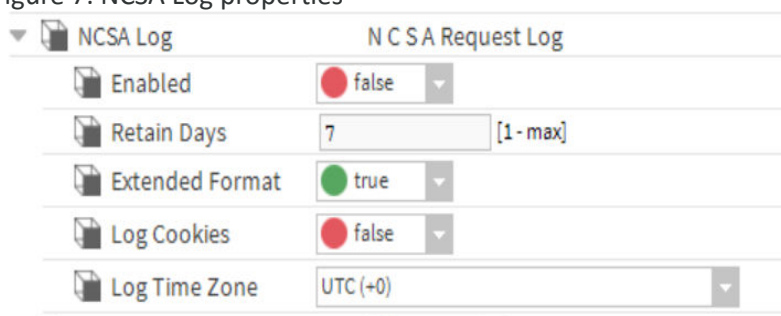
In addition to the standard property, Enabled, these properties support inet access handler properties.

Property	Value	Description
Allow Loopback	true (default) or false	
Allow Local Adapters	true (default) or false	
Allowed List	text	
Blocked List	text	

NCSA Log

This is a common format for a standardized text file that web servers use to keep track of processed requests.

Figure 7. NCSA Log properties



To access these properties, expand **Config > Services > WebService**, double-click **JettyWebServer** and expand **NCSA Log**.

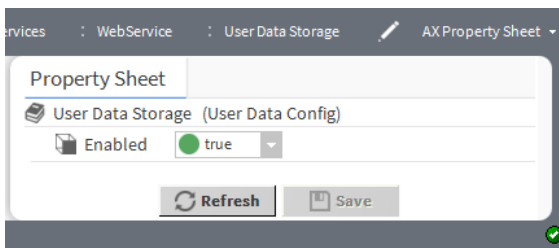
In addition to the common property (Enabled,,), these properties are unique to this component.

Property	Value	Description
Retain Days	number (defaults to 7)	Limits the size of the log by defining how many days to save log information.
Extended Format	true (default) or false	Extends the format of a standardized text file.
Log Cookies	true or false (default)	Logs the cookies of processed requests.
Log Time Zone	drop-down list	Identifies the time zone to use for time stamps.

Parent topic: [WebService \(web-WebService\)](#)

This component provides an option to store user data.

Figure 1. User Data Storage property



Property	Value	Description
Enabled	true (default), or false	Allows web apps to store user-specific data (that is, user options for the HTML5 Alarm Console) in the userdata folder of the station's file system. Typically, you leave this property enabled. A user with admin privileges can disable this property to clear user data.

Parent topic: [WebService \(web-WebService\)](#)

MobileClientEnvironment (mobile) is a child of the ClientEnvironments container of a station's **WebService**, and present only if the host is licensed with the **mobile** feature. It is used in the automatic selection of the appropriate webProfile type for a user, based on the detection of the incoming browser client type (e.g. desktop or mobile).

This container slot allows the station to automatically detect the user agent of an incoming client and use the appropriate Web Profile for the user:

- Default Web Profile if using a Java-enabled device, such as a PC

- Mobile Web Profile if using a mobile device, such as a cell phone or tablet

Refer to separate topic and *Niagara Mobile Guide*

Figure 1. Client Environments properties

The screenshot shows a 'Property Sheet' window. Under the 'Client Environments (Client Environments)' section, the 'mobile' sub-section is expanded, showing 'Mobile Client Environment' properties. The properties are:

- Enabled:** A dropdown menu with a green circle icon and the value 'true'.
- Status:** A text input field containing '{ok}'.
- Fault Cause:** An empty text input field.
- User Agent Pattern:** A text input field containing 'mobile|android|playbook|touch'.

At the bottom of the sheet are two buttons: 'Refresh' (with a circular arrow icon) and 'Save' (with a floppy disk icon).

To access the above properties, expand **Client Environments > mobile**.

In addition to the common properties (Enabled, Status and Fault Cause), this property is unique to this component.

Property	Value	Description
User Agent Pattern	text separated by the pipe symbol ()	Provides a list of one or more user agents that identify the target display types. The pipe symbol separates agents.

Parent topic: [Components in web module](#)

The following topics describe components that are in the workbench module.

- [workbench-KioskService](#)
workbench-KioskService is used to enable the Kiosk mode. Kiosk mode provides a way to run a Workbench station so that it fulfills many of the needs of a stand-alone operator workstation as well as many needs of a touchscreen interface. In order to automatically start Kiosk mode, you must have a locally connected display.
- [workbench-WbFieldEditorBinding](#)
WbFieldEditorBinding is used to bind WbFieldEditors to an object. It allows any existing WbFieldEditor (BooleanFE, EnumFE, AbsTimeFE, etc) to be used in a PX presentation.
- [workbench-WbViewBinding](#)
WbViewBinding is used to bind WbViews to an object. It allows any existing WbView (PropertySheet, WireSheet, manager view, etc.) to be used in a PX presentation.
- [workbench-WsOptions](#)
These are stored under /user/{user}/wiresheet.options.
- [workbench-WebBrowser](#)
This component can be added to a Px page to expose an external website (e.g. www.google.com) inside Workbench in a **Web Browser View**. The component properties allow you to configure browser display details for the referenced site.
- [workbench-WebWidget](#)
This is a baux, HTML5-based application that incorporates a view with interactive

functionality which allows you to edit properties and invoke commands from the view. You can easily add data to a WebWidget, such as the WebChart or Dashboard, simply by dragging one or more components onto the widget. The widget renders in both Workbench and HTML5 Hx interfaces. The widget also integrates into the environment. For example, commands defined for a WebWidget render as added tool bar icons in Workbench, as well as in the HTML5 Hx profile in a web browser.

Parent topic: [Component guides](#)

workbench-KioskService is used to enable the Kiosk mode. Kiosk mode provides a way to run a Workbench station so that it fulfills many of the needs of a stand-alone operator workstation as well as many needs of a touchscreen interface. In order to automatically start Kiosk mode, you must have a locally connected display.

Note: The Kiosk profile is not invoked or displayed by remote browser clients. You cannot use a remote computer with a touchscreen and expect to get the same Kiosk interface.

This component is located in the Workbench palette. To set a station to Kiosk mode, add the service to the station's **Services** container and set the enabled property to true. While Kiosk mode is not limited to touchscreen applications, it does provide advantages that make it well suited for use in a touchscreen application.

Parent topic: [Components in workbench module](#)

WbFieldEditorBinding is used to bind WbFieldEditors to an object. It allows any existing WbFieldEditor (BooleanFE, EnumFE, AbsTimeFE, etc) to be used in a PX presentation.

Parent topic: [Components in workbench module](#)

WbViewBinding is used to bind WbViews to an object. It allows any existing WbView (PropertySheet, WireSheet, manager view, etc.) to be used in a PX presentation.

Parent topic: [Components in workbench module](#)

These are stored under /user/{user}/wiresheet.options.

The WireSheet options allow you to view and change the following:

- Show Thumbnail
- Show Grid
- Show Status Colors

Parent topic: [Components in workbench module](#)

This component can be added to a Px page to expose an external website (e.g. www.google.com) inside Workbench in a **Web Browser View**. The component properties allow you to configure browser display details for the referenced site.

Figure 1. Web Browser properties

WebBrowser (Web Browser)

Visible: ☒ true

Enabled: ☒ true

Layout:

	Value	Units	
X	0.00	Abs	
Y	0.00	Abs	
Width	640.00	Abs	
Height	480.00	Abs	

☐ Fill

Title:

Ord:

Location:

Progress:

Progress Running: ☒ false

Show Progress Indicator: ☒ false

Context Menu Enabled: ☒ true

impl: Jx Web Browser Impl

This component is located in the workbench palette. Double-click the **WebBrowser** component to open the Property Sheet view.

In addition to the standard properties (Enabled and Layout) the following properties are present for this component:

Property	Value	Description
Visible	true (default) or false	Determines if this object is visible or hidden.
Title	text	Creates a name for the view.
Ord	ORD	Identifies the location of the content to display in this view.
Location	read-only	
Progress	read-only	
Progress Running	read-only	
Show Progress Indicator	true or false (default)	Turns the progress indicator on and off.
Context Menu Enabled	true (default) or false	Turns the context menu on and off.
impl	additional properties	See the section on jxBrowser-JxWebBrowserImpl.

- [jxBrowser-JxWebBrowserImpl](#)

Parent topic: [Components in workbench module](#)

Figure 1. Property sheet Jx Web Browser Impl

impl (Jx Web Browser Impl)

Visible ☒ true

Enabled ☒ true

Layout

	Value	Units
X		
Y		
Width		
Height		

☒ Fill

The Impl (JxBrowser-JxWebBrowserImpl) component is found in the Workbench palette, under the WebBrowser folder.

In addition to the standard properties (Enabled), these properties are unique to this component.

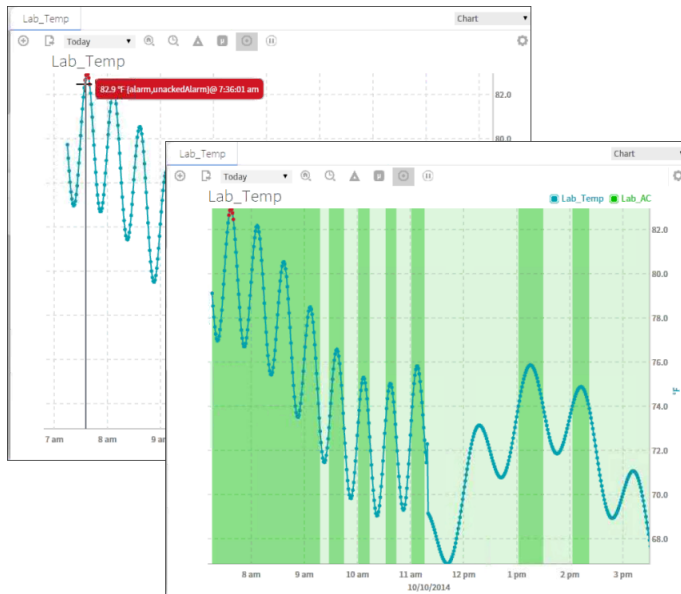
Type	Value	Description
Visible	true or false	Display of the widget Enables if set true and disables when set false.
Layout	Default values: X=0.00, Y=0.00, Width=000.00, Height=000.00 Fill=on, off (default)	Provides X and Y positioning coordinates for the widget as well as, Width, Height, and Fill. X and Y units can be specified as Absolute or Percent. While Width and Height dimensions can be specified as Value, Unit, or Preferred units. The Fill checkbox turns fill on or off.

Parent topic: [workbench-WebBrowser](#)

This is a bjaux, HTML5-based application that incorporates a view with interactive functionality which allows you to edit properties and invoke commands from the view. You can easily add data to a WebWidget, such as the WebChart or Dashboard, simply by dragging one or more components onto the widget. The widget renders in both Workbench and HTML5 Hx interfaces. The widget also integrates into the environment. For example, commands defined for a WebWidget render as added tool bar icons in Workbench, as well as in the HTML5 Hx profile in a web browser.

Examples of the bjaux WebWidget include the following:

- The WebChart displays the **Chart** view which can display historical data and update with live data. Also, in the view you can easily add data and invoke numerous commands and settings to modify data presentation.
- Figure 1. Chart WebWidget



- The CircularGauge displays the graphical gauge view which updates with live data and provides contextual information for the current value. At any time you can dynamically switch the display to another component simply by dragging and dropping a different component onto this widget.

Figure 2. CircularGauge WebWidget



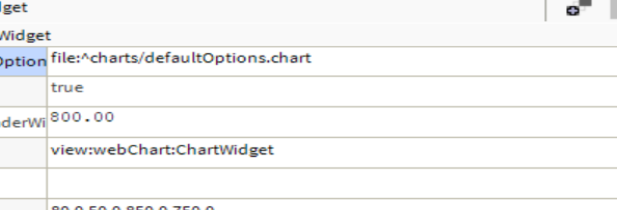
- A Dashboard may be added to any PxPage and displayed in the PxViewer. Additional WebWidgets may be added to the **Dashboard** pane to customize the presentation of data. The dashboard is used to write dashboard-specific data to and from a station for a specific user.

Figure 3. Dashboard WebWidget



Configurable properties for the Chart widget

Figure 4. Chart widget properties



The screenshot shows the 'WebWidget' properties window. The 'defaultOption' property is highlighted in blue. The window is titled 'Properties' and has a close button in the top right corner. The properties are listed in a table-like structure with a left column for the property name and a right column for the value. Some properties have a dropdown arrow on the right side of the value field.

WebWidget	
Web Widget	
defaultOption	file:^charts/defaultOptions.chart
enabled	true
hideHeaderWidth	800.00
js	view:webChart:ChartWidget
layer	
layout	80.0,50.0,850.0,750.0
legendVisibility	Hide
preferredSize	640.0,480.0
title	
titleVisibility	Responsive
visible	true
Wb View Binding	
ord	null
degradeBehavior	None
requiredPermissions	

Property	Value	Description
defaultOption	file:^charts/defaultOptions.chart	Provides an ORD for the defaultOption widget. You can browse to select another widget.
hideHeaderWidth	number	Controls header width. All headers are visible when you set this value to 800 pixels and above.
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).

Property	Value	Description
Enabled	true (default) false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).
Js	view:webChart:ChartWidget (default)	Provides an ORD for the Javascript widget. You can browse to select another widget.
layer	drop-down	null
legendVisibility	drop-down	Selects how to display the legend: Responsive, Show, Hide)
preferredSize	Default values: Width=000.00, Height=000.00	preferredSize can use to set the values of height and width of widgetbar.
title	text	You can add the title.
titlevisibility	drop-down	Provides titleVisibility as per set value. (for example Responsive, Show, Hide)
wbViewBinding	Binding null —>WebWidget (default)	Defines an ORD for the bound label. You can browse to select the ORD. This property also provides selectable options for Degrade Behavior (None, Disable, and Hide).

Parent topic: [Components in workbench module](#)

Parent topic: [Plugins in workbench module](#)

There are many ways to view plugins (views). One way is directly in the tree. In addition, you can right-click on an item and select one of its views. Plugins provide views of components.

In Workbench, access the following summary descriptions on any plugin by selecting **Help > On View (F1)** from the menu, or pressing **F1** while the view is open.

- [Types of plugin modules](#)
Following, is a list of the types of plugin modules:
- [backup-BackupManager](#)
The **Backup Manager** is the default view for a station's **BackupService**. From this view you can issue a backup command, to back up the station's configuration to your local PC, in dist file format. When you issue a **Backup** command, a **File Chooser** window opens to select the destination directory on your PC and the file name for the backup .dist file.
- [chart-ResourceManager](#)
The **Resource Manager** is an available view on any running station.
- [help-BajadocViewer](#)
The BajadocViewer Plugin provides the ability to navigate and browse Baja reference documentation. Baja reference documentation includes both Java API details as well as Baja slot documentation.
- [Plugins in html module](#)
This module has two plugins.
- [Plugins in ldap module](#)
Plugins in ldap module are as follows:
- [Plugins in program module](#)
- [raster-RasterViewer](#)
The **RasterViewer** displays bitmapped image files: GIF, JPEG, PNG in the main window with **Format** and image **Size** at the bottom.
- [Plugins in wbutil module](#)
The wbutil module contains several plugins.
- [Plugins in wiresheet module](#)
Plugins in wiresheet module are as follows:
- [Plugins in workbench module](#)
This module has several plugins which provide many of the standard views.

Following, is a list of the types of plugin modules:

- Plugins in alarm module
- Plugins in backup module
- Plugins in chart module
- Plugins in email module
- Plugins in help module
- Plugins in history module
- Plugins in html module
- Plugins in program module
- Plugins in raster module

- Plugins in schedule module
- Plugins in wiresheet module
- Plugins in wbutil module
- Plugins in Workbench module

Parent topic: [Plugin guides](#)

The **Backup Manager** is the default view for a station's **BackupService**. From this view you can issue a backup command, to back up the station's configuration to your local PC, in dist file format. When you issue a **Backup** command, a **File Chooser** window opens to select the destination directory on your PC and the file name for the backup .dist file.

The default backup destination depends on your station connection, as either:

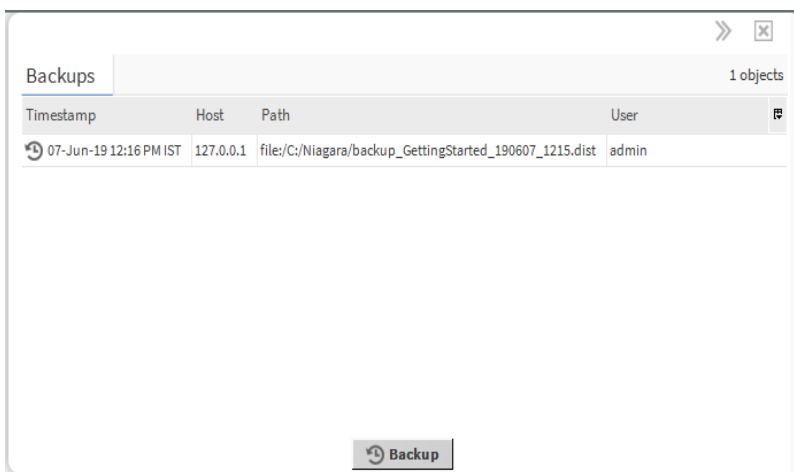
- Workbench (Fox) — !backups: A subdirectory backups under your Niagara release directory. If you have not previously made station backups, this directory is automatically created.
- Browser access (Wb Web Profile) — !backups: A subdirectory backups under the Niagara subfolder of your installed Java 2 runtime environment (Java plugin).

For example: C:\Program Files\Java\j2rel1.4.2_05\niagara\backups. The first time you make a station backups, the system automatically creates this directory.

The default name for a backup file uses a format of:
backup_stationName_YYMMDD_HHMM.dist.

The **Backup Manager** provides a progress bar and Job Log (>> control) for an initiated backup. The **Backup Manager** displays a table of the 10 most recent backups, with the following data columns:

Figure 1. Backup Manager



Column	Description
Timestamp	Indicates the station date and time when the backup was initiated.
Host	Indicates the IP address of the requesting (remote) PC for the backup.
Path	Indicates the file path used on the requesting (remote) PC for saving the backup. Typically, this is relative to the default Niagara directory (!), however, it may be an absolute file path.
User	Indicates the station user that initiated the backup.

Parent topic: [Plugin guides](#)

The **Resource Manager** is an available view on any running station.

It provides a line chart of both CPU and memory usage of the host platform, and updates in real time. In addition, individual resource statistics are provided in a table, which you can refresh by clicking the **Update** button.

Parent topic: [Plugin guides](#)

The BajadocViewer Plugin provides the ability to navigate and browse Baja reference documentation. Baja reference documentation includes both Java API details as well as Baja slot documentation.

The viewer shows documentation for the following:

- Subclasses
- Properties
- Actions
- Topics
- Constructors
- Methods
- Fields

To access Bajadoc, select **Bajadoc On Target** from the main **Help** menu.

Table 1. BajadocViewer menu

Menu	Description
Show Baja Only	Shows the documentation for slots (Properties, Actions, and Topics). When it is set to false, documentation on the Java constructors, methods and fields is also displayed.
Flatten Inheritance	Flattens the inheritance hierarchy into a single set of documentation. When it is false only the Java members and Baja slots declared in the specified class are displayed. When it is set to true all the Java members and Baja slots inherited from super classes are also shown.

Subclasses

Subclasses provides a subclass tree of all that are subclassed from this item.

Properties

Properties represent a storage location of another Niagara object. Flags are boolean values which are stored as a bitmask on each slot in a Baja Object. Some flags apply to all slot types, while some only have meaning for certain slot types.

Table 2. Configuration flags

Flag	Char	Applies	Description
readonly	r	P	The <code>readonly</code> flag is used to indicate slots which are not accessible to users.
transient	t	P	Transient properties do not get persisted when saving a object graph to the file system. Transient properties are usually also readonly, unless they are designed to be a linkable input slot.
hidden	h	P,A,T	Hidden slots are designed to be invisible to the user, and exist only for Java developers. User interfaces should rarely display hidden slots.
summary	s	P	Summary properties are the focal points of any given BComponent. This flag is used by user interface tools to indicate primary properties for display. This might be as a columns in a Table, or as a glyph in a graphical programming tool.
async	a	A	By default Action are invoked synchronously on the callers thread. By using the <code>async</code> flag on an Action, invocations are coalesced and executed asynchronously at some point in the near future on the engine's thread.
noExecute	x	P	No execute properties prevents start/stop from recursing on properties with this flag set.
defaultOnClone	d	P	Specifies that when an object is cloned via the <code>newCopy()</code> method these properties retain their default value, not the clone source's value.
confirmRequired	c	A	When the Action is invoked by a user, a confirmation window must be acknowledged before proceeding.
operator	c	P,A,T	This makes a slot as operator security level. By default when this flag is clear, the slot is an admin security level.
userDefined1	1	P,A,T	User defined.
userDefined2	2	P,A,T	User defined.
userDefined3	3	P,A,T	User defined.
userDefined4	4	P,A,T	User defined.

Actions

An Action is a slot that specifies behavior which may be invoked either through a user command or by an event. Actions provide the capability to provide direction to Components. They may be issued manually by the operator or automatically through links. Actions can be issued by Right-clicking on the Component. The Component bajadoc provides a complete list of Actions available for each Component. Typical Actions include:

- Manual actions are available based on Component type. The following are commonly available:

- Auto (BooleanWritable, NumericWritable and EnumWritable)
- Active (BooleanWritable)
- Inactive (BooleanWritable)
- Override (NumericWritable and EnumWritable)

Each of the above actions is issued to the priorityArray of the Component at level 8 (Manual Operator).

- Many other Actions are available on other Components. Each Action available for a Component is listed in the Actions sect2 of the Component bajadoc.

Topics

Topics represent the subject of an event. Topics contain neither a storage location, nor a behavior. Rather a topic serves as a place holder for a event source.

Parent topic: [Plugin guides](#)

This module has two plugins.

- [html-WbHtmlView](#)
This component allows you to view the contents of HTML files.
- [html-SpyViewer](#)
SpyViewer allows you to view diagnostic information about the system.

Parent topic: [Plugin guides](#)

This component allows you to view the contents of HTML files.

WbHtmlView Menus

The Workbench main menu functions are available.

WbHtmlView Toolbar

In the **WbHtmlView**, the toolbar contains navigation and editing buttons. In addition, **Find**, **FindNext** and **FindPrev** toolbar buttons are available.

HTML Support: HTML Tags

The **WbHtmlView** attempts to ignore mismatched or missing Tags. It can parse any HTML, no matter how badly messed up the Tags are, although it might do a pretty bad job of rendering the results. It does simplistically attempt to repair missing <p>, but other problems are solved by ignoring Tags. Warnings will appear on the command line showing its best guess as to what the problem was. Valid tags include:

- a - anchor Ex:Title<a> and Title<a> (Attributes href and name)

- **b** - bold text style Ex: `<b>bold text<b>`
- **body** - document body
- **br** - forced line break
- **code** - computer code fragment
- **col** - not supported
- **colgroup** - not supported
- **dd** - definition description
- **div** - not supported
- **dl** - definition list
- **dt** - definition term
- **em** - emphasis Ex: `<em>emphasis text<em>`
- **font** - local change to font (Attributes Color, name and size)
- **h1** - heading Ex: `<h1 class='title'>Title<h1>`
- **h2** - heading
- **h3** - heading
- **h4** - heading
- **h5** - heading
- **h6** - not supported
- **head** - document head
- **hr** - horizontal rule
- **html** - document root element
- **i** - italic text style
- **img** - embedded image Ex: `<img src='../doc/icons/x16/home.png'>` An `<img>` without an `align` attribute is treated as an 'inline' image. An `align` of `left` or `right` causes text to flow around the image.
- **li** - list item
- **link** - a media-independent link Ex: `<link rel='StyleSheet' href='module://bajau/doc/style.css' type='text/css'/>`
- **meta** - not supported

- object - not supported
- ol - ordered list
- p - paragraph Ex:<p class='note'>Note<p>
- pre - preformatted text
- span - not supported
- style - not supported
- table - table (Attributes align, border, bordercolor, cellpadding, cellspacing and width)
- tbody - not supported
- td - table data cell supports ALIGN (left, right, and center), BGCOLOR, COLSPAN, ROWSPAN and WIDTH
- tfoot - not supported
- th - table header cell supports ALIGN (left, right, and center), BGCOLOR, COLSPAN, ROWSPAN and WIDTH
- thead - not supported
- title - document title
- tr - table row supports ALIGN (left, right, and center) and BGCOLOR
- tt - teletype or monospaced text style
- ul - unordered list
- webLauncher - not supported

HTML Attributes

HTML Elements have associated properties, called attributes, which may have values. Attribute/value pairs appear before the final ">" of an element's start tag. Valid attributes include:

- align - vertical or horizontal alignment Deprecated; elements: img, object? values: bottom, left, middle, right or top.
- align - table position relative to window Deprecated; elements: table; values: center, left or right
- align - align, text alignment Deprecated; elements: div?, h1?, h2?, h3?, h4?, h5?, h6?, p; values: center, justify, left or right
- align - alignment; elements: col?, colgroup?, tbody?, td, tfoot?, th, thead?, tr; values: center, char, justify, left or right
- bgcolor - background Color Deprecated; elements: h1, h2, h3, h4, h5, h6?, p, td, th, tr

- border - controls frame width around table; elements: table
- cellpadding - spacing within cells; elements: table
- cellspacing - spacing between cells; elements: table
- class - class name or set of class names for stylesheets; elements: most elements
- color - text Color Deprecated; elements: font, h1, h2, h3, h4, h5, h6?, p, pre, td?, th?, tr?
- colspan - number of cols spanned by cell; elements: td, th
- content - associated information; elements: meta?
- href - URI for linked resource; elements: a, link
- id - name to an element; elements: most elements
- lang - Language Code; elements: most elements not supported
- name - named link end; elements: a
- name - meta information name; elements: meta?
- rel - forward link types; elements: link; values: stylesheet
- rowspan - number of rows spanned by cell; elements: td, th
- size - size of font; elements: font; value: fixed 1-7 or relative -7 to +7
- summary - purpose/structure for speech output; elements: table
- type - advisory content type; elements: link; values: text/css
- width - table width; elements: table

Character Entity References

Character Entity References are supported including the following:

- & - ampersand &
- ' - apostrophe '
- © - copyright ©
- > - greater than >
- “ - double quotation, left “
- ‘ - single quotation, left ‘
- < - less than <

- — - em dash —
- - non breaking space " "
- – - en dash –
- ” - double quotation, right ”
- ’ - single quotation, right ’
- " - quotation mark "
- ® - registered trademark ®
- ™ - trademark ™

Stylesheet support

Stylesheets are supported using a link in the header as follows:

```
<head>
<title>Sample</title>
<link rel='StyleSheet' href='module://bajau/doc/style.css' type
='text/css' ^
</head>
```

See the default stylesheet used for *Niagara Developer Guide*: `module://bajau/doc/style.css` or the CSS stylesheet used for this document at `docbook.css`.

Pseudo-classes and Pseudo-elements

Anchor pseudo-classes include:

- A:link unvisited links unsupported
- A:visited visited links unsupported
- A:active active links unsupported

CSS1 Properties

Stylesheet elements supported include:

- background - The 'background' property is a shorthand property for setting the individual background properties (i.e., 'background-color', 'background-image', 'background-repeat', 'background-attachment' and 'background-position') at the same place in the style sheet.
- background-attachment unsupported
- background-color - This property sets the background Color of an element.
- background-image unsupported
- background-position unsupported

- background-repeat unsupported
- border unsupported
- border-bottom unsupported
- border-bottom-width unsupported
- border-color unsupported
- border-left unsupported
- border-left-width unsupported
- border-right unsupported
- border-right-width unsupported
- border-style unsupported
- border-top unsupported
- border-top-width unsupported
- border-width unsupported
- clear unsupported
- color - This property describes the text Color of an element (often referred to as the foreground Color).
- display unsupported
- float unsupported
- font unsupported
- font-family unsupported
- font-size unsupported
- font-style unsupported
- font-variant unsupported
- font-weight unsupported
- height unsupported
- letter-spacing unsupported
- line-height unsupported
- list-style-image unsupported

- list-style-position unsupported
- list-style-type unsupported
- margin unsupported
- margin-bottom unsupported
- margin-left unsupported
- margin-right unsupported
- margin-top unsupported
- padding unsupported
- padding-bottom unsupported
- padding-left unsupported
- padding-right unsupported
- padding-top unsupported
- text-align
- text-decoration unsupported
- text-indent unsupported
- text-transform unsupported
- white-space unsupported
- width unsupported
- word-spacing unsupported

Parent topic: [Plugins in html module](#)

SpyViewer allows you to view diagnostic information about the system.

It contains the following:

- sysinfo - sysinfo provides system information.
- stdout - stdout provides access to standard output.
- systemProperties - systemProperties provides access to system properties.
- logSetup - logSetup allows you to config your log severities dynamically. There is also an option to flush the current settings to log.properties.
- sysManagers - sysManagers provides information on managers. These include:

- registryManager
 - schemaManager
 - componentNavEventManager
 - moduleManager
 - engineManager
 - leaseManager
 - serviceManager
 - licenseManager
 - stationManager
- navSpace - provides information on the navSpace.
 - userinterface - if System - userInterface provides information on the user interface framework.
 - fox - fox provides information on fox client and server sessions.

Parent topic: [Plugins in html module](#)

Plugins in ldap module are as follows:

- ldap-BasicKrb5ConfEditor
- ldap-AdvancedKrb5ConfEditor
- [Basic Krb5 Conf Editor](#)
In Niagara, an added editor view is available under the **Tools** menu, **Basic Krb5 Conf Editor**. It configures certain properties of an existing Kerberos configuration file (krb5.conf).
- [ldap-KerberosConfigurationTool](#)
In Niagara, two editors provide simple text editors, which you can use to manually edit an existing Kerberos configuration file (krb5.conf) or to create a new one.

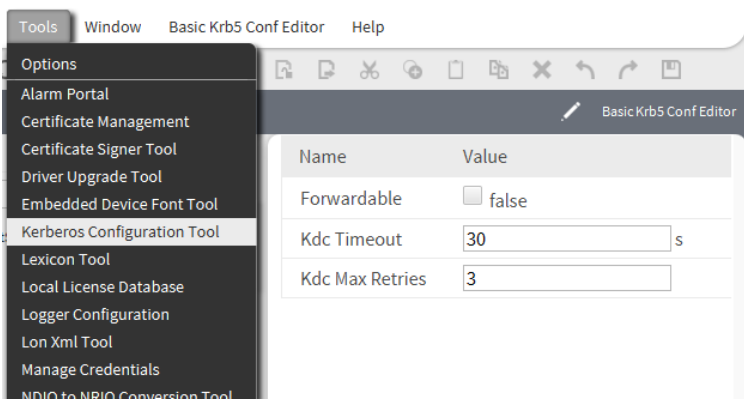
Parent topic: [Plugin guides](#)

In Niagara, an added editor view is available under the **Tools** menu, **Basic Krb5 Conf Editor**. It configures certain properties of an existing Kerberos configuration file (krb5.conf).

Properties

Kerberos authentication requires the ability to acquire Kerberos tickets that can be forwarded. The editor allows you to enable and disable the Forwardable property.

Figure 1. Basic Krb5 Conf Editor view



To access this view, click **Tools > Kerberos Configuration Tool**.

Property	Value	Description
Forwardable	true (default), false	Enables and disables forwarding of Kerberos tickets.
Kdc Timeouts	30 (default)	Required for redundant server support, specifies the length of time the station attempts to connect to the key distribution center before failing the connection attempt.
Kdc Max Retries	3 (default)	Required for redundant server support, specifies the maximum number of times the station attempts to connect to one key distribution center before to the next one.

Note: Values entered for the Kdc Timeouts and Kdc Max Retries properties should be tailored to your specific scenario based on how long successful kdc connections generally take and when to configure the the cut-off time after which the connection is considered to have failed. As with the connection timeout above, this time needs to be not too short to cause false connection failures, but not so long as to cause excessive delays when a server is down.

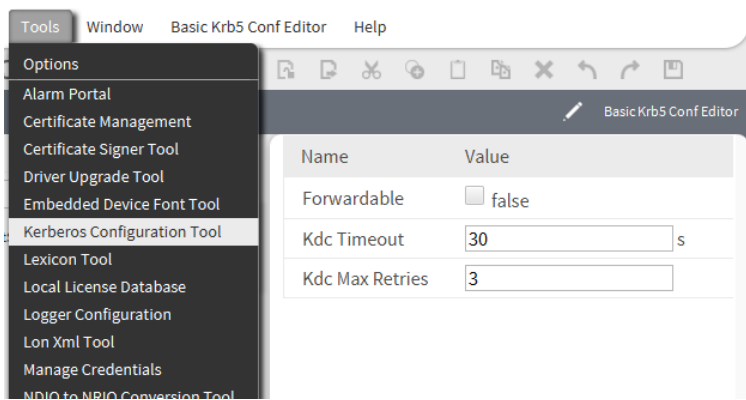
Parent topic: [Plugins in Idap module](#)

In Niagara, two editors provide simple text editors, which you can use to manually edit an existing Kerberos configuration file (krb5.conf) or to create a new one.

Basic Krb5 Conf Editor

Kerberos authentication requires the ability to acquire Kerberos tickets that can be forwarded. The editor allows you to enable and disable the Forwardable property.

Figure 1. Basic Krb5 Conf Editor



Property	Value	Description
Forwardable	true (default), false	Enables and disables forwarding of Kerberos tickets.
Kdc Timeouts	30 (default)	Required for redundant server support, specifies the length of time the station attempts to connect to the key distribution center before failing the connection attempt.
Kdc Max Retries	3 (default)	Required for redundant server support, specifies the maximum number of times the station attempts to connect to one key distribution center before to the next one.

Note: Values entered for the Kdc Timeouts and Kdc Max Retries properties should be tailored to your specific scenario based on how long successful kdc connections generally take and when to configure the the cut-off time after which the connection is considered to have failed. As with the connection timeout above, this time needs to be not too short to cause false connection failures, but not so long as to cause excessive delays when a server is down.

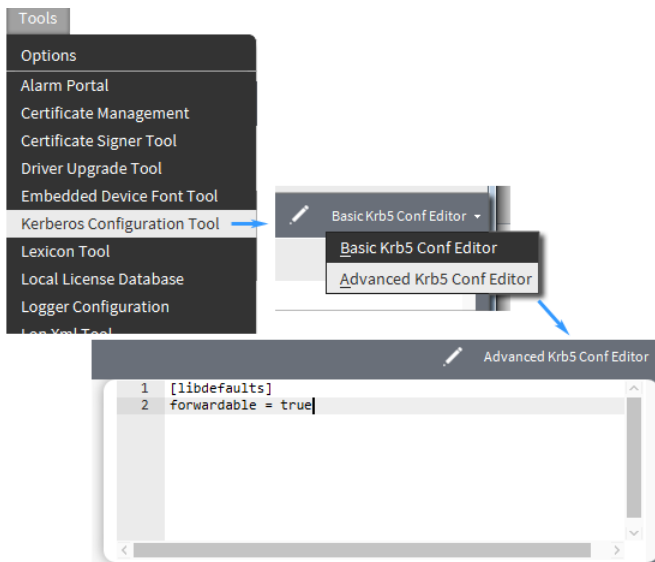
Advanced Krb5 Conf Editor

On a Windows host, the primary location for the file is: NIAGARA_HOME/security/krb5.conf. Only if this file is missing would you fall back to the Java krb.conf or operating system specific krb.conf/ini.

On a Linux host the file location is: /etc/krb5.conf.

Note: If you are working with Linux, some systems may require a more advanced krb5.conf file. If that is the case, have your Kerberos administrator set up this file for you.

Figure 2. Advanced Krb5 Conf Editor



The file requires only the two lines contained in this view.

Parent topic: [Plugins in ldap module](#)

- BatchEditor
- ProgramEditor
- ProgramModuleBuilder
- RobotEditor
- [program-BatchEditor](#)
The **Batch Editor** is the default view on the ProgramService. It allows you to perform a variety of operations on slots of multiple components by issuing a single “batch” command.
- [Program Editor](#)
The **Program Editor** view provides the ability to view and edit Program components.
- [program-ProgramModuleBuilder](#)
The **Program Module Builder** is the default view on the ProgramModule.
- [program-RobotEditor](#)
The **Robot Editor** is used to write Robots that can be run via the ProgramService.

Parent topic: [Plugin guides](#)

The **Batch Editor** is the default view on the ProgramService. It allows you to perform a variety of operations on slots of multiple components by issuing a single “batch” command.

You can add (specify) components by dragging from the Nav tree, or copy and pasting into the view, or by using the Find objects (Bql Query Builder) function—or any combination of the three methods.

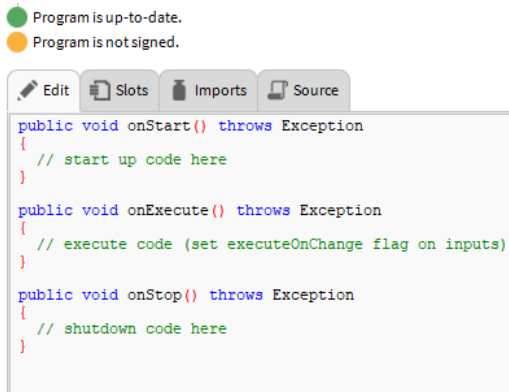
As needed, use the **Clear Selected Items** control to remove any items before running the operation.

The **Batch Editor** can be a real time saver when the same configuration change needs to be replicated among multiple component slots. For complete details, refer to the *Niagara Engineering Notes*.

Parent topic: [Plugins in program module](#)

The **Program Editor** view provides the ability to view and edit Program components.

Figure 1. Program Editor



To view, right-click a Program and select **Program Editor**. It shows **Edit**, **Slots**, **Imports**, and **Source** tabs.

Tab	Description
Edit	Edits the onExecute , onStart , onStop and freeForm methods. An example from the demo Database follows: <pre> BStatusNumeric inA = getInA(); BStatusNumeric inB = getInB(); BStatusNumeric out = getOut(); out.setValue(inA.getValue() + inB.getValue()); </pre>
Slots	Shows and changes the slots of the program component. It includes Slot , # , Name , Display Name , Definition , Flags , Type and, Facets for each slot.
Imports	Shows the modules that have been imported. It also allows the following: • Import Type—imports a new type. • Import Package—imports a new package. • Remove—Removes added Import Type and Import Package.
Source	Edits the source of the program component. The editor supports special Color coding for Java Files. An example from the demo Database follows: <pre> /* Auto-generated ProgramImpl Code */ import java.util.*; /* java Predefined*/ import javax.baja.sys.*; /* baja Predefined*/ import javax.baja.status.*; /* baja Predefined*/ import javax.baja.util.*; /* baja Predefined*/ </pre>

Tab	Description
	<pre> import com.tridium.program.*; /* program Predefined*/ public class ProgramImpl extends com.tridium.program.ProgramBase { ////////////////////////////////////// // Getters ////////////////////////////////////// public BStatusNumeric getOut() { return (BStatusNumeric)get("out"); } public BStatusNumeric getInA() { return (BStatusNumeric)get("inA"); } public BStatusNumeric getInB() { return (BStatusNumeric)get("inB"); } ////////////////////////////////////// // Setters ////////////////////////////////////// public void setOut(javax.baja.status.BStatusNumeric v) { set("out", v); } public void setInA(javax.baja.status.BStatusNumeric v) { set("inA", v); } public void setInB(javax.baja.status.BStatusNumeric v) { set("inB", v); } ////////////////////////////////////// // onExecute ////////////////////////////////////// public void onExecute() throws Throwable { BStatusNumeric inA = getInA(); BStatusNumeric inB = getInB(); BStatusNumeric out = getOut(); out.setValue(inA.getValue() + inB.getValue()); } } </pre>

Program Editor Menus

The Workbench main menu functions are available. When the **Program Editor** is visible, the following **Program Editor** menu function is also available:

- Import Type (🔍)
- Import Package (🔍)

- Remove ()
- Add Slot (Ctrl + A)
- Delete (Ctrl + Delete)
- Rename Slot (Ctrl + R)
- Compile and (F9) () - Compiles and saves the source of the program component
- Compile (Ctrl + F9) () - Compiles the source of the program component.

Program Editor Toolbar

The Workbench toolbar contains navigation and editing buttons. When the **Program Editor** is visible, additional toolbar buttons include:

- Find ()
- Replace ()
- Compile and Save ()
- Compile ()
- Console Prev
- Console Next

Parent topic: [Plugins in program module](#)

The **Program Module Builder** is the default view on the **ProgramModule**.

It lets you create a module from one or more **Program** components, such that they may be versioned and provisioned just like other modules.

Parent topic: [Plugins in program module](#)

The **Robot Editor** is used to write Robots that can be run via the **ProgramService**.

RobotEditor Menus

The Workbench main menu functions are available. When the **RobotEditor** is visible, the following RobotEditor menu function is also available:

- RobotEditor Compile (Ctrl + F9) - Compiles the source of the Robot component.
- RobotEditor Compile & Run (F9) - Compiles and run the source of the Robot component.

RobotEditor Toolbar

The Workbench toolbar contains navigation and editing buttons. When the **Robot Editor** is visible, additional toolbar buttons include:

- Find ()
- Replace ()
- Compile ()
- Compile & Run ()
- Console Prev ()
- Console Next ()

Parent topic: [Plugins in program module](#)

The **RasterViewer** displays bitmapped image files: GIF, JPEG, PNG in the main window with **Format** and image **Size** at the bottom.

Parent topic: [Plugin guides](#)

The wbutil module contains several plugins.

The following topics describe plugins that are included in the wbutil module.

- [Category Browser \(wbutil-CategoryBrowser\)](#)
This view is the default view of the station's CategoryService, and typically where you spend most of your time assigning categories to components after initially creating the categories.
- [HTML5- Category Browser](#)
In Niagara 4.14 and later, there is added browser support for Category Browser. The HTML 5 version of this view is a web-browser-based implementation and it provides the same functions as the Workbench view.
- [Category Manager \(wbutil-CategoryManager\)](#)
This view of the CategoryService allows you to create, enable and delete the groups that the security model uses to control access to the objects in a station. Once you create categories, you use the **Category Browser** view to centrally assign system objects to categories. Or, at the individual component level, you use a component's **Category Sheet** view to assign the component to one or more categories.
- [wbutil-CategorySheet](#)
This view assigns a component to one or more categories (or configures it to inherit categories from its parent. Every component has a **Category Sheet** view.
- [wbutil-PermissionsBrowser](#)
This view allows you to quickly review the objects that someone, who has been assigned a given role may access. You access this view by right-clicking RoleService in the Nav tree and clicking **Views > Permissions Browser**.
- [wbutil-ResourceEstimator](#)
The **Resource Estimator** tool allows you to estimate station resources based a number of variables, which you enter in various fields.

- [wbutil-TodoList](#)
The **Todo List** tool allows you to enter, summarize, group, and prioritize pending Workbench tasks.
- [wbutil-UserManager](#)
The **User Manager** is the primary view of the **UserService**.

Parent topic: [Plugin guides](#)

This view is the default view of the station's **CategoryService**, and typically where you spend most of your time assigning categories to components after initially creating the categories.

Note: When an admin user (user with Admin write privilege on the **CategoryService** and on a particular station component being adjusted) makes a category mask adjustment, the user must also have at least the Operator write privilege on the category being adjusted in the category mask for the station object. This includes changes to check marks in the "Inherit" column—the user must have at least Operator write access to any altered categories applied from the Inherit change.

Figure 1. Category Browser view

Category Browser									
	Inherit	User	Admin	Operator	Viewer	Category 5	Category 6	Category 7	Category 8
Alarm	n/a								
Config	n/a								
Services	✓								
Drivers	✓								
Apps									
category	✓								
Temp1	✓								
Temp2	✓								
Alarm									
Ramp	✓								
SineWave	✓								
Files	n/a								

To access this view, expand **Config > Services** and double-click **CategoryService**.

Columns

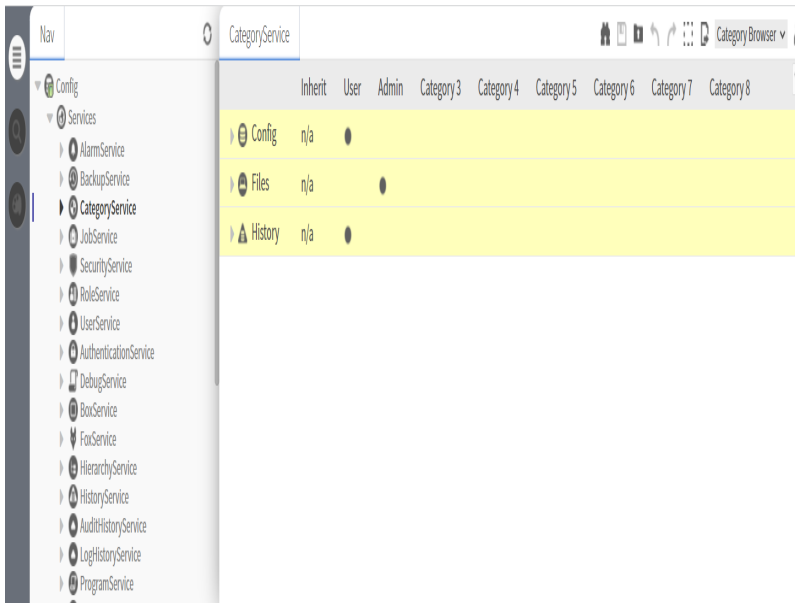
Column	Description
Inherit	A check mark indicates that the object inherits the category from its parent in the table.
User	Indicates the category. All system objects except for those listed as assigned to Admin are assigned to this category.
Admin	These objects default to the Admin category: • The configuration services: UserService, CategoryService, and ProgramService • All files (the entire file space)

Column	Description
Categories 3–8	A bold bullet indicates that the object is assigned to the category. A grayed out bullet indicates inheritance. Blank indicates that the category has not been assigned.

Parent topic: [Plugins in wbutil module](#)

In Niagara 4.14 and later, there is added browser support for Category Browser. The HTML 5 version of this view is a web-browser-based implementation and it provides the same functions as the Workbench view.

Figure 1. CategoryUxBrowser



To access this view, expand **Config > Services** and double-click **CategoryService** or right-click **CategoryService > Views > Category Browser**.

Columns

Column Name	Description
Inherit	Displays that the object inherits the category from its parent in the table.
User	Displays the category.
Admin	Displays the services and files in admin category.
Category 3 — 8	Displays the object assigned to particular category.

Toolbar for Category Browser

Toolbar Icon and Name	Description
 Expand All	Expands all the services.
 Collapse All	Collapse all the services.
 Save	Save your changes.
 Undo	Reverses the previous command.
 Redo	Restores a command action after the Undo command has removed it.
 Multi-selection Mode	Enables you to individually select multiple points without holding down the ctrl key.
 Export	Exports the current view or object.

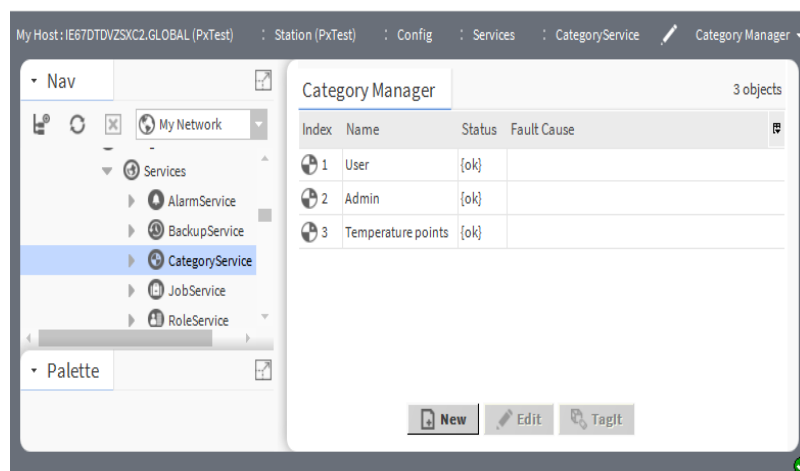
Parent topic: [Plugins in wbutil module](#)

This view of the CategoryService allows you to create, enable and delete the groups that the security model uses to control access to the objects in a station. Once you create categories, you use the **Category Browser** view to centrally assign system objects to categories. Or, at the individual component level, you use a component's **Category Sheet** view to assign the component to one or more categories.

Columns

You can assign an object to many categories at the same time. Each object stores its own categories.

Figure 1. Category Manager, Temperature Points as Category



To access this view, expand **Config > Services** and right-click **CategoryService** and click **Views > Category Manager**.

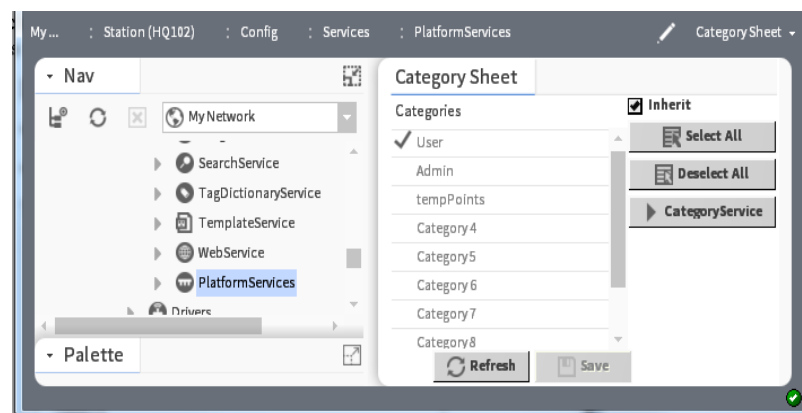
Column	Description
Index	Reports a unique number for the category as it is known to the station.
Name	Displays descriptive text that reflects the purpose of the entity or logical grouping.
Status	Reports the current condition of the entity as of the last refresh: {alarm}, {disabled}, {down}, {fault}, {ok}, {stale}, {unackedAlarm}
Fault Cause	Indicates the reason for a fault.

Parent topic: [Plugins in wbutil module](#)

This view assigns a component to one or more categories (or configures it to inherit categories from its parent. Every component has a **Category Sheet** view.

Note: When an admin user (user with Admin write privilege on the **CategoryService** and on a particular station component being adjusted) makes a category mask adjustment, the user must also have at least the Operator write privilege on the category being adjusted in the category mask for the station object. This includes changes to check marks in the “Inherit” column—the user must have at least Operator write access to any altered categories applied from the Inherit change.

Figure 1. Category Sheet



Column/option	Description
Categories	Provides one table row for each category name.
Inherit	A check mark indicates that the component belongs to the same categories as its parent component. No check mark allows you to make explicit category assignments for this component.

Buttons

- **Select All** is effective if Inherit is cleared. Clicking this button assigns this component to all categories in this station.

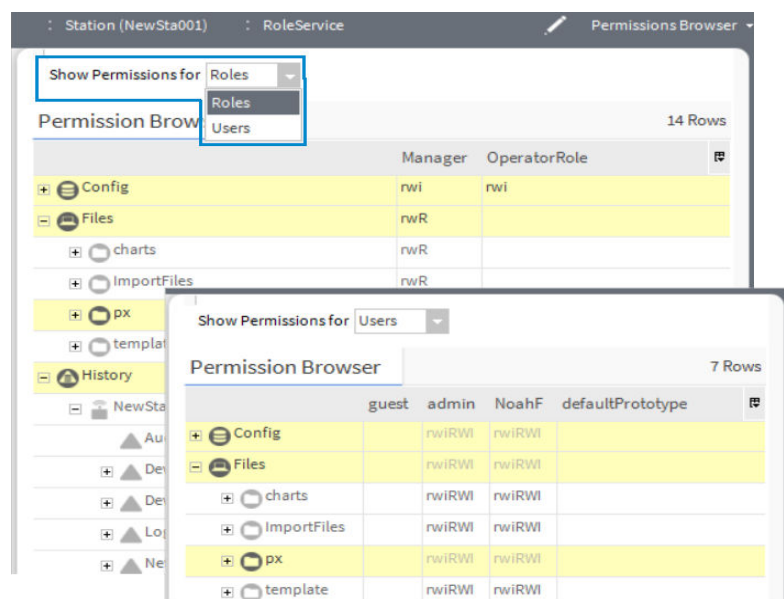
- **Deselect All** is effective if Inherit is cleared. Clicking this button removes this component from all categories.
- **Category Service** opens the **Category Browser**.
- **Refresh** re-displays the **Category Sheet**.
- **Save** records the changes made.

Parent topic: [Plugins in wbutil module](#)

This view allows you to quickly review the objects that someone, who has been assigned a given role may access. You access this view by right-clicking RoleService in the Nav tree and clicking **Views > Permissions Browser**.

Note: In Niagara, there is added support for the **UserService** in the **Permissions Browser** view. Use the **Show Permissions for** drop-down list to switch between permissions for Users, and for Roles. When viewing permissions for **Users**, the view displays a separate column for each user as well as any prototype.

Figure 1. Permissions Browser view showing permissions for roles and users



Columns represent roles or users, and rows identify the objects in the station, with each table cell showing user permissions.

- Yellow rows are objects explicitly assigned with permissions.
- Dimmed rows represent objects that inherit their permissions from their parent object.

Double-click a cell to bring up the permissions window for that role or user depending on which option is selected in the **Show Permissions for** drop-down list. This allows you to globally change permission levels for any category in the station.

Column	Description
First column	Each Nav tree node occupies a row in the table. This expandable tree lets you navigate to objects of interest to your current permissions.
Admin	Reports the rights assigned to the <code>admin</code> role. As this is a super user, <code>admin</code> has rights to read, write and invoke action for all objects. permissionsR = readW = writeI = invoke action admin level permissions appear in upper case.
User	Reports the rights assigned to the <code>user</code> role. The default is no rights assigned. permissionsr = readw = writei = invoke action operator permissions appear in lower case.

Parent topic: [Plugins in wbutil module](#)

The **Resource Estimator** tool allows you to estimate station resources based a number of variables, which you enter in various fields.

It is one of several tools in Workbench's **Tools** menu.

Parent topic: [Plugins in wbutil module](#)

The Todo List tool allows you to enter, summarize, group, and prioritize pending Workbench tasks.

It is one of several tools in Workbench's **Tools** menu.

Parent topic: [Plugins in wbutil module](#)

The **User Manager** is the primary view of the **UserService**.

You use it to add, edit, and delete users for accessing the station.

Parent topic: [Plugins in wbutil module](#)

Plugins in wiresheet module are as follows:

- wiresheet-WebWireSheet
- wiresheet-WireSheet
- [wiresheet-WebWireSheet](#)
In Niagara 4.11 and later, there is added support for **Web Wire Sheet**. This is an implementation of the **Wire Sheet** view that runs natively in a web browser. It provides the same basic functionality as the existing **Wire Sheet**, but is accessible from your desktop or mobile device as part of the HTML5-based web interface.
- [wiresheet-WireSheet](#)
The **Wire Sheet** view shows the contents of this component. It can be used on a component of a running station or a component in a Bog File. If in a running station, it is active and real-time updates are provided. In order to command or select a different view of the item, you may right-click to get the menu.

Parent topic: [Plugin guides](#)

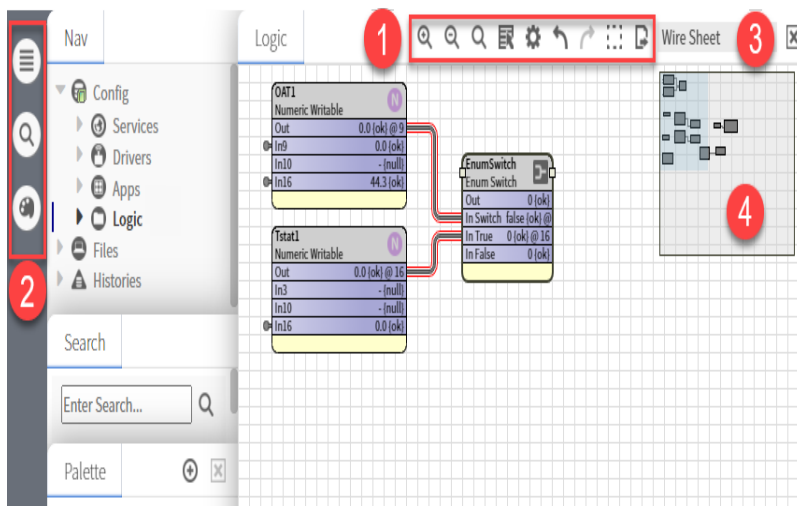
In Niagara 4.11 and later, there is added support for Web **Wire Sheet**. This is an implementation of the **Wire Sheet** view that runs natively in a web browser. It provides the same basic functionality as the existing **Wire Sheet**, but is accessible from your desktop or mobile device as part of the HTML5-based web interface.

Prerequisites

Only users with admin WRITE privileges can access the Web **Wire Sheet**.

When you are logged in to the Niagara web interface (station connection via browser), the Web **Wire Sheet** is available for any component on which you have the necessary permission. To view, right-click a component and click **Views > Wire Sheet**.

Figure 1. Web Wire Sheet runs in a browser connection



1. The Web **Wire Sheet** toolbar offers similar functions as Workbench.
2. The **Nav** tree, **Search** and **Palette** sidebar buttons expand and close the sidebar panes.
3. The view selector allows you to quickly display different views.
4. This feature provides a small thumbnail view of the entire wire sheet for orientation and navigation purposes.

Web **Wire Sheet** matches the existing **Wire Sheet** behavior in Workbench, but there are a few differences:

- Copy/Paste functionality is restricted to the current station (that is, you cannot copy components out of one station and paste into another, or to/from your computer's hard drive).
- **Wire Sheet** functionality specific to the ACE drivers is not yet available.

What's new

As of Niagara 4.12, the following features are available in Web **Wire Sheet**:

Table 1. Linking, relating or adding objects

	Workbench (4.12)	Niagara 4.11 (browser connection)	Niagara 4.12 (browser connection)
Relate objects by dragging a wire from the bottom bar of a selected object to the bottom bar of a target object.	yes	no	yes
Drag components from the palette and Nav tree onto Web Wire Sheet .	yes	yes	yes
Link or relate by dragging components from the palette, search side bar or Nav tree onto another object. Link/relate toggle remembers the setting of the previous action.	no	no	yes

Table 2. Displaying Web Wire Sheet objects

	Workbench (4.12)	Niagara 4.11 (browser connection)	Niagara 4.12 (browser connection)
Objects, links, relations and knobs	yes	yes	yes
Lucid icon displays for each selection	yes	no	yes
Wire Sheet Thumbnail to navigate to the desired logic	yes	yes	yes

Table 3. Using the object context menu

	Workbench (4.12)	Niagara 4.11 (browser connection)	Niagara 4.12 (browser connection)
Cut, copy, paste, paste special, delete within component space	yes	yes	yes
Duplicate objects	yes	no	yes
Paste special to specify the number of copies and whether to keep all links and relations.	yes	yes	yes
Export current view or object.	yes	yes	yes
Edit tags	yes	yes	yes
Edit link	yes	yes	yes
Delete individual knob links from Delete command menu.	yes	no	yes
Select and navigate to individual knob links from the GoTo Link command menu.	yes	no	yes
Individually select and follow linked components (property sheet) from the Goto Linked Component command menu.	yes	no	yes

Composite editor	yes	yes	yes
Display or hide pin slots.	yes	yes	yes

Touch devices

Web **Wire Sheet** works largely the same on a touch device, with the following differences:

- Use a long press to bring up the context (right-click) menu.
- Dragging from a palette may not be fully supported on all devices.
- To start a rubber band (selection), use a long-press and then move your finger.
- Use one finger to scroll, or two fingers to zoom.
- For hyperlinking, double-click is unavailable, but you can always long press and choose a view.
- As of Niagara 4.12, you can select all or individual object glyphs without holding down the ctrl key.

Web Wire Sheet Toolbar

Figure 2. Web Wire Sheet toolbar

The toolbar contains the following tools in order from left to right:

Zoom in	
Zoom out	
Search	
Select all objects (as of Niagara 4.12)	
Wire Sheet Options	
Undo	

Redo	
Enable selection mode: toggle (as of Niagara 4.12)	
Export the current view or object	

For additional wire sheet usage details, see “Wire Sheet object management” in the “Data and control model” section of this guide.

Parent topic: [Plugins in wiresheet module](#)

The **Wire Sheet** view shows the contents of this component. It can be used on a component of a running station or a component in a Bog File. If in a running station, it is active and real-time updates are provided. In order to command or select a different view of the item, you may right-click to get the menu.

Wire Sheet Menus

The WireSheet includes the following menus:

- WireSheet Main Menu
- WireSheet Component Menu
- WireSheet Background Menu
- WireSheet Link Menu

Wire Sheet Main Menu

The **Wire Sheet** main menu functions are available. When the *wiresheet; is visible, the following **Wire Sheet** menu functions are also available:

- Delete
- Arrange
- Select All
- Show Thumbnail
- Show Grid
- Show Status Colors
- Show Relations

- Show Links

Wire Sheet Component Menu

If you right-click any Component in the **Wire Sheet**, you can choose from the following:

- Views - Go to any of the views of the Component.
- Actions - Perform any of the Actions on the Component.
- Edit Tags
- Make Template
- Cut (Ctrl + X)
- Copy (Ctrl + C)
- Paste (Ctrl + V)
- Paste Special
- Duplicate(Ctrl + D)
- Delete (Delete)
- Composite Editor
- Link Mark
- Link Form
- Link To
- Relation Mark
- Relate From
- Relate To
- Rename (Ctrl + R)
- Set Display Name
- Reorder
- Composite
- Export
- **More > Pin Slots**

WireSheet Background Menu

If you right-click the Background of the wire sheet, you can choose from the following:

- New - You can select new component from this menu.
- Cut (Ctrl + X)
- Copy (Ctrl + C)
- Paste (Ctrl + V)
- Paste Special
- Duplicate (Ctrl + D)
- Delete (Delete)
- Delete (Ctrl + Delete)
- Edit
- Tags
- Rename (Ctrl + R)
- Arrange
- Select All
- Recorder
- Composite

WireSheet Link Menu

If you right-click any link in the wire sheet, you can choose from the following:

- New - You can select new component from this menu.
- Cut (Ctrl + X)
- Copy (Ctrl + C)
- Paste (Ctrl + V)
- Paste Special
- Duplicate (Ctrl + D)
- Delete (Delete)
- Delete (Ctrl + Delete)

- Edit
- Tags
- Rename (Ctrl +R)
- Arrange
- Select All
- Recorder
- Composite

WireSheet Toolbar

The Workbench toolbar contains navigation and editing buttons. When the **Wire Sheet** is visible, additional toolbar buttons include:

Menu	Description
Delete	Eliminates the selected link(s). You can Delete links only in the Wire Sheet .
Arrange	Arranges the items in the Wire Sheet . You can Arrange All or Arrange Selection.
Select All	Selects all items in the Wire Sheet .
Show Thumbnail	Shows the thumbnail view in the upper right corner of the wire sheet. This allows you to toggle whether this function is enabled. This function may be accessed from the menu under Wire Sheet . It allows you to display a thumbnail of the Wire Sheet view in the corner of the wire sheet. You can use the thumbnail to move around the wire sheet by dragging the shaded area in the thumbnail. You can also hold down the Ctrl key and move the thumbnail around the wire sheet to keep it out of your way.
Show Grid	When enabled, the grid is displayed in the background of the wire sheet. This helps you to align Components when you move them. You may control whether the grid layer is enabled or visible from the Wire Sheet menu or the Tools > Options menu. This allows you to enable and disable this function.
Show Status Colors	Shows the Status Colors. This allows you to toggle whether this function is enabled. This function may be accessed from the Menu under Wire Sheet . It allows you to display StatusColors in the Wire Sheet .
Show Relations	Shows the relations of the components.
Show Links	Shows the links.

Wire Sheet Options

Menu item	Description
Show Grid	When enabled, displays the grid in the background of the Wire Sheet . This helps you to align components when you move them. To control whether the grid layer is enabled or visible, use the Workbench Options menu. This allows you to enable and disable this function.

Menu item	Description
Show Status Colors	Shows the Status Colors in the Wire Sheet . You may toggle whether this function is enabled. Use the menu under Wire Sheet to display Status Colors in the Wire Sheet .
Show Thumbnail	Shows a thumbnail view in the upper right corner of the Wire Sheet . You may toggle whether this function is enabled. Use the menu under Wire Sheet to access this function. It allows you to display the thumbnail view. To use the thumbnail to move around the Wire Sheet , drag the shaded area in the thumbnail. To keep the thumbnail out of your way, hold the Ctrl key and move the thumbnail around the Wire Sheet .
Show Links	Shows the links.
Show Relations	Enables and disables showing the relationships among components.
Thumbnail X	Configures the x axis for the default position of the thumbnail view.
Thumbnail Y	Configures the y axis for the default position of the thumbnail view.
Link Highlighting	Enables and disables link highlighting.
Max Width	Specifies the maximum size for the Wire Sheet width.
Max Height	Specifies the maximum size for the Wire Sheet height.

For additional wire sheet usage details, see “Wire Sheet object management” in the “Data and control model” section of this guide.

Parent topic: [Plugins in wiresheet module](#)

This module has several plugins which provide many of the standard views.

- [workbench-CollectionTable](#)
The **CollectionTable** allows you to view tables.
- [workbench-DirectoryList](#)
The **Directory List** view provides a listing of the subdirectories and files found in a given Directory.
- [workbench-HexFileEditor](#)
The **HexFileEditor** allows you to view hexadecimal files.
- [workbench-JobServiceManager](#)
The **Job Service Manager** is the default view on a station's **JobService**.
- [workbench-ModuleSpaceView](#)
The **Module Space View** allows you to view Modules.
- [workbench-NavContainerView](#)
Nav Container View is a default listing of nNav children.
- [workbench-NavFileEditor](#)
The **Nav File Editor** allows you to view and edit Nav Files.
- [workbench-PropertySheet](#)
The property sheet views display all of the user visible properties of the selected component. You can change any properties that you have permissions to change. The property sheet views apply to a component of a running station or a component in a bog file. To see properties of components in a **PropertySheet**, expand each component.
- [webEditors-Mulitsheet](#)
- [Relation Sheet](#)
- [workbench-ServiceManager](#)
Service Manager allows you to view services.

- [workbench-SlotSheet](#)
This component shows all user-visible component slots. Each includes Property, Action, and, Topic slots.
- [workbench-StationSummary](#)
StationSummary is the default view on a station.
- [workbench-Synthetic Module File View](#)
The **Synthetic Module File View** is the default view on a  synthetic module.
- [workbench-TextFileEditor](#)
The TextFileEditor Plugin provides a powerful Color coded text editor. It supports Color coding of C, java and xml file types.
- [workbench-WbPxView](#)
PxView is a dynamic view which may be added to Components as a property.
- [workbench-WbServiceManagerView](#)
 **Workbench Service Manager** allows you to view services. It is available from the **Tools** menu.
- [workbench-WebBrowserView](#)
The **Web Browser View** is an instance of the BWebBrowser class. It provides a browser view within the Workbench interface.
- [workbench-WebWidget](#)
This is a bajaux, HTML5-based application that incorporates a view with interactive functionality which allows you to edit properties and invoke commands from the view. You can easily add data to a WebWidget, such as the WebChart or Dashboard, simply by dragging one or more components onto the widget. The widget renders in both Workbench and HTML5 Hx interfaces. The widget also integrates into the environment. For example, commands defined for a WebWidget render as added tool bar icons in Workbench, as well as in the HTML5 Hx profile in a web browser.

Parent topic: [Plugin guides](#)

The **CollectionTable** allows you to view tables.

One way to create a table is through a BQL collection like:

```
local:|fox:|station:|slot:/ControlObjects|bql:select
displayName,type, out, facets from control:ControlPoint
```

The Table options menu, allows you to **Reset Column Widths**, **Print** and **Export**.

Parent topic: [Plugins in workbench module](#)

The **Directory List** view provides a listing of the subdirectories and files found in a given Directory.

Double-clicking an item opens its default view. Files are displayed with an icon based on file type.

Directory List Toolbar

The Workbench toolbar contains navigation and editing buttons as described in “About the toolbar”.

Refresh

Refresh allows you to synchronize the cached components with the actual file system.

Parent topic: [Plugins in workbench module](#)

The **HexFileEditor** allows you to view hexadecimal files.

It provides a binary view of a file's contents.

Parent topic: [Plugins in workbench module](#)

The **Job Service Manager** is the default view on a station's **JobService**.

It provides a table listing of up to the last 10 Jobs executed by the station since the last station start. Order is oldest job at top, most recent job at bottom.

To see details on any job, click the button next to its status descriptor. A **Job Log** window displays all the interim steps for the job, including timestamps and relevant messages.

To dispose of any job, click the close (X) button to remove it from the station.

Note: Only the last ten jobs are saved. The system clears all jobs when it restarts.

Parent topic: [Plugins in workbench module](#)

The **Module Space View** allows you to view Modules.

OptionsButton

The **Options** button enables and disables columns (turns them on and off).

Parent topic: [Plugins in workbench module](#)

Nav Container View is a default listing of nNav children.

Parent topic: [Plugins in workbench module](#)

The **Nav File Editor** allows you to view and edit Nav Files.

It provides a view of the pages in the station and a view of the Nav tree. You can drag pages to the Nav tree to add them to the Nav File. The name and Ord are shown at the bottom of the window.

To use the Nav File, place its filename in the Default Nav File property of the **WebService**.

Parent topic: [Plugins in workbench module](#)

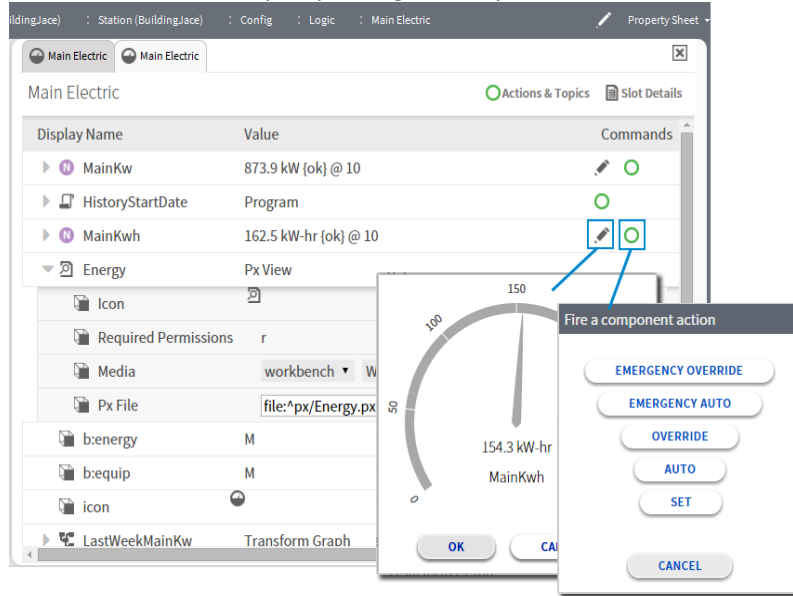
The property sheet views display all of the user visible properties of the selected component. You can change any properties that you have permissions to change. The property sheet views apply to a component of a running station or a component in a bog file. To see properties of components in a **PropertySheet**, expand each component.

There are two types of **Property Sheet** views:

- **Property Sheet** view

An HTML5 property sheet view, shown here, which provides functionality such as interactive field editors and action commands, graphical web gauge display for points, and

slot sheet details. Property changes that you make in this view are saved automatically.



- **AX Property Sheet view**

The default property sheet view for the Niagara 4 releases.

In the **AX Property Sheet** view, if you want to enter a URL, copy the value and paste it into the property sheet. When you change any property, its symbol will become red until you click **Save**.

AX Property Sheet Menus

The Workbench main menu functions are available. If you right-click any component in the **AX Property Sheet**, you can choose from the following:

Item	Description
Views	Opens the views for the selected component
Actions	Provide one or more actions to perform on the component.
New	Creates a standard item, such as a folder, text block, NumericWritable, etc.
Edit Tags	Opens the Edit Tags window, permitting you to add or remove tags on a component.
Make Template	Creates a template from the selected root component, collects all associated Px and graphic files, and invokes the Template view, allowing you to configure the template for deployment.
Cut	Copies to clipboard and after pasting the copy, deletes the cut item.
Copy	Copies the selected item to clipboard where it remains.
Paste	Pastes the copied item from the clipboard into the current cursor location.
Duplicate	Makes a copy in the same location as the selected item.

Item	Description
Delete	Removes the selected item.
Find	Opens the Component Finder window so you can search for a component.
Link Mark	Sets up a selected component in your popup menu, making it temporarily available for linking-from or linking-to other points.
Link From	Links to a selected component from another component that has been marked, using Link Mark from the popup menu.
Link To	Links from a selected component to another component that has been marked, using Link Mark from the popup menu.
Relation Mark	Sets up a selected component in your popup menu, making it temporarily available for relating-from or relating-to other points.
Relate From	Adds a relation from a selected component to another component that has been marked using Relation Mark from the popup menu.
Relate To	Adds a relation to a selected component from another component that has been marked using Relation Mark from the popup menu.
Rename	Opens the Rename window for changing the slot name of the selected component as it appears in the ORD. You can only rename one component at a time.
Set Display Name	Creates a display name for the selected component as it appears in a Nav tree, Wire Sheet , and Property Sheet views.
Reorder	Opens the Reorder Points window, which provides the following commands for reordering points within the selected parent component: Move Up moves the item higher in the list. Move Down moves the item lower in the list. Sort by Name sorts the list by name. Sort by Type sorts the list by type of component. Reset returns the list to its default setting.
Composite	Opens the Composite Editor window, which manages the visible slots.
Export	Exports a selected component in the oBix .xml format.
Config Flags	Assigns or removed a permissions flag on a component. This is available on properties in the AX Property Sheet view. Right-click a property to open the Config Flags window.

Property Sheet Toolbar

The Workbench toolbar contains navigation and editing buttons.

Icon	Description
 Back	Hyperlinks back to last displayed URL.

Icon	Description
 Forward	Hyperlinks forward to the URL selected by the Back command.
 Up Level	Jumps to the parent of current page.
 Side Bars	Displays the Side Bars list.
 Recent Ords	Displays a recent ORDs list.
 Home	Jumps to the home URL (Workbench splash screen).
 Refresh	Refreshes the current page.
 Refresh Tabs	Refreshes all open tabs.
 Session Info	Provides information on the current connection.
 Open	Displays the Open list.
 Save(Ctrl + S)	Saves the changes to the current component.
 Save Bog	Saves changes to the bog file.
 Bog File Protection	Enters, changes, or adds a bog file passphrase.
 Export	Exports the current view or object.
 Cut (Ctrl + X)	Cuts the selected item and places it on the clipboard.
 Copy (Ctrl + C)	Copies the selected item and places it on the clipboard.
 Paste (Ctrl + V)	Pastes the items on the clipboard into the current location.
 Duplicate (Ctrl + D)	Copies and pastes to duplicate the item.
 Delete	Deletes the current selection.
 Undo (Ctrl + Z)	Reverses the previous command/action.
 Redo (Ctrl + Alt + Z)	Restores a command/action after the Undo command removed it.

BQL is one Scheme used to Query in the Niagara Framework. An Ord is made up of one or more Queries. A Query includes a Scheme and a body. The bql Scheme has a body with one of the following formats:

- BQL expression
- Select projection FROM extent Where predicate

You can create the **Ord Qualifier**, Select, FROM and Where portions of a Query.

Ord Qualifier

In the left window, you can select an Ord to use as the qualifier. It will immediately be placed in the BQL statement at the top when you select it.

If you select the history Scheme, your options will vary from those shown here.

workbench:ProjectionBuilder

To build the projection for a BQL request instead of typing it, use the ProjectionBuilder in Bql Builder. You can select an item from the center window to use with the select statement. Press the $\Rightarrow$ right arrow to add each one to the projection.

workbench:ExtentBuilder

An extent can be one or more of the following:

- "*" all available from the target
- all property slots
- all methods that return non-void and take zero parameters

In order to build the Extent for a BQL request instead of typing it, you can use the ExtentBuilder in Bql Builder. You can select the **Restrict Type** and choose the module and item to use with the FROM clause. It will immediately appear in the BQL statement at the top.

This also changes the items that are available in the projection.

If you choose a history Scheme, the ExtentBuilder will provide different selections. The extent for a History typically can be one of the following:

- "from /demo/Float" where demo is the station name
- "from !Float" where "!" signifies the current station

workbench:QualifierBuilder

To build the Qualifier for a BQL request, type the request in the **QualifierBuilder** of the Bql Builder. The text you type immediately appears in the BQL statement at the top.

Edit Facets

Edit Facets allows for the viewing and editing of facets. To change facets, use the button to the right of the facets. It opens the **Edit Facets** window.

From here you can **Add**, **Remove** or select the **Enum Range** window.

Add

Add adds a Key and Type.

Remove

Remove deletes facets.

Enum Range window

To view or set the range of values for a Enum, use the »button. It opens the **Enum Range** window.

You can enter one of the standard Enumerations in the field under the **Use Enum Type in Range (module:name)**.

Click the **Use Enum Type in Range (module:name)** to use the Enumeration that you entered. To enter or change an Enumeration, enter the Ordinal in the field above the **Add** button. Next, enter the new value for the Tag and click **Modify**. Click **OK** to complete the procedure.

Parent topic: [Plugins in workbench module](#)

Service Manager allows you to view services.

It is available on the ServiceContainer.

Parent topic: [Plugins in workbench module](#)

This component shows all user-visible component slots. Each includes Property, Action, and, Topic slots.

Figure 1. Workbench Slot Sheet

Slot Sheet							
Slot	#	Name	Display Name	Definition	Flags	Type	Facets
Property	0	visible	Visible	Frozen		baja:Boolean	
Property	1	enabled	Enabled	Frozen		baja:Boolean	
Property	2	layout	Layout	Frozen		bajauti:Layout	
Property	3	styleClasses	Style Classes	Frozen	h	baja:String	
Property	4	styleId	Style Id	Frozen	h	baja:String	
Topic	5	keyEvent	Key Event	Frozen		bajauti:KeyEvent	
Topic	6	mouseEvent	Mouse Event	Frozen		bajauti:MouseEvent	
Topic	7	focusEvent	Focus Event	Frozen		bajauti:FocusEvent	
Property	8	title	Title	Frozen		baja:String	
Property	9	ord	Ord	Frozen		baja:Ord	
Property	10	location	Location	Frozen	rto	baja:String	
Property	11	progress	Progress	Frozen	rto	baja:Double	
Property	12	progressRunning	Progress Running	Frozen	rto	baja:Boolean	
Property	13	showProgressIndicator	Show Progress Indicator	Frozen		baja:Boolean	
Property	14	contextMenuEnabled	Context Menu Enabled	Frozen		baja:Boolean	
Action	15	reload	Reload	Frozen		void (void)	
Topic	16	initialized	Initialized	Frozen		baja:Value	
Topic	17	closing	Closing	Frozen		baja:Value	
Topic	18	loaded	Loaded	Frozen		baja:Boolean	
Topic	19	statefulMenu	Stateful Menu	Frozen		baja:String	

The **Slot Sheet** is a table that shows the following for each slot:

Column	Description
Slot	Components are collections of three types of slots: Property represents a storage location for the object. Action specifies a behavior that may be invoked either through a user command or by an event. Topic represents the subject of an event. Topics contain neither a storage location nor a behavior. Rather, a topic serves as a place holder for an event source.
#	Identifies a serial numbers given to the each slot as it is added.
Name	Provides unique name within the slot type. Slot names must contain ASCII letters or numbers.
Display Name	Reports the name for the slot that the user sees.
Definition	Identifies if the slot is frozen or dynamic. A frozen slot is defined at compile time within a Type's Java class. That means that frozen slots are consistent across all instances of a specified Type—they don't change. Dynamic slots may be added, removed, renamed, and reordered during runtime—they can change.
flags	Reports the indicators (flags) that modify an object's presentation or behavior. For example, read-only, operator allowed, and hidden, are three of the slot flags that may be used to restrict the presentation or behavior of an object.
Type	Indicates the data type of the configuration property.
Facets	Contain metadata about an object. For example, the object's unit of measurement is a type of facet. Facets may be viewed in the slot sheet and edited from a component Property Sheet .

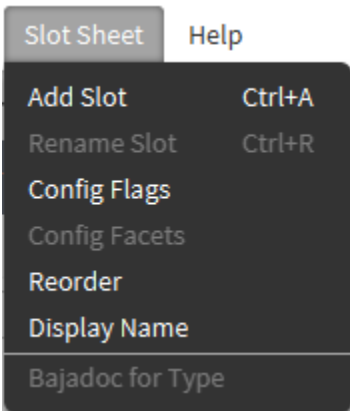
The **Slot Sheet** also optionally shows any Name Maps. The **Slot Sheet** includes the following menus:

- SlotSheet Main Menu
- SlotSheet Component Menu

SlotSheet Main Menu

The Workbench main menu functions are available. When the **Slot Sheet** is visible, the following menu functions are also available:

Figure 2. Slot Sheet Main Menu



Menu Name	Description
 Add Slot (Ctrl + A)	Adds a new Slot to the component.
 Rename Slot (Ctrl + R)	Renames the added slot in the component.
 Config Flags	Sets config flags for the slot in the component.
 Config facets	Configures facets for the slot in the Component.
 Reorder	Reorders the slot in the Component.
 Display Name	Changes the display name of the slot in the Component.

SlotSheet Component Menu

If you right-click any Component in the **Slot Sheet** or the background, you can choose from the following:

Figure 3. SlotSheet Component Menu

<u>A</u> dd Slot	Ctrl+A
<u>C</u> opy	Ctrl+C
<u>D</u> elete	Delete
<u>R</u> ename Slot	Ctrl+R
Con <u>f</u> ig Flags	
Con <u>f</u> ig Facets	
Re <u>o</u> rder	
Display <u>N</u> ame	
<u>B</u> ajadoc for Type	

Component Name	Description
 Add Slot (Ctrl + A)	Adds a slot to the component.
 Copy (Ctrl + C)	Copies an existing slot to the component.
 Delete (Delete)	Deletes the slot from the component.
 Rename Slot (Ctrl + R)	Renames the slot in the component.
 Config Flags	Sets config flags for the slot in the component.
 Config facets	Configures facets for the slot in the component.
 Reorder	Reorders the slot in the component.
 Display Name	Changes the display name of the slot in the component.

SlotSheet Toolbar

The Workbench toolbar contains navigation and editing buttons. When the **Slot Sheet** is visible, additional toolbar buttons include:

Toolbar Menu	Description
 Add Slot (Ctrl + A)	Adds a slot to the component.
 Rename Slot (Ctrl + R)	Renames the slot in the component.

Toolbar Menu	Description
Add Name Map	Adds a Name Map to the component. You right-click on the property displayName to delete or rename Name Map, and displayName_xx to delete or rename Name Map (xx) where xx is the Language Code.
 Config facets	Configures facets for the slot.

Parent topic: [Plugins in workbench module](#)

StationSummary is the default view on a station.

It holds primary components (for example, **Config, Files, History**) and shows specific configuration information about the station's host platform, including:

- Station Name
- Host
- Host Model
- Host Id
- Niagara Version
- Java Version
- OS Version
- Locale
- Current Time

Parent topic: [Plugins in workbench module](#)

The **Synthetic Module File View** is the default view on a  synthetic module.

For details, see the *Niagara Engineering Notes*.

Parent topic: [Plugins in workbench module](#)

The TextFileEditor Plugin provides a powerful Color coded text editor. It supports Color coding of C, java and xml file types.

TextFileEditor Menus

The Workbench main menu functions are available.

TextFileEditor Toolbar

The Workbench toolbar contains navigation and editing buttons as described in “About the toolbar”.

File Types

The TextFileEditor Plugin provides a powerful Color coded text editor. It supports Color coding of C, java, properties, Python and xml file types. See Text Editor options to change editor options including Color coding.

C Files

The TextFileEditor supports special Color coding for C files including:

- Preprocessor - `#include`
- Line Comment - `/ comment /`
- Multiline Comment - `/* comment */`
- String literal - `"string"` and `'string'`
- Number literal - `'0'` and `'F'`
- Keyword - blue - `if`

CSS Files

The TextFileEditor supports special Color coding for CSS files including:

- Identifier - HTML `element` or CSS `identifier`
- Line Comment - `/ comment /`
- Multiline Comment - `/* comment */`
- String literal - `"string"` and `'string'`
- Number literal - `'0'` and `'F'`
- Keyword - blue - `if`

HTML Files

The TextFileEditor supports special Color coding for HTML files including:

- Identifier - HTML `element`
- Multiline Comment - `<!-- Comments here -->`
- String literal - `"string"` and `'string'`
- Number literal - `'0'` and `'F'`
- Keyword - blue - `if`

Java Files

The TextFileEditor supports special Color coding for Java files including:

- Bracket - `( { [`
- Keyword - `if`
- Line Comment - `/ comment /`
- Multiline Comment - `/* comment */`
- String literal - `"string"` and `'string'`
- Number literal - `'0'` and `'F'`

JavaScript Files

The TextFileEditor supports special Color coding for JavaScript files including:

- Bracket - `( { [`
- Keyword - `if`
- Line Comment - `/ comment /`
- Multiline Comment - `/* comment */`
- String literal - `"string"` and `'string'`
- Number literal - `'0'` and `'F'`

Properties Files

The TextFileEditor supports special Color coding for properties files including:

- Line Comment - `#`
- Bracket - `=`

Python Files

The TextFileEditor supports special Color coding for Python files including:

- Bracket - `{ } ( ) [ ]`
- Keyword - `if`
- Line Comment - `#`
- String literal - `"string"` and `'string'`
- Number literal - `'0'` and `'F'`

Xml Files

The TextFileEditor supports special Color coding for Xml files including:

- Multiline Comment - `<!-- comment -->`
- Bracket - `< > < >`
- String literal - `"string"` and `'string'`

Parent topic: [Plugins in workbench module](#)

PxView is a dynamic view which may be added to Components as a property.

PxViews store the view contents in a PxFile which is an XML file with a Px extension. The view itself is defined as a tree of `bajoui:Widgets`.

For more information about Px views, see the *Niagara Graphics Guide*.

WbPxView Menus

The Workbench main menu functions are available. When the **Px Viewer** is visible.

PxViewer View Source Xml

You can view the source XML of a Px Page by selecting **View Source Xml** from the main menu.

Parent topic: [Plugins in workbench module](#)

 **Workbench Service Manager** allows you to view services. It is available from the **Tools** menu.

Parent topic: [Plugins in workbench module](#)

The **Web Browser View** is an instance of the `BWebBrowser` class. It provides a browser view within the Workbench interface.

Note: The **Web Browser View** often acts as a “wrapper” for other views that provide specific functionality. In cases such as this, when you click **Help > On View**, help details will pertain only to **Web Browser View** not to the contents of the view.

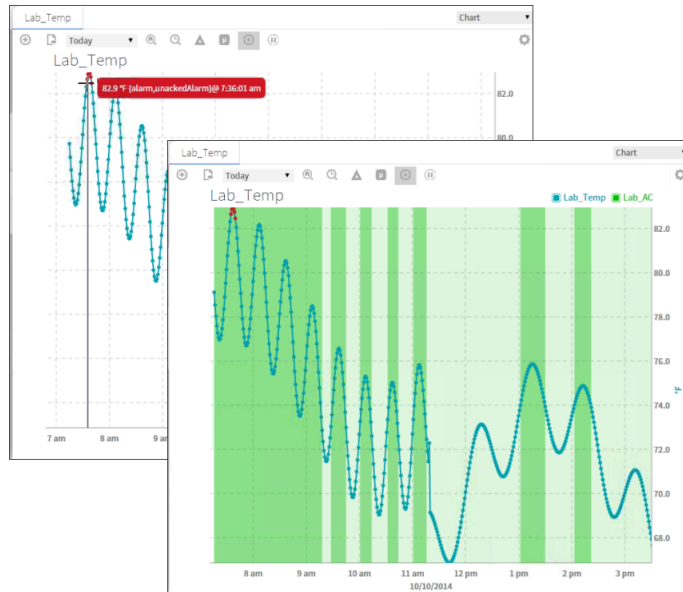
Parent topic: [Plugins in workbench module](#)

This is a `bajaux`, HTML5-based application that incorporates a view with interactive functionality which allows you to edit properties and invoke commands from the view. You can easily add data to a `WebWidget`, such as the `WebChart` or `Dashboard`, simply by dragging one or more components onto the widget. The widget renders in both Workbench and HTML5 Hx interfaces. The widget also integrates into the environment. For example, commands defined for a `WebWidget` render as added tool bar icons in Workbench, as well as in the HTML5 Hx profile in a web browser.

Examples of the `bajaux WebWidget` include the following:

- The WebChart displays the **Chart** view which can display historical data and update with live data. Also, in the view you can easily add data and invoke numerous commands and settings to modify data presentation.

Figure 1. Chart WebWidget



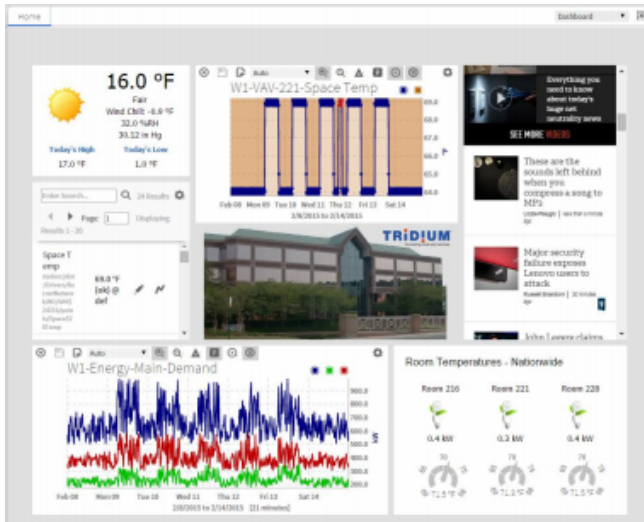
- The CircularGauge displays the graphical gauge view which updates with live data and provides contextual information for the current value. At any time you can dynamically switch the display to another component simply by dragging and dropping a different component onto this widget.

Figure 2. CircularGauge WebWidget



- A Dashboard may be added to any PxPage and displayed in the PxViewer. Additional WebWidgets may be added to the **Dashboard** pane to customize the presentation of data. The dashboard is used to write dashboard-specific data to and from a station for a specific user.

Figure 3. Dashboard WebWidget



Configurable properties for the Chart widget

Figure 4. Chart widget properties

The screenshot shows the 'Properties' window in Qt Designer for a 'WebWidget'. The 'defaultOptions' section is expanded, displaying various configuration options. The 'Wb View Binding' section is also visible at the bottom.

WebWidget		
Web Widget		
defaultOptions	file: ^charts/defaultOptions.chart	...
enabled	true	▼
hideHeaderWidth	800.00	
js	view:webChart:ChartWidget	...
layer		▼
layout	80.0,50.0,850.0,750.0	...
legendVisibility	Hide	▼
preferredSize	640.0,480.0	...
title		
titleVisibility	Responsive	▼
visible	true	▼
Wb View Binding		
ord	null	...
degradeBehavior	None	▼
requiredPermissions	r	...

Property	Value	Description
defaultOption	file:^charts/defaultOptions.chart	Provides an ORD for the defaultOption widget. You can browse to select another widget.
hideHeaderWidth	number	Controls header width. All headers are visible when you set this value to 800 pixels and above.
Visible	true (default) false	Sets the table to be visible in the Px page interface (true) or not (false).

Property	Value	Description
Enabled	true (default) false	Activates (true) and deactivates (false) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Layout	additional properties	Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width).
Js	view:webChart:ChartWidget (default)	Provides an ORD for the Javascript widget. You can browse to select another widget.
layer	drop-down	null
legendVisibility	drop-down	Selects how to display the legend: Responsive, Show, Hide)
preferredSize	Default values: Width=000.00, Height=000.00	preferredSize can use to set the values of height and width of widgetbar.
title	text	You can add the title.
titlevisibility	drop-down	Provides titleVisibility as per set value. (for example Responsive, Show, Hide)
wbViewBinding	Binding null —>WebWidget (default)	Defines an ORD for the bound label. You can browse to select the ORD. This property also provides selectable options for Degrade Behavior (None, Disable, and Hide).

Parent topic: [Components in workbench module](#)

Parent topic: [Plugins in workbench module](#)

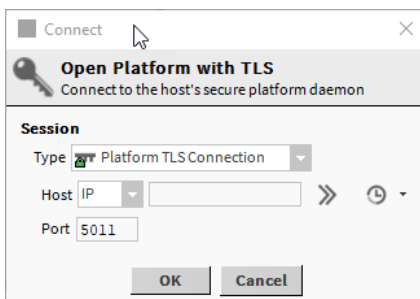
Windows create and edit database records or collect information when accessing a component. You access them by dragging a component from a palette into a station or by clicking a button.

Windows do not support **On View (F1)** and **Guide on Target** help. To learn about the information each contains, search the help system for key words.

- [Platform Connect](#)
This window opens when you open a platform (supervisor PC or controller).
- [Station Connect window](#)
This window opens when you open a station.
- [Bql Query Builder](#)
You can use the **Bql Query Builder** to create and execute a discovery query for finding various types of system entities, including components, points, video elements, and other objects. For example, you can use a BQL query to locate points and video elements that you can bind to Px widgets for visualization.

This window opens when you open a platform (supervisor PC or controller).

Figure 1. Platform connect window

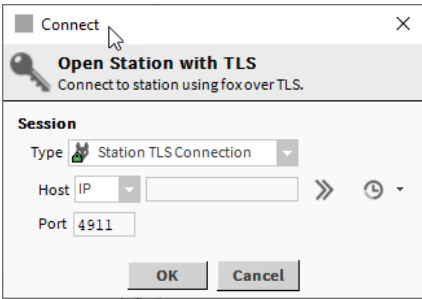


Name	Value	Description
Type	drop-down list (defaults to Platform TLS Connection.)	Selects the type of connection, standard or TLS secure.
Host (type)	type drop-down box and chooser (defaults to IP)	Identifies the platform.
IP address	IP address	Identifies the IP address or URL of the host platform.
Port	number (defaults to 5011)	Identifies the port for secure platform communication.

Parent topic: [Windows](#)

This window opens when you open a station.

Figure 1. Station Connect window



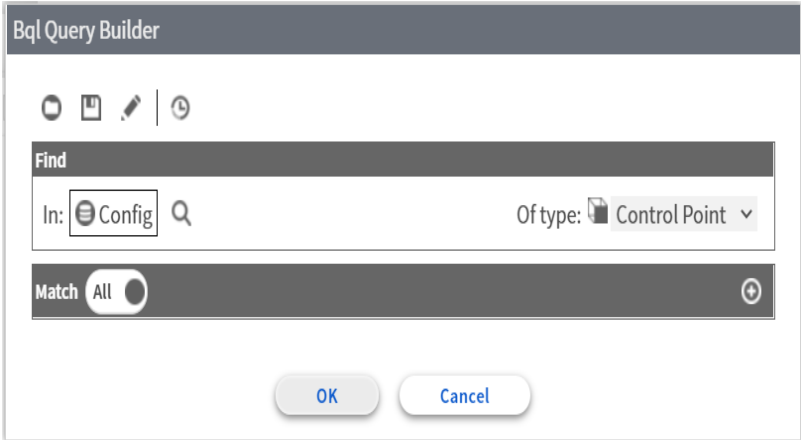
Name	Value	Description
Type	drop-down list	Defaults to Station TLS Connection.
Host (type)	drop-down list	Defaults to IP.
IP address	text	This is where you enter the IP address or URL of the host platform.
Port	number	The port for secure station (foxs) communication. Defaults to 4911.

Parent topic: [Windows](#)

You can use the **Bql Query Builder** to create and execute a discovery query for finding various types of system entities, including components, points, video elements, and other objects. For example, you can use a BQL query to locate points and video elements that you can bind to Px widgets for visualization.

The **Bql Query Builder** provides a means for you to find objects and filter them in a single discovery process. Starting with Niagara 4.12, the **Bql Query Builder** window is also available when you are connected to your station through a web browser.

Figure 1. Bql Query Builder properties



Property	Value	Description
Find In: (root identifier)	read-only	Use this field to browse and find a root location or database to begin the query. Click the search icon to select your starting location.

Property	Value	Description
 search icon	icon	Opens the Choose Root window. This window selects where to begin the search. Your choice updates the root identifier to the left and, depending on the root and Of type selection, populates the area to the right of the icon with additional search criteria.
Of type	drop-down list	Identifies the entity type. (Custom Type) opens two drop-down lists: one for choosing the module and the other for selecting a module component.
Match	drop-down list	Determines the type of match to execute. All finds all records that precisely match the BQL query. Any finds all records that generally match the query.

Parent topic: [Windows](#)

A notification that a defined event has occurred or an indication that some value is not within an appropriate or expected range. For example, a security breach, temperature limit, or equipment malfunction can initiate an alarm notification. Text and icons on the Alarm Console identify alarm severity.

A table view that lists all current alarms for an individual station. This view is available on the alarm recipient component (in the Alarm Service).

Note: Use AlarmConsole (concatenated) if referring to the class.

A table view that lists alarms collected from multiple stations. To access this view from the main menu, click **Tools > AlarmPortal**.

Note: Use AlarmPortal if referring to the class.

A logical grouping of system objects (components, files and histories) by directly assigning objects to the category. For example, basic categories may be used to group objects by geography (floor 1, floor 2, etc.) or type of object (lighting, HVAC, etc.). Each category may be subject to separate user roles and permissions. Each new station has two default categories: user (category 1) and Admin (category 2).

You manage categories using the CategoryService.

A PKI (Public Key Certificate) or digital certificate is an electronic document used to prove ownership of a public key. The certificate includes information about the key, the identity of its owner, and the digital signature of an entity that verified the validity of the certificate's contents. If the signature is valid, and the client can trust the signer, the client can be confident that it can use the public key contained in the certificate to communicate with the server.

A piece of self-describing framework software that can be assembled like building blocks to create new applications. Components represent individual points—such as outside temperature, office temperature, occupancy, and schedules—that generate analog data for analysis within the system. Components differ from modules in that components comprise an implementation of the framework, whereas modules comprise the framework software itself.

A configuration flag is a boolean value that is stored as part of a bitmask on each slot of a Baja object. Some flags apply to all slot types, while others only have meaning for certain slot types. You access a slot's config flags by right-clicking the slot and clicking **ConfigFlags**.

Represents a folder used to hold components, logic, Px graphics, etc. For example: WebWidget

In the narrowest terms, control points refer to the eight point types found in the baja control palette under the **Points** folder. In broader terms, control points include components from the control and kitControl palettes. Most of these components are based on the eight basic point types. They inherit from BooleanPoint, EnumPoint, NumericPoint, and StringPoint.

ControlPoint is the base class for all point types in the Baja control architecture. A ControlPoint maps to one value that a driver reads or writes. All ControlPoints have a StatusValue property called Out.

If the predefined proxyExt is not a NullProxyExt, the system considers the point a proxy point. This means that it is a local representation of a point that actually exists in an external device. The framework uses the driver to maintain synchronization.

Software that provides generic functionality. The framework can be customized by adding user-written code.

Note: Do not initial cap the word framework.

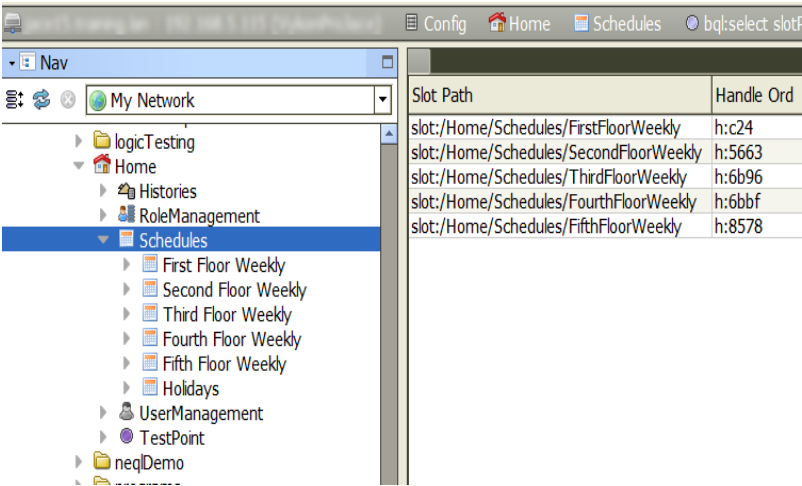
system

A handle Ord is a unique reference to a component that persists throughout the lifecycle of the component in a station.

A BOrd property is often used to reference a component in the station. It may use the slot path Ord, for example, something like `station:|slot:/Home/Schedules/FirstFloorWeekly`. When you add a component to the station, the framework assigns a unique handle Ord, a shorthand reference, which might be something like `h:c24`. You can rename the component, move the component to another location, or rename a parent and the handle Ord remains the same even though the slot path Ord changes.

Consider these weekly schedules with components nested under folders in a location called **Home > Schedules**. A bql query (`bql:select slotPath, handleOrd from schedule:WeeklySchedule`) displays the current slot path Ord and handle Ord for each weekly schedule component.

Figure 1. Slot Path and Handle Ord



If you rename the parent folder from Schedules to WeeklySchedules, the slot path Ords change for each of the weekly schedule components, but the handle Ords remain the same.

An ordered collection of timestamped records. Each history is identified by a unique id. Histories can be periodically archived to a remote history database (archive). A history database is a set of histories. History is also used as a scheme in ORDs to refer to collective histories.

The Niagara daemon (niagarad) is a server process used to communicate between Workbench (as a client) and the platform that it is connected to.

1. A connection point between the system and a real or virtual device. Devices become a node when they register with the system, providing a name and connection information. An ORD provides access to the device.
2. A position in a Nav tree hierarchy.
3. The primary organizational unit of a Niagara Analytics Framework data model tree. Like a folder, a node is a container that holds points or other containers. The nodes of the data model provide the structural framework for the model.

An object is the base class required for all system entities that conform to the baja model. Objects group information used to construct a model that includes building devices, virtual devices, individual points, users, system features and services. Objects appear in the Nav tree as files, modules, installers, administrators, copiers, drivers and apps. Metadata associated with objects, including categories, roles (permissions), and hierarchies, provide access control and configuration options to manage automated buildings efficiently.

An ORD is an “Object Resolution Descriptor”. The ORD is the Niagara universal identification system and is used throughout the framework. The ORD unifies and standardizes access to all information. It is designed to combine different naming systems into a single string and has the advantage of being parsable by a host of public APIs.

The palette provides a hierarchical view of available components located in the left side bar pane. You copy and paste or drag a component from a palette to where you need it, for example, to a wire sheet, property sheet, Px View, or the Nav side bar pane.

A component that extends control of point behavior in a consistent manner. Each property of a ControlPoint that exists as a subclass of a PointExtension is considered an extension on the point. Extensions allow plug-in functionality, such as alarming and historical data collection via special hooks that a ControlPoint provides to the PointExtension.

A tool for creating graphical representations of ducting, piping, and the equipment used in a building. The editor allows the creation of PC and mobile views.

A deployable package of Niagara objects used to streamline repetitive configuration steps when making multiple installations with similar functionality. For example, when setting up a new device by deploying a device template, only unique device properties require configuration. Templates are indexed and searchable.