



# Getting Started with Niagara Framework® Software Manual



# PROPRIETARY DATA NOTICE

This document, as well as all reports, illustrations, data, information, and other materials are the property of LG Electronics U.S.A., Inc., and are disclosed by LG Electronics U.S.A., Inc., only in confidence.

⊙ *Do not throw away, destroy, or lose this manual.*

Please read carefully and store in a safe place for future reference.

Content familiarity required for proper installation.

*The instructions included in this manual must be followed to prevent product malfunction, property damage, injury, or death to the user or other people. Incorrect operation due to ignoring any instructions will cause harm or damage.*

*For more technical materials such as submittals, engineering databooks, and catalogs, visit [www.lghvac.com](http://www.lghvac.com).*

For continual product development, LG Electronics U.S.A., Inc., reserves the right to change specifications without notice.

©LG Electronics U.S.A., Inc.

*This document, as well as all reports, illustrations, data, information, and other materials are the property of LG Electronics U.S.A., Inc.*

# CONTENTS

Confidentiality and Trademark Information.....	6
Certifications.....	7
Compliance And Approvals .....	8
Getting started with Niagara .....	9
About this Guide .....	9
Chapter 1 About the Niagara Framework.....	11
About Niagara 4 .....	11
About control systems integration .....	13
About Java .....	14
About common networking and Internet protocols .....	15
About programming for non-programmers.....	15
About systems capabilities.....	15
About component software design .....	16
About the software architecture .....	16
About Niagara building blocks .....	18
Formats (BFormat) .....	35
BFormat errors .....	43
Chapter 2 About Workbench .....	45
Tour of the Workbench GUI .....	45
Workbench window controls .....	48
About the side bar panes.....	49
About popup menus.....	50
Table controls and options.....	50
Controls and options for charts .....	52
Customizing the Workbench environment.....	66
Chapter 3 Data and Control Model .....	100
Wire Sheet object management .....	100
About control points.....	107
About point extensions.....	119
About control triggers.....	125

About point status.....	126
About writable points.....	131
About composites.....	134
<b>Chapter 4 About Workbench tools.....</b>	<b>140</b>
Alarm portal.....	140
Security management tools.....	142
Driver Upgrade Tool.....	142
Embedded Device Font Tool.....	142
Jar Signer Tool.....	145
Kerberos Configuration Tool.....	146
Lexicon Tool.....	146
Local License Database.....	149
Logger Configuration tool.....	150
Lon Xml Tool.....	153
Manage Credentials.....	153
Module Info View.....	155
NDIO to NRIO Conversion Tool.....	156
New Driver Wizard.....	159
New Module Wizard.....	159
New Station wizard.....	160
Request License.....	166
Resource Estimator.....	167
Time Zone Database Tool.....	167
Todo List.....	168
Workbench Fox Analyzer.....	169
Workbench Job Service.....	170
Workbench Library Service.....	171
Workbench Service Manager.....	172
<b>Chapter 5 Component Guides.....</b>	<b>173</b>
Components in alarmRdb module.....	174
Components in backup module.....	174
Components in baja module.....	176
Components in chart module.....	195
Components in control module.....	195

Components in file module .....	197
Components in help module .....	200
Components in net module .....	200
Components in program module .....	201
Components in saml module .....	203
Components in web module .....	209
Components in workbench module .....	217
Chapter 6 Plugin Guides .....	224
Types of plugin modules .....	224
backup-BackupManager .....	224
chart-ResourceManager .....	225
Plugins in html module.....	227
Plugins in ldap module .....	232
Plugins in program module .....	234
raster-RasterViewer.....	238
wiresheet-WireSheet .....	238
Plugins in wbutil module.....	241
Plugins in workbench module.....	247
Chapter 7 Interface reference.....	261
About keyboard shortcuts.....	261
Types of menu bar items.....	262
Types of popup menu items .....	271
Types of side bars.....	281
About the Nav side bar.....	286
About the Search side bar .....	291
Types of edit commands.....	296
Types of toolbar icons .....	296
Types of console commands .....	302
Glossary .....	305
Index.....	307

# **Confidentiality and Trademark Information**

Tridium, Inc.

3951 Westerre Parkway, Suite 350

Richmond, Virginia 23233

U.S.A.

## **Confidentiality**

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation ("Tridium"). Such information and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

## **Trademark notice**

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, and Sedona Framework are registered trademarks, and Workbench are trademarks of Tridium Inc. All other product names and services mentioned in this publication that are known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

## **Copyright and patent notice**

This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc.

Copyright © 2020 Tridium, Inc. All rights reserved.

The product(s) described herein may be covered by one or more U.S. or foreign patents of Tridium.

## **Certifications**

The MultiSITE VM3 controller has the following agency listings, compliances, and certifications:

UL-916, Energy Management Equipment - Edition 4

FCC Part 15, Class B - Federal Communications Commission, with FCC Part 15, Subpart C - WiFi

ICES-003, Class B - Industry Canada Interference-Causing Equipment Standard

RoHS 2 (Restriction of Hazardous Substances), Directive 2011/65/EU



CE Declaration of Conformity (Council Directive 004-108-EC)



ACMA, complies with the requirements of the relevant ACMA Standards. This document covers mounting and wiring of the following products.

# **Compliance And Approvals**

## **Federal Communications Commission (FCC)**

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. This device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

## **Canadian Department of Communications (DOC)**

This device complies with Industry Canada License-exempt RSS standard(s). Operation is subject to the following two conditions: 1) this device may not cause interference, and 2) this device must accept any interference, including interference that may cause undesired operation of the device.

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (EIRP) is not more than that necessary for successful communication.

The device for operation in the band 5150–5250 MHz is only for indoor use to reduce the potential for harmful interference to co-channel mobile satellite systems.

## **Approved Antenna Listing**

- ANT-DB1-RAF-RPS

## **Transmitter Module Listing**

- Contains Transmitter Module FCC ID: W98-12977
- Contains Transmitter Module IC: 8339A-12977

To comply with FCC and Industry Canada RF exposure limits for general population /uncontrolled exposure, the antenna(s) used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.



# **GETTING STARTED WITH NIAGARA**

## **About this Guide**

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. In order to make the most of the information in this book, readers should have some training or previous experience with Niagara 4 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.

## **Related Documentation**

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

### Related Documents

<b>Document Title</b>	<b>Description</b>
<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.

# **CHAPTER 1 ABOUT THE NIAGARA FRAMEWORK**

## **Topics covered in this chapter**

- ◆ About Niagara 4
- ◆ About control systems integration
- ◆ About Java
- ◆ About common networking and Internet protocols
- ◆ About programming for non-programmers
- ◆ About systems capabilities
- ◆ About component software design
- ◆ About the software architecture
- ◆ About Baja
- ◆ About Niagara building blocks
- ◆ Formats (BFormat)

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.

Niagara 4 is a UX framework that provides an infrastructure that enables systems integrators and developers to 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. The framework supports the use of tags that can be queried, thus providing a foundation for many of the new features (search, tagging, relations, templates, hierarchies). The framework also includes a cutting-edge toolset that enables non-programmers to build rich applications in a drag-and-drop environment.

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

## **About Niagara 4**

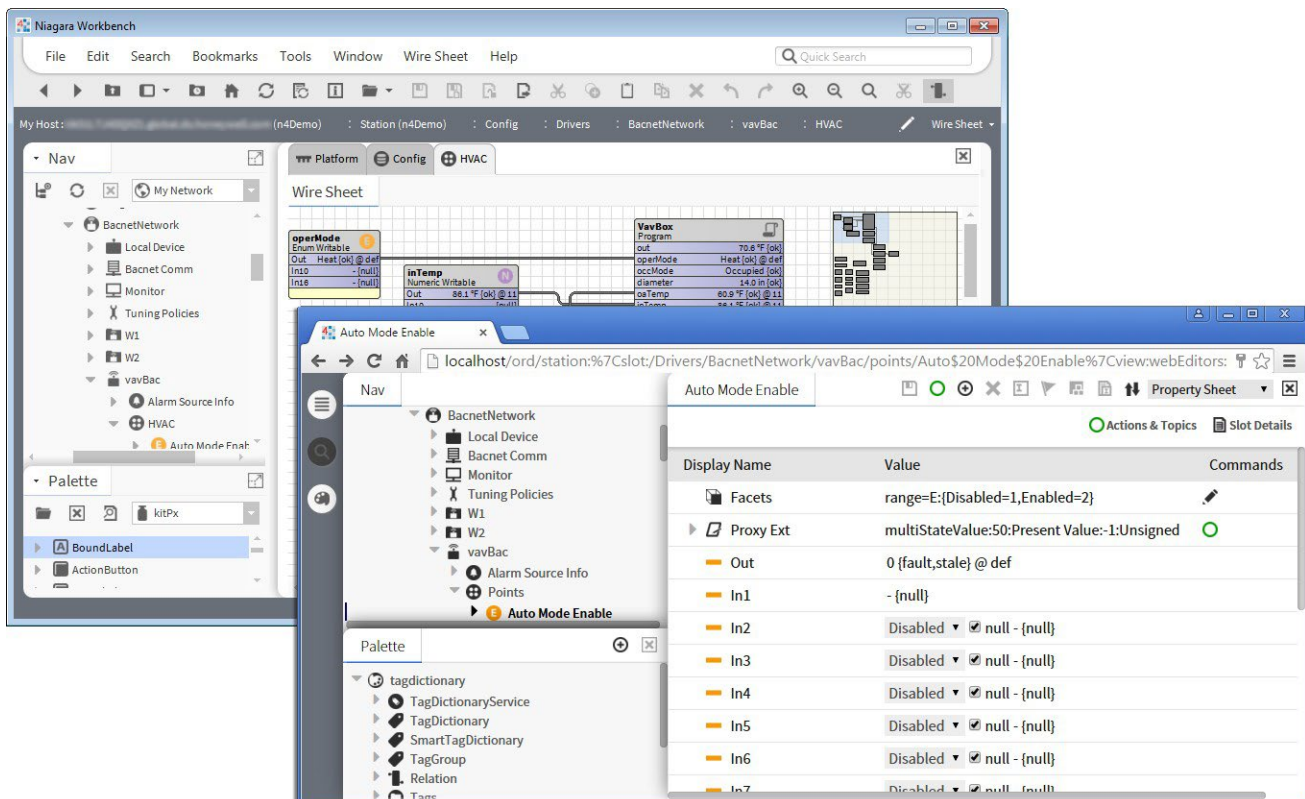
The user experience (UX) is at the heart of Niagara 4. The Niagara 4 “bajaux” framework is based on open web technologies, such as HTML5, CSS3, JSON, and Bajascript v2.0.

The bajaux framework effectively provides an improved user experience in utility, 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 in order to accommodate changing usage circumstances and changes to individual systems over time. The improvements of this new framework can be summed up in the following categories: visualization improvements, tagging and templating, and advanced security.

## Visualization improvements

The cutting-edge 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 in order to drive efficiencies can utilize the dashboard and web chart reporting capabilities. Developers can easily create dynamic, interactive applications and views using bajaux widgets which display across Niagara media, such as Workbench or in a modern web browser (no browser plug-in required). Search functionality integrated in Workbench enables quickly locating data to drop into other views. Optimized workflows for common tasks require fewer clicks and provide more intuitive interaction. Additionally, the new look and feel of the interface incorporates a clean and crisp design, as shown here, with a subdued and focused use of col- or to emphasize important information.

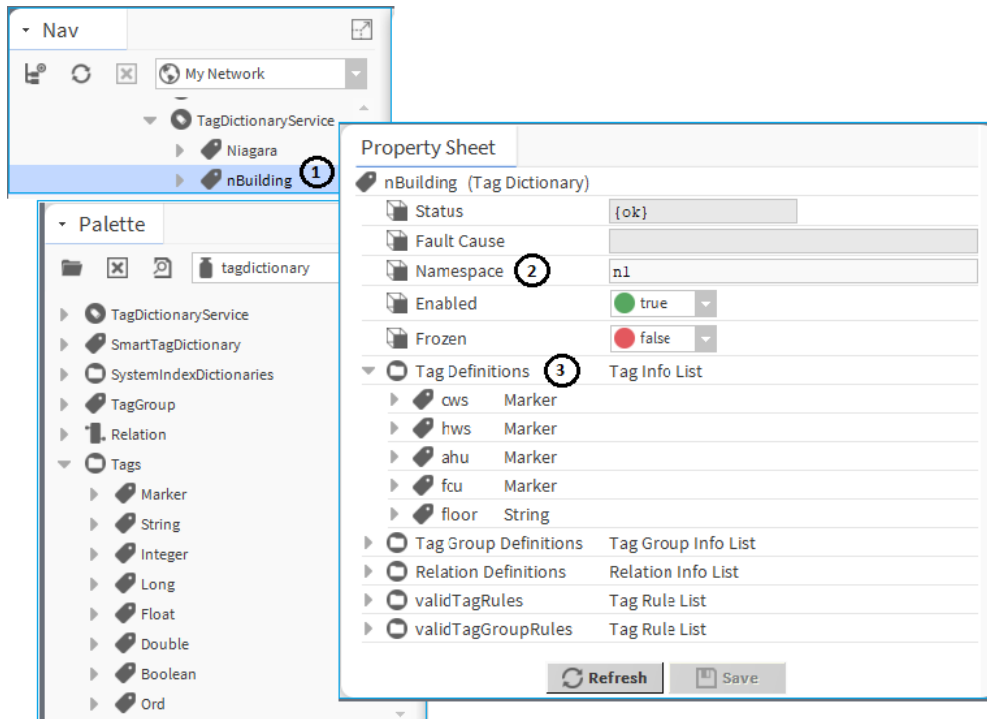
Figure 1: Workbench interface (left) compared with web browser view (right)



## Tagging and templating

Systems integrators will find that metadata tagging at the component level enables a number of ways to find data, via search and navigation hierarchies, as well as providing ways to narrow results via filtering. Similarly, creating templates using pre-tagged devices results in built-in reusability which, of course, translates to shorter integration time. Using tag-based hierarchies, multiple navigation schemes can easily be created for different user roles. Also, there is no need to update Nav files every time new devices are added to a system.

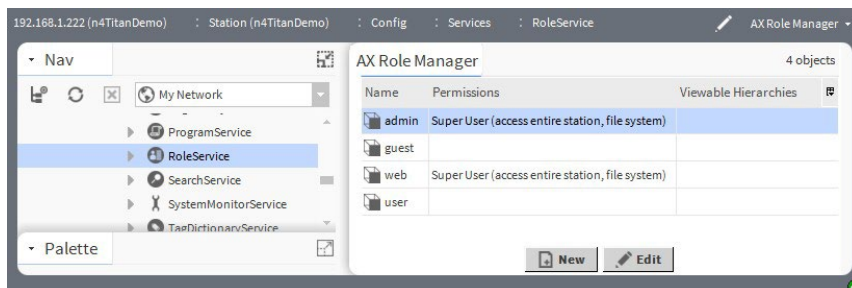
Figure 2: Custom tag dictionary (1), tag definitions in Property Sheet (3), and tagdictionary palette (lower left)



## Advanced security

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

Figure 3: Role Manager view



## About control systems integration

Using the Niagara Framework, control systems integration means:

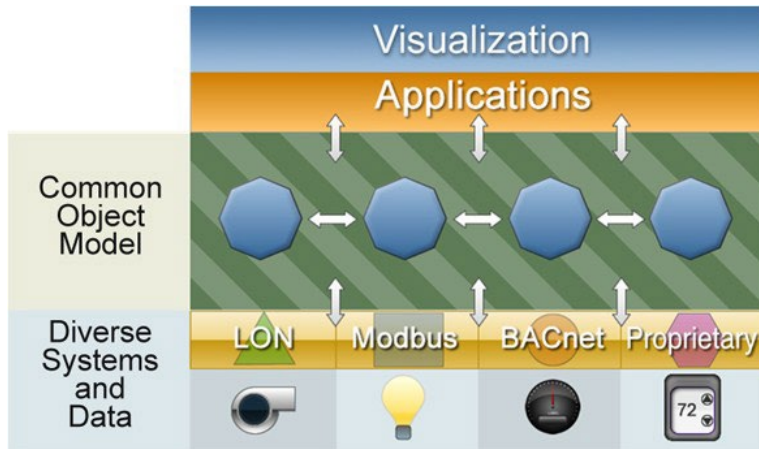
1. connecting devices on a common communications media
2. modeling those devices in software
3. programming applications to use the information in those devices

Before a device, such as a chiller, VAV box, or temperature sensor can be used, information from those devices must be pulled into the system software.

Niagara then models those 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 Niagara 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 Niagara 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, you can use the objects to build control system solutions quickly and efficiently.

Figure 4: Niagara common object model



The Niagara 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.

## About Java

Much of the Niagara 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 needs to run on some other processor, the program has to be compiled again. Java, on the other hand, compiles once. Niagara software runs on embedded JACE 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.

## About Virtual Machines

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 Niagara 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.

## About common networking and Internet protocols

Niagara is designed from the ground up to assume that there will never be any one “standard” network protocol, distributed architecture, or fieldbus. The design goal is to integrate cleanly with all networks and protocols. The framework standardizes what’s inside the box, not what the box talks to.

The Niagara 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 a wide range of 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.

## About programming for non-programmers

Most features in the Niagara Framework are designed for dual use (for programmers as well as non-programmers). These features are designed 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.

## About systems capabilities

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

### About embedded systems capabilities

Niagara is one of the only software stacks designed to run across the entire range of processors from small embedded devices to server class machines. Niagara 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 Niagara-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 Niagara software. Systems of this size span a wide range of network topologies and usually communicate over unreliable Internet connections. Niagara is designed to provide an infrastructure for managing systems of this scale.

## About component software design

Niagara uses an architecture centered around the concept of “Component Oriented Development.” A component is a piece of self-describing software that can be assembled like building blocks to create new applications.

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. Niagara supports custom development and extension of the framework. Niagara components are extensible and go beyond data and protocols to unify the entire development environment.

## About the software architecture

The following images illustrate the Niagara software subsystems and the software processes and protocols, respectively.

Figure 5: Niagara software subsystems

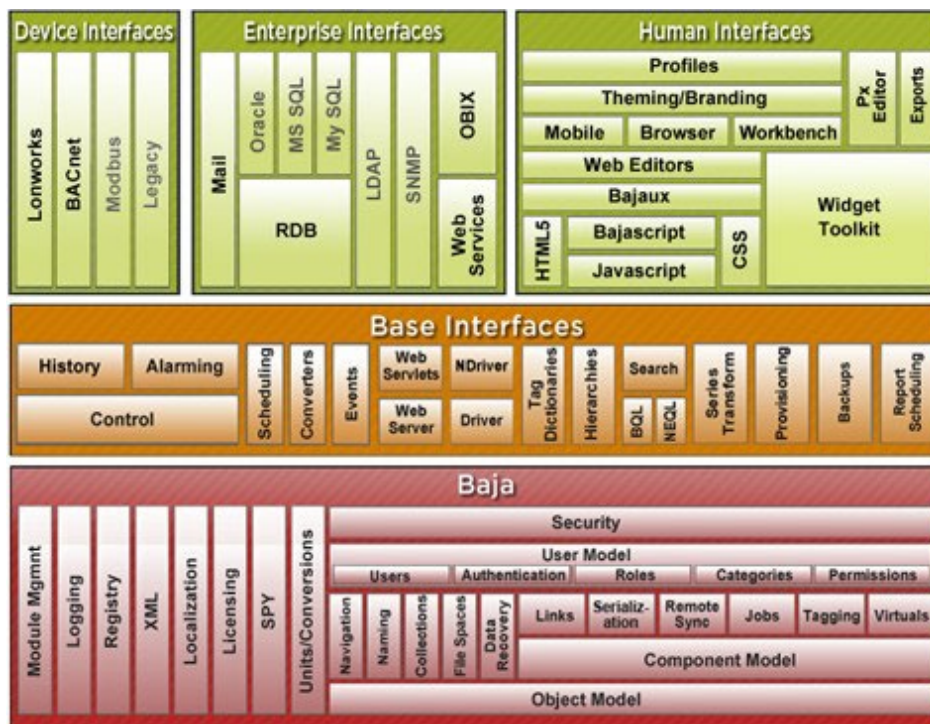
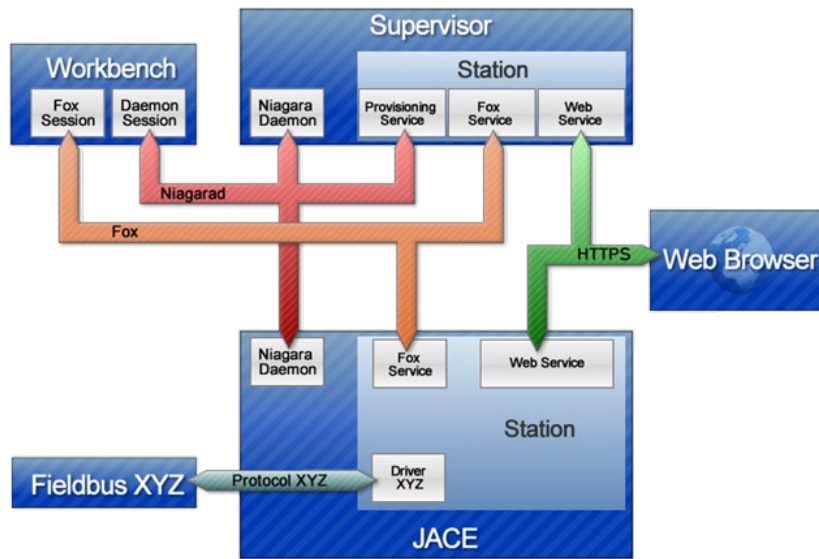




Figure 6: Niagara software processes and protocols



## About 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.

## About APIs

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

Many features found in the Niagara 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 (see About Baja section for more about information) 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 Niagara implements the Baja APIs. The com.tridium code may or may not be documented. If com.tridium APIs are publicly documented then Tridium encourages developers to use them, but does 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.

## About Niagara building blocks

This section describes some of the fundamental building blocks of the Niagara 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:

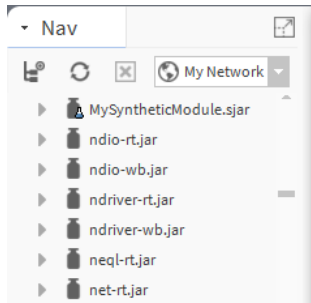
- Modules — the most basic unit of the software that comprises Niagara.
- Components — the primary building blocks of the Niagara framework.
- Presentation XML (Px) — an XML file format that defines how Niagara visualizes information (text, graphics, alarms, and so on) across diverse media, such as: Workbench, web browsers, and mobile devices.
- Stations — the main unit of server processing in the Niagara 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 Niagara 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 — Niagara 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 Niagara architecture.

Major releases of the Niagara 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 side bar pane.

Figure 7: Module listing in the nav side bar tree



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

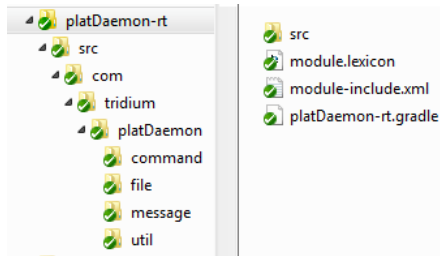
Niagara uses the module version number to make sure that you have the latest available module. Versions are specified as a series of whole numbers separated by a period, for example 4.0.16. To understand which version of a module is more recent, simply observe the module version number. For example, 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 8: Directory structure used to build the module's jar



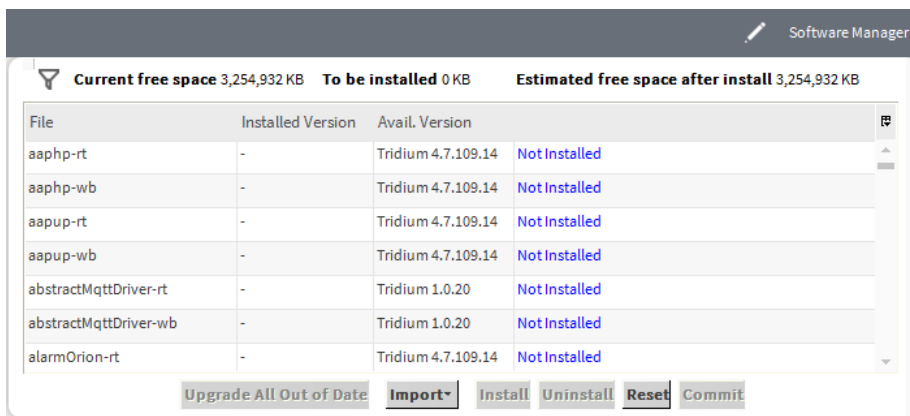
## Benefits of modular software development

Modular software development provides several benefits, including the following:

- Improves tracking deployment and the assigning of version numbers

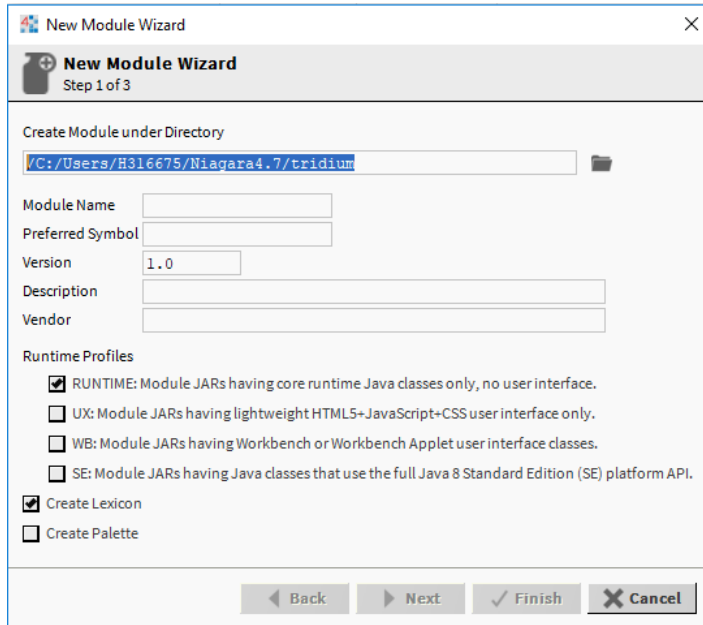
For example, if a new driver is available to be added to Niagara Framework, it can be packaged, delivered, and added in as a single module or with multiple modules, using the platform Software Manager view. Refer to the Niagara Platform Guide.

Figure 9: Software manager view



- Requires less space on the controller — Niagara's modular development allows you to save space on your JACE 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 10: New module wizard



## About components

A component is the primary building block that you use to engineer an application using Workbench. As described in the section, “About component software design”, components provide 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.

## About slots

Niagara 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 11: Slot details for a single component

Display Name	Commands	Name	Flags	Type	Facets
Facets		facets	(none)	baja:Facets	(none)
Proxy Ext		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 Niagara 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 (e.g: %, &, ., #, and so on) are illegal in component names. For further details, see the section “Escaped names”, which covers 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`

### 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.

## Escaped names

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

Further, various drivers have learn features to automate the creation of points, some of which (by default) may also have such 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.

In any case, 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 presentation

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

Presentation is a term that is used to describe how the software framework provides visualization of information across different types of media. The terms information, visualization, and media may comprise the following:

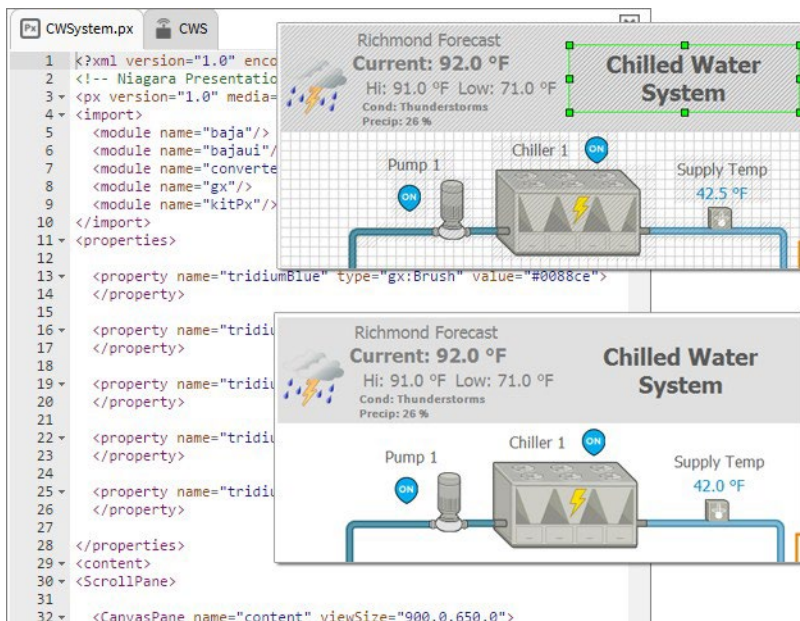
- information
  - real-time data
  - historical data
  - configuration data
  - alarm data
  - scheduling data
  - graphical data
  - textual data
- visualization
  - graphics (bitmap, JPG, PNG, SVG, vector)
  - videos
  - text documents
  - tables
  - charts
  - dashboards
  - input controls (actions, field editors, text fields, check boxes, trees)
- media
  - web browsers (HTML5, CSS3, JavaScript)

- Workbench (Niagara stack)
- mobile devices
- csv
- pdf
- svg
- xml
- printed pages

### About presentation xml (Px)

An XML file format is used to define the Niagara presentation. This file format is called “Px” for presentation XML and the term “Px” is commonly used to describe the Niagara presentation architecture. Presentation is a term that describes how Niagara visualizes information (text, graphics, alarms, and so on) across diverse media – such as: Workbench, web browsers, mobile devices, and so on. Niagara uses “presentation xml” (Px) to accomplish this. 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 12: Px file in Text Editor (left), Px Editor (top right), and Px Viewer





## About presentation modules

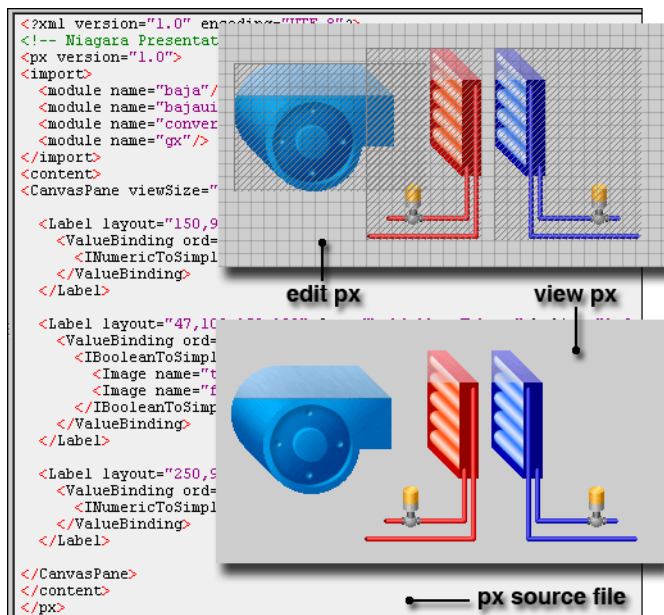
Niagara's presentation architecture is based on many modules and their associated public APIs:

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

## About presentation xml (Px)

An XML file format is used to define the Niagara presentation. This file format is called "Px" for presentation XML and the term "Px" is commonly used to describe the Niagara presentation architecture. Presentation is a term that describes how Niagara visualizes information (text, graphics, alarms, and so on) across heterogeneous media – such as: Workbench, desktop browsers, handheld devices, and so on. Niagara uses "presentation xml" (Px) to accomplish this. 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 13: Px file in Text Editor, Px Editor, and Px Viewer



## About presentation design philosophy

The presentation architecture is based on the following design principles:

- **Component model** — the core philosophy of every subsystem in Niagara is to build atop the component model. The presentation architecture is no exception. The design embraces a pure component model solution. The component model is used for the normalized representation of presentation - the “document object model” (DOM) of the Niagara presentation 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 using the Workbench, as an HTML web page, or as a PDF file. All presentations are stored in the normalized component model as Px files.

## About presentation media

Niagara supports four primary target media.

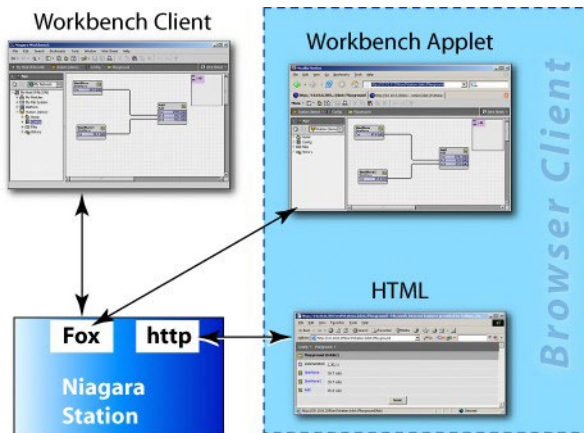
- **Workbench**— the Niagara stack must be available to take full advantage of the presentation architecture. The desktop version of Workbench provides a stand alone application that can render presentations with their full power. With the WbProfile feature, new custom applications can easily be built using Workbench and its presentation engine.
- **Niagara 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 the 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 will not work in Web Launcher. An applet contained within an HTML frame, for instance, cannot be supported in Web Launcher.

- **PDF** — Adobe's PDF format is the standard way to export a presentation to print it. 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 which allows you to edit properties and invoke commands right in the view. Other HTML5 functionality includes context sensitive menus, 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 toolbar or in a modern HTML5-capable web browser.

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

Figure 14: Presentation media options



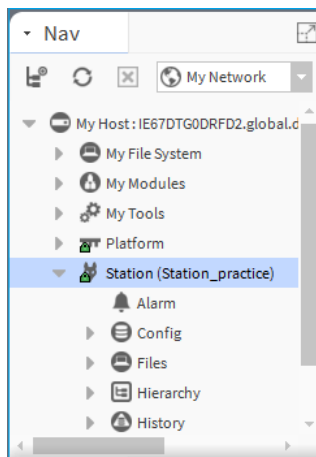
## About stations

A station is the main unit of server processing in the Niagara 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 JACEs). 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 Niagara Framework and provides access for client browsers to view and control these components. The primary parts of a station include components and services. It is the combination of a database, a web server, and a control engine. The station either runs on a Web Supervisor PC or the JACE controller.

Figure 15: Station in nav tree

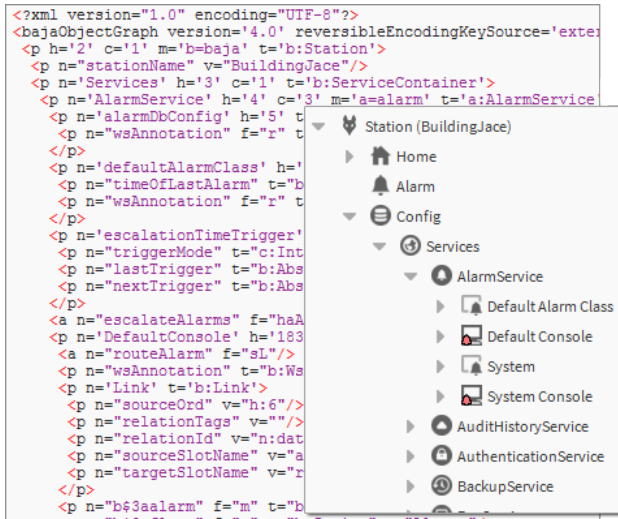


A system can be a single station or multiple stations depending on the size of the project and it is defined by a bog file.

## About BOG files

A bog file (Baja object graph) contains Niagara 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 16: Sample bog file and Nav tree

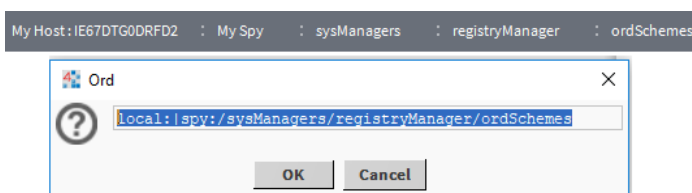


## About ORDs

An ORD is an “Object Resolution Descriptor”. The ORD is the Niagara universal identification system and is used throughout the Niagara 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.

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 17: 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 18: 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:”, or “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

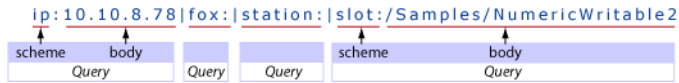
- `ip:somehost|fox:|station:|slot:/MyService`
- `ip:somehost|fox:|station:h:/42`
- `ip:somehost|fox:|file:/C:/dir/file.txt`
- `local:|file:!jre/lib/logging.properties`
- `local:|module://icons-ux/x16/cloud.png`
- `local:|spy:/`

In Niagara 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>”.

Figure 19: Example ORD scheme and body



- **scheme** — the scheme name is a globally unique identifier which specifies, in Niagara, how to find a piece of code to lookup an object from the body string. Refer to Types of schemes, for a listing of different types of schemes.
- **body** — the 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 lookup 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 is used to identify 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 is used to identify 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.

## Types of schemes

A scheme is a globally unique identifier which specifies, in Niagara, how to find a piece of code to lookup an object.

- **ip:**

The “ip” scheme is used to identify an Ip Host instance. Ords starting with “ip” are always absolute and ignore any base that may be specified. The body of a “ip” query is a DNS hostname or an IP address of the format “dd.dd.dd.dd”.

- **fox:**

The “fox” scheme is used to establish a Fox session. Fox is the primary protocol used by Niagara for IP communication. A “fox” query is formatted as “fox:” or “fox:<port>”. If port is unspecified then the default 1911 port is assumed.

- **file:**

The “file” scheme is used to identify 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 Niagara installation directory identified via `Sys.getBajaHome()`.

- User Absolute: “^config.bog”

User absolute paths are 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.

- Relative: “myfile.txt”
- Relative with Backup: “./myfile.txt”

- module

The “module” scheme is used to access BIFiles 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 also. 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:

The “station” scheme is used to resolve the BComponentSpace of a station database.

- slot:

The “slot” scheme is used to resolve a BValue within a BComplex by walking down a path of slot names. Slot queries always parse into a SlotPath.

- h:

The “h” scheme is used to resolve 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:

The “service” scheme is used to resolve a BComponent by its service type. The body of the query should be a type spec.

- spy:

The “spy” scheme is used to navigate spy pages. The javax.baja.spy APIs provide a framework for making diagnostics information easily available.

- bql:

The “bql” scheme is used to encapsulate a BQL query.

- nspace:

The “nspace” scheme is used to provide 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 the Niagara Virtual,  
example- station:|slot:/Drivers/NiagaraNetwork/DeviceName/virtual|virtual:/a/b/c

- nspace:

The “nspace” scheme is used to provide 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 Niagara Virtual,  
example- station:|slot:/Drivers/NiagaraNetwork/ControllerName/virtual|virtual:/a/b/c

- sys:

The “sys” scheme is used to allow queries against the systemDb.

NOTE: It supports NEQL only.

- Query entire systemDb for all devices in the system,  
example- ip:<SupervisorIP>|foxs:|sys:|neql:n:device
- Scoped query against systemDb for ModbusTCP devices in subordinate station,  
example- ip:<SupervisorIP>|foxs:|station:|slot:/Drivers/ModbusTCPNetwork|sys:|neql:n:device

- single:

The "single:" ORD scheme 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.

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

## Types of space

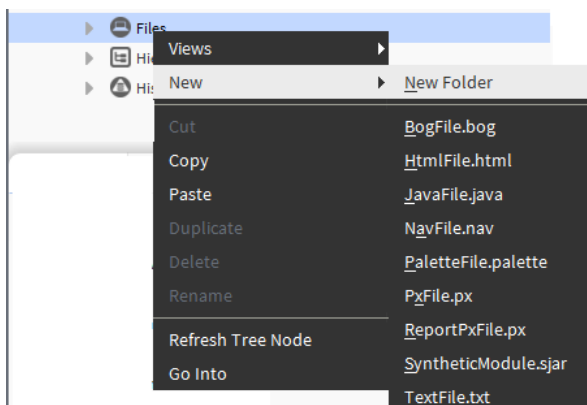
Space defines a group of objects that share common strategies for loading, caching, lifecycle, naming, and navigation. Following, is a list of some of the different types of space:

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

## 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:

Figure 20: Types of files available from the New menu



- BogFile.bog

Bog files are database files.



- HtmlFile.html

Html files are edited in the Text File Editor view and viewed in the Html View.

- JavaFile.java

Java files are edited in the Text File Editor view.

- NavFile.nav

Nav files are edited in the nav file editor and viewed in the nav tree. Refer the NiagaraAX Graphics Guide for more information about nav files.

- PaletteFile.palette

These files are custom collections of components that you create and save for viewing in the palette side bar.

- PxFile.px

Px files are edited in the Px view and are used to store graphic presentations that are available for viewing in the Px viewer and in a browser.

- ReportPxFile.px

The ReportPxFile is a regular Px file that has been “pre-loaded” with a ReportPane widget. The reporting functionality helps you design, display, and deliver data to online views, web browser, and various other formats.

- TextFile.txt

Text files are edited and viewed in the text file editor.

## About file naming

When working in a station, you often create various 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), we strongly recommend that you restrict all characters in file and folder names to ones in the “original” set of ASCII characters in the BNF for Niagara file paths, namely:

```
a-z | A-Z | 0-9 | specials  
specials= space | . | : | - | _ | $ | + | ( | ) | & | ' | ' | @ | [ | ]
```

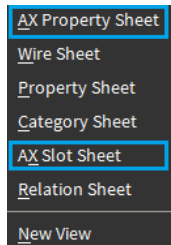
Otherwise, use of other characters, for example, a tilde ( ~ ) in file or folder names may result in obscure problems, particularly on JACE platforms (QNX-based hosts).

## About Views

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 side bar. In addition, you can right-click on an item and select one of its views to display in the view pane. You can see many different views of components. For example, a component that appears in the Nav tree may have a Wire Sheet view, a Property Sheet view, and a Px View, that all display 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.

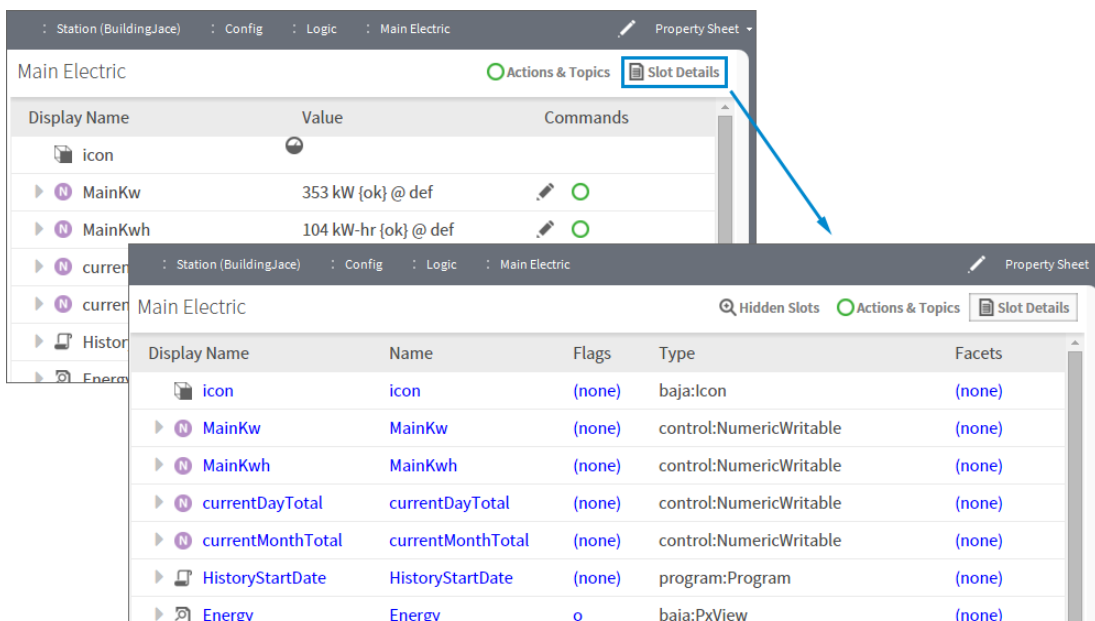
Two AX views are available under the Views selector menu and the right-click pop-up menu (as shown below): the AX Property Sheet view, and AX Slot Sheet view.

Figure 21: AX views under Views selector



Optionally, you can use the Niagara 4 Property Sheet view which combines the functionality of the two views. When working in this view, simply click the Slot Details toggle command (upper right) to display and edit slot sheet details, as shown below. Enhanced functionality in this view also allows you to invoke actions and edit properties.

Figure 22: Property Sheet view with enhanced functionality



Refer to Chapter 6, Plugin Guides for a comprehensive list of views.

## About lexicons

Niagara provides non-English language support by use of lexicons. Lexicons are identified by two-digit Java locale codes, such as “fr” (French) or “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, see the *Niagara Lexicon Guide*.

## Formats (BFormat)

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

In Niagara, 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>")`

## Example scenarios

These example scenarios demonstrate how to configure text properties for which no default BFormat scripts exist.

Alarm extensions, history extensions, Px Widgets, and **WeatherService** provide good examples to illustrate the use of BFormat scripts.

### BFormat example: naming points, VAV scenario

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

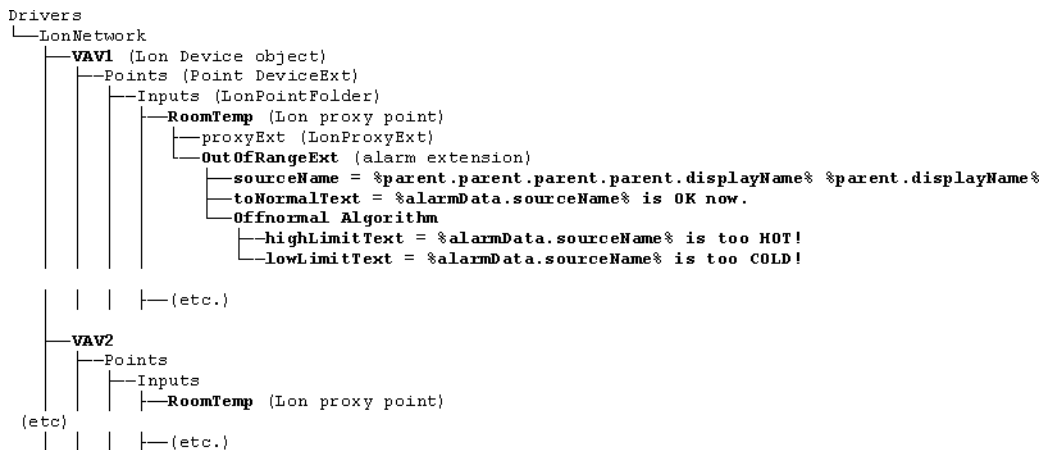
This would involve entering unique names for all BFormat-capable properties under each alarm extension. In some cases, you may also have to modify related properties under the device's `offNormalAlgorithm` slot.

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 23: 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 (VAVn). The variable on the right resolves to the proxy point name, in this example, RoomTemp. In the alarm console the alarm source displays as VAVn 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.

## 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.

### History extension scenario

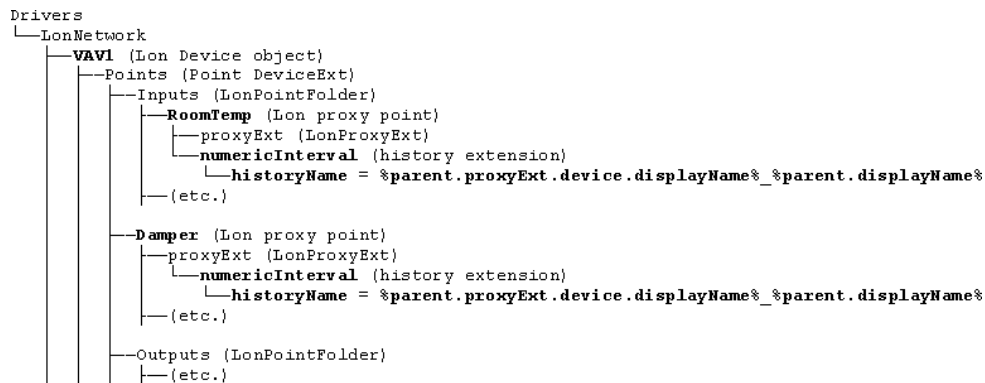
Consider a VAV network with replicated device applications. In each VAV zone, you need the histories of several proxy points, all identically named. For example:

- 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.

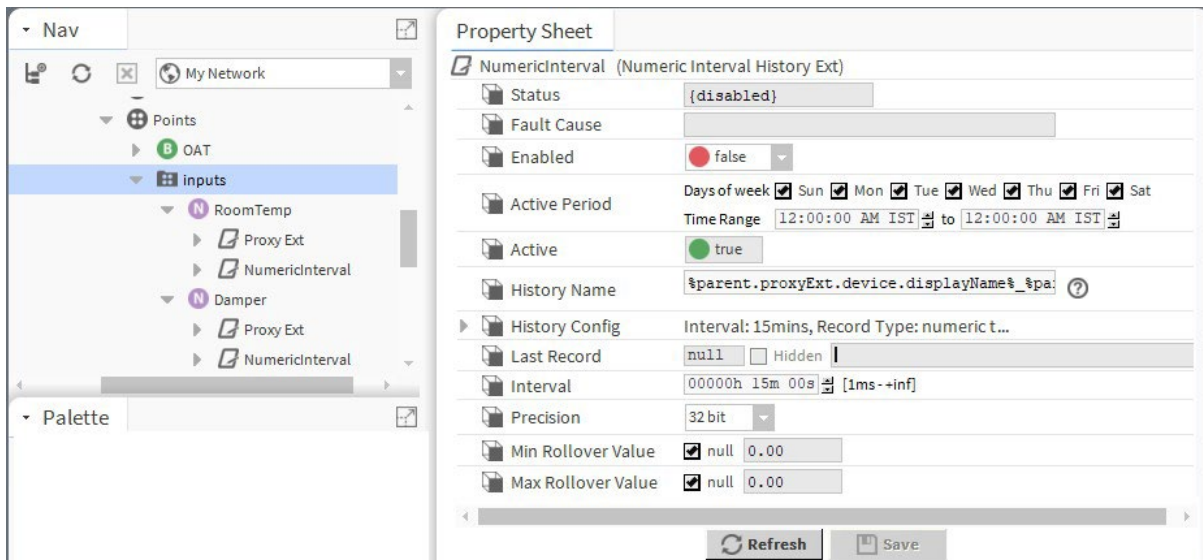
To automatically create unique histories names, before replicating this VAV application, you add a BFormatto the `historyName` in all history extensions, similar to editing the `sourceName` when naming points.

Figure 24: 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 25: 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.

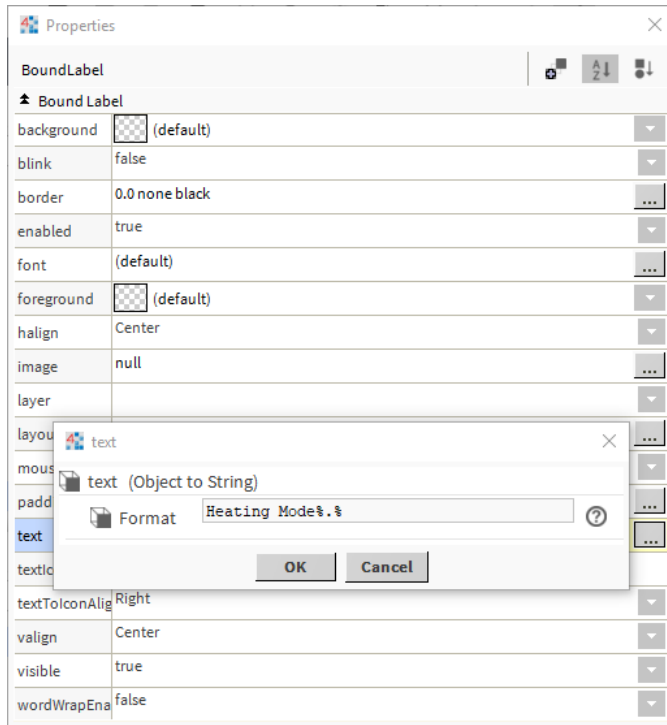
## BFormat Px Widget examples

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 26: 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 outvalue 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.
- `@jpriorityLevel>` 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.

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)
<code>%out.value%</code>	Value only (with facets).	On	AHU is %out.value% AHU is On.
<code>%out.status%</code>	Status including priority level, if writable point.	{ok} @ 16	Status of AHU is %out.status%. Status of AHU is {ok} @ 16.
<code>%activeLevel%</code>	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.
<code>%status.flagsToString%</code>	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. 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).

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.

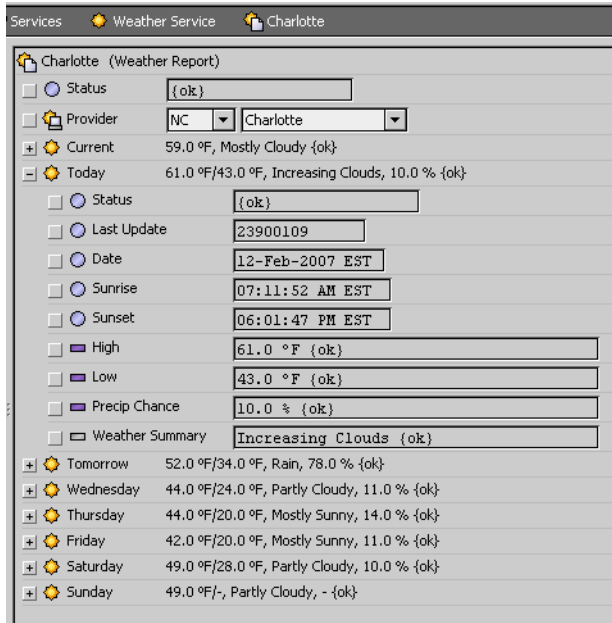
## BFormat: WeatherService example

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 27: Example Weather Report property sheet

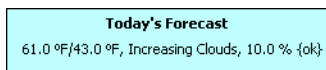


To display this information on a graphic you create a bound label that references the WeatherService and the applicable property. For example, to display the forecast for today in Charlotte, NC, the referenced ord would be `station:|slot:/Services/WeatherService/Charlotte/day0`.

### 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 28: 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 29: 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 (%%).

## BFormat Errors

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, Niagara 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 `BEnumSetSchedule`.

## 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.

This table lists examples of the components with default scripts.

Default BFormat scripts for a few properties

Component	Property	Default Script	Notes
Alarm extension for points, e.g. <code>OutOfRangeExt</code> , etc.	<code>sourceName</code>	<code>%parent.displayName%</code>	Suitable as is in many cases, such as where all parent points are uniquely named.
History extension for points, e.g. <code>NumericInterval</code> , etc.	<code>historyName</code>	<code>%parent.name%</code>	
Any network component's <code>AlarmSourceInfo</code> slot.	<code>sourceName</code>	<code>%parent.displayName%</code>	Often both properties work with these default values.
Any device-level component's <code>AlarmSourceInfo</code> slot.	<code>sourceName</code>	<code>%parent.parent.displayName% % %parent.displayName%</code>	
<code>EmailRecipient</code>	<code>subject</code>	Niagara Alarm From <code>%alarmData.sourceName%</code>	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.

# CHAPTER 2 ABOUT WORKBENCH

## Topics covered in this chapter

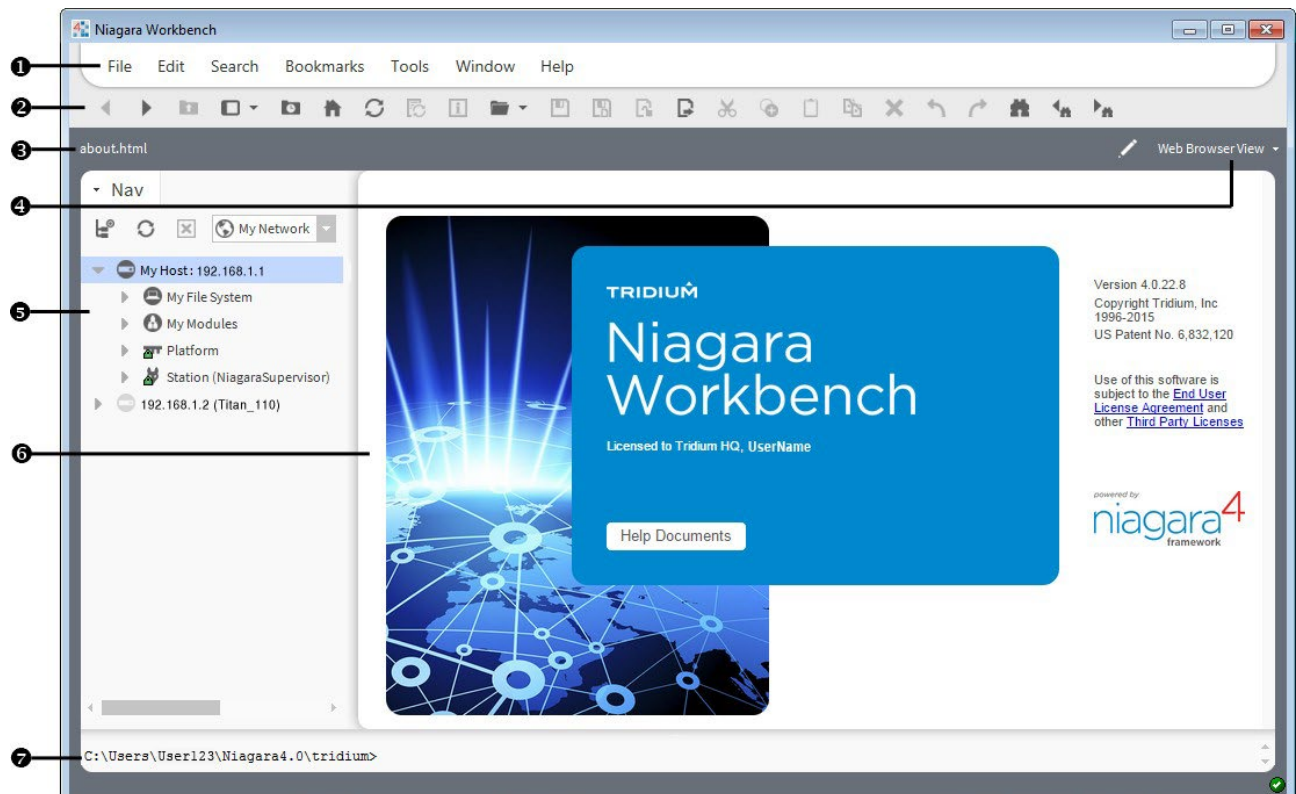
- ◆ Tour of the Workbench GUI
- ◆ Workbench window controls
- ◆ About the side bar panes
- ◆ About popup menus
- ◆ Table controls and options
- ◆ Controls and options for charts
- ◆ Customizing the Workbench environment

Workbench is the term for the Niagara graphical user interface. The following sections describe the general layout and many of the features of Workbench.

## Tour of the Workbench GUI

When you start Workbench, the home window opens.

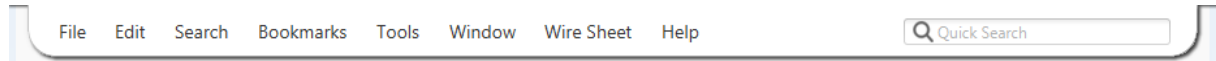
Figure 30: Workbench home window



The home window is divided into seven areas:

- 1 Menu bar — contains available program menus.

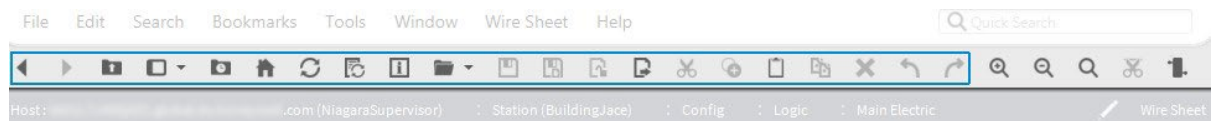
Figure 31: Menu bar



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 toolbar 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, toolbar icons provide single-click access to many of the most commonly used features of the Workbench.

Figure 32: Toolbar

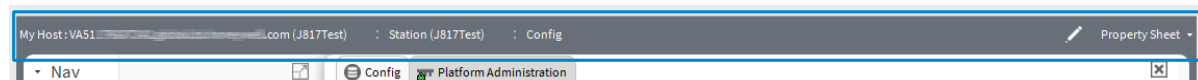


The primary icons (highlighted in the above image) are always visible. Additional sets of icons are added to the toolbar when you select certain views. For example, when the Wire Sheet view is active, the Delete Links icon and Zoom icons are available.

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 33: Pathbar

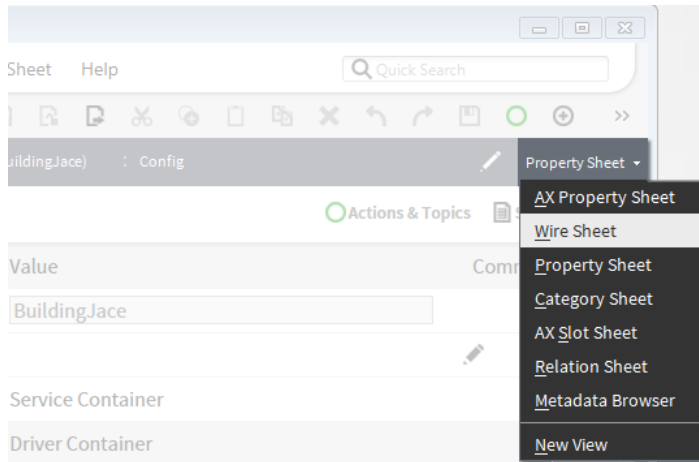


The left side of the path bar shows your current location (Ord or web address). The View selector appears on the right side.

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 34: 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:

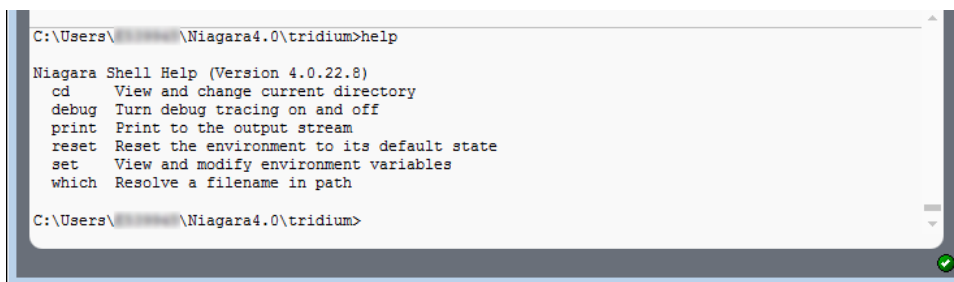
- Double-click on an item in the Nav tree.
- Select a view or action from a Nav tree palette menu.
- Select an option from a menu or submenu.
- Select an option from the locator bar.

The thumbnail view, when active, appears in the top right corner of the wire sheet and 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 35: Console

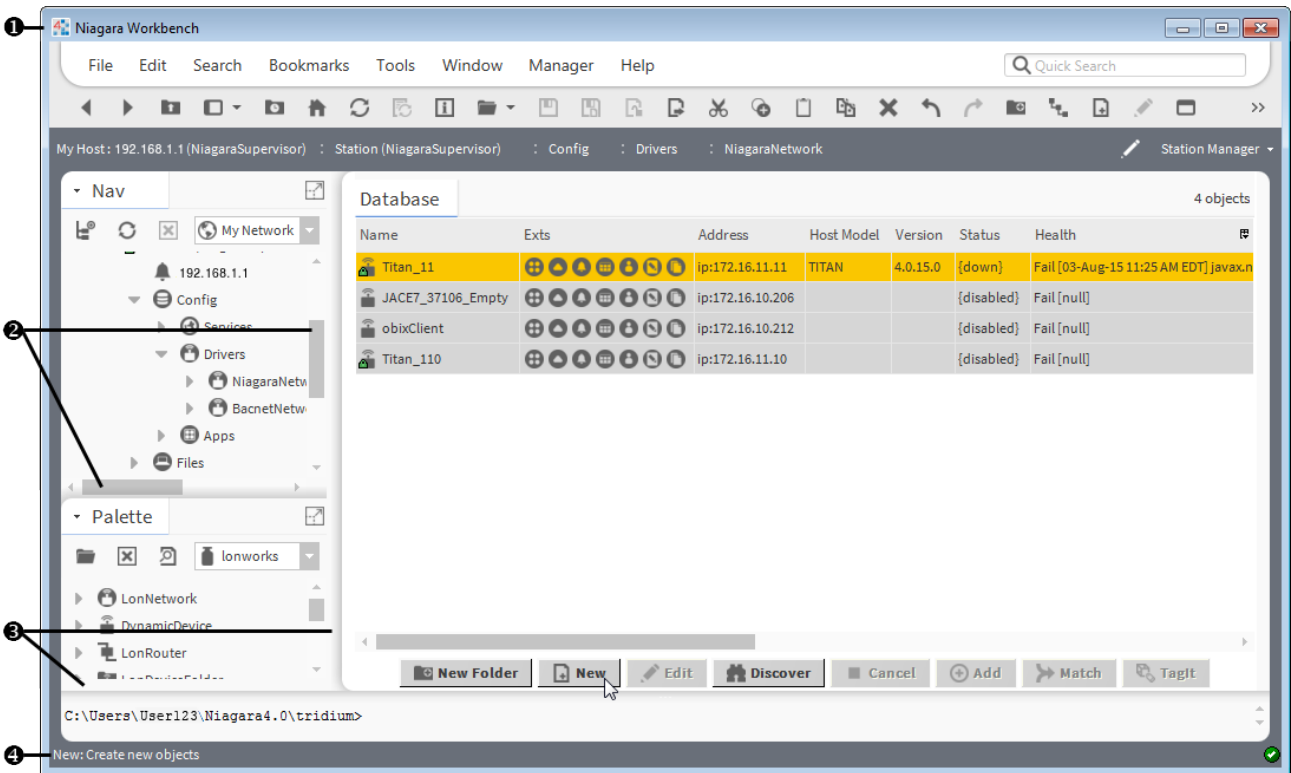


The console has scroll bars on the right side and the window size may be adjusted by dragging the top border bar. From the console you may type in commands directly, including the help command for additional help.

## Workbench window controls

The Workbench interface provides typical Windows-type controls plus other features unique to Niagara.

Figure 36: Window controls



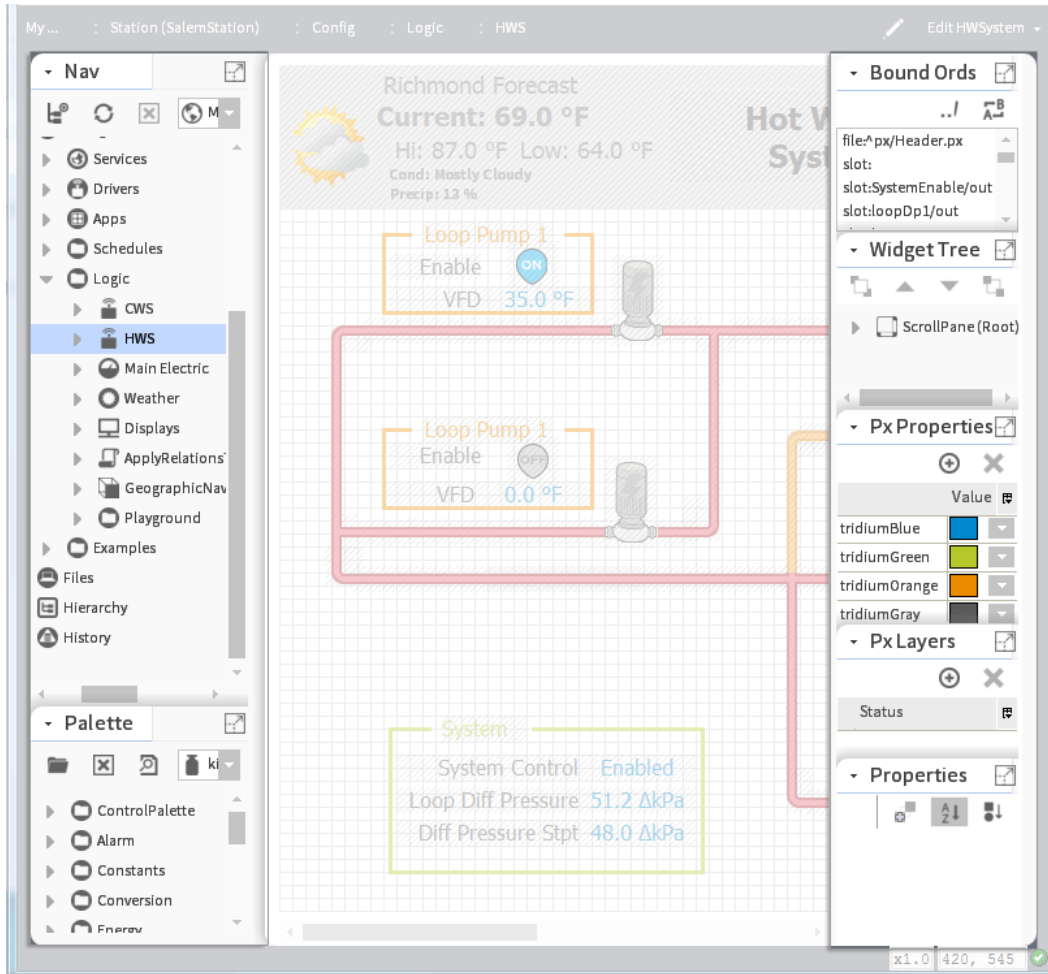
1	<p>Title bar — has standard Windows title bar features, including windows name and icons to minimize,maximize, and close. Double-click the title bar to toggle between maximized and a sizable window.</p> <p>You may create additional windows after starting Workbench—all have these basic features.</p>
2	<p>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.</p> <p>Simply drag a scroll slider (highlighted area) to scroll quickly. Or, click an ending scroll arrow to move in incrementally.</p>
3	<p>Border controls — as needed, drag any outside border to 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.</p>
4	<p>Status bar — at the bottom left of the Workbench window is a status bar.</p> <p>The status bar 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.</p>



## 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.

Figure 37: Side bar panes



Two types of side bar panes are:

- Primary side bar pane

The primary side bar pane is the first area on the left, below the locator bar.

- Px Editor side bar pane

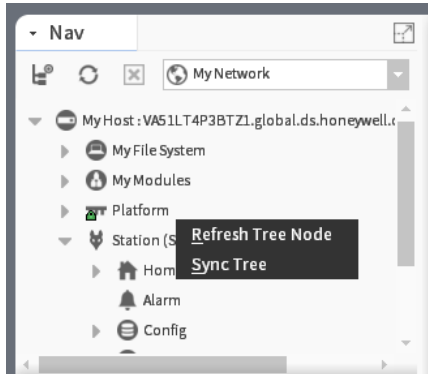
This side bar pane appears on the right side of Workbench and is only available when the Px Editor is active.

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 side bars. To resize side bar height and width, click and drag on the borders.

## About popup menus

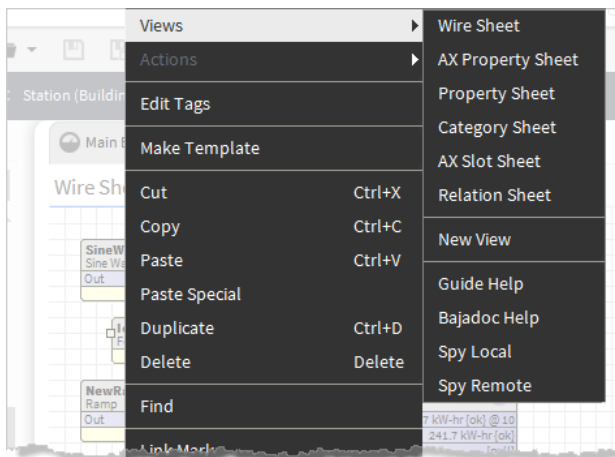
Workbench provides context-specific popup menus for controlling system options. Right-clicking on a component or a view opens these popup menus. The contents of each menu vary depending on the context.

Figure 38: Nav side bar popup menu with nothing selected



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 39: Wire sheet popup menu



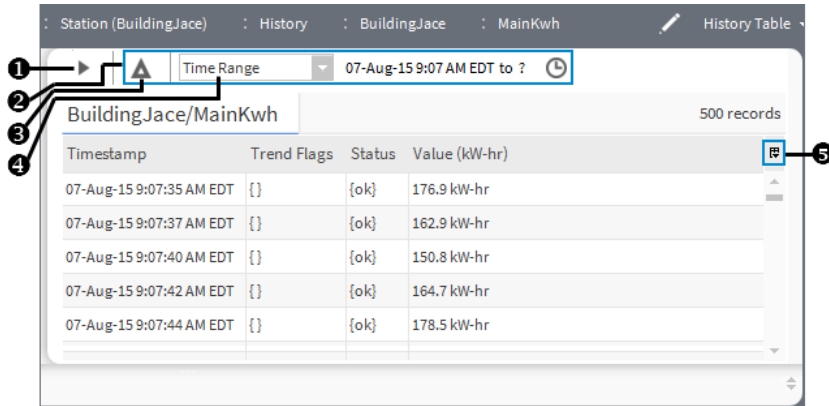
Right-clicking on the Wire sheet displays a menu with different options.

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

## Table controls and options

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

Figure 40: Table controls and options



<b>1</b>	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.
<b>2</b>	Data parameters — These controls include Delta (for history logging) and Time Range settings.
<b>3</b>	Delta — Useful for history logging, displays value changes (delta) in your table.
<b>4</b>	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 resize 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.
<b>5</b>	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: <ul style="list-style-type: none"> <li>• 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).</li> <li>• Export — opens the Export window where you can choose to export the table to PDF, text,HTML, or CSV (comma separated variable).</li> <li>• Context-sensitive menu items — additional context-sensitive menu items appear depending on the component that you are viewing.</li> </ul>

## Editing multiple rows in a table

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

- Step 1 Select the rows in the table to process using the Ctrl or Shift keys while you click the desired rows.
- Step 2 To display the popup menu of available controls and actions, right-click.
- Step 3 Select the control or action.
- Step 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.

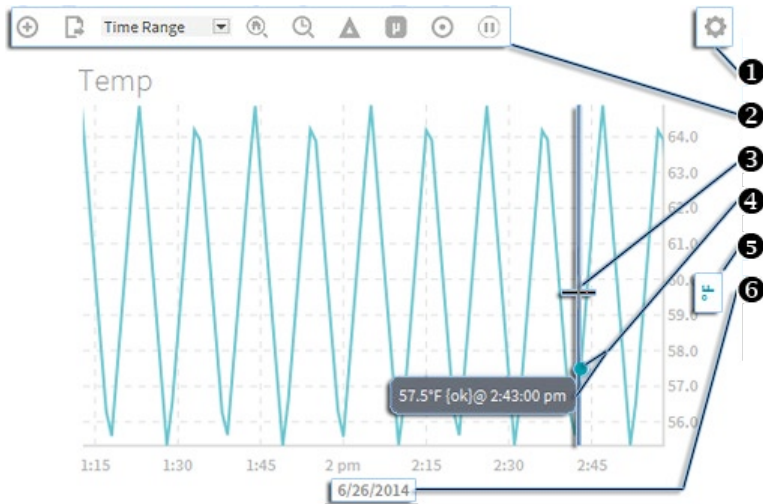
## Controls and options for charts

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.

### Chart view

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.

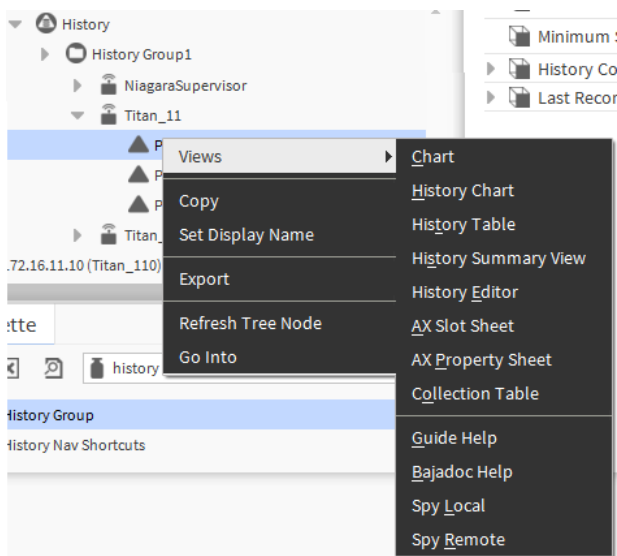
Figure 41: Chart view description



- ❶ Settings icon — click to access chart Settings window
- ❷ Command bar — click icons to launch chart commands
- ❸ Cursor position indicator
- ❹ Data Value popup — displays when cursor is on a point
- ❺ Y-Axis label — default orientation of Y-axis for primary data
- ❻ 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.

You can view histories in different ways in Workbench.

Figure 42: 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

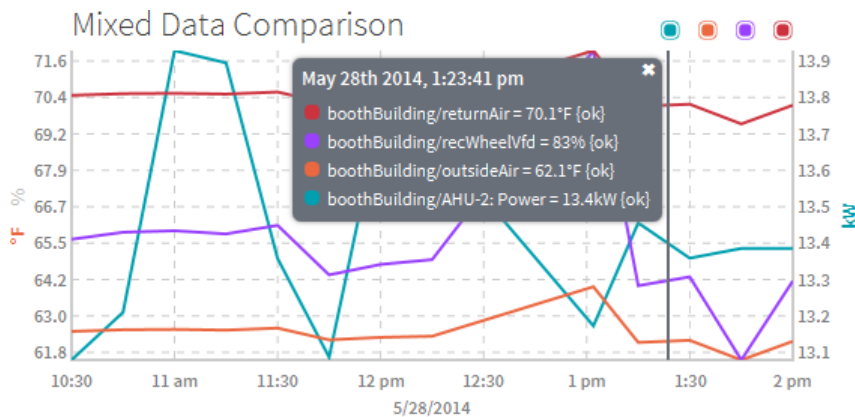
Although you can configure **Chart Type** via the Settings→Series window, the default chart type is determined by the type of data being presented. For example:

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 Enum 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 Enum 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.

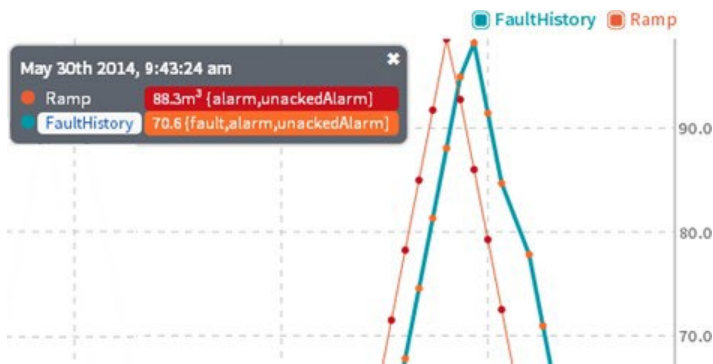
Figure 43: Multiple Y-axes accommodate data with different units of measure



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 44: 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.

Figure 45: Command Bar

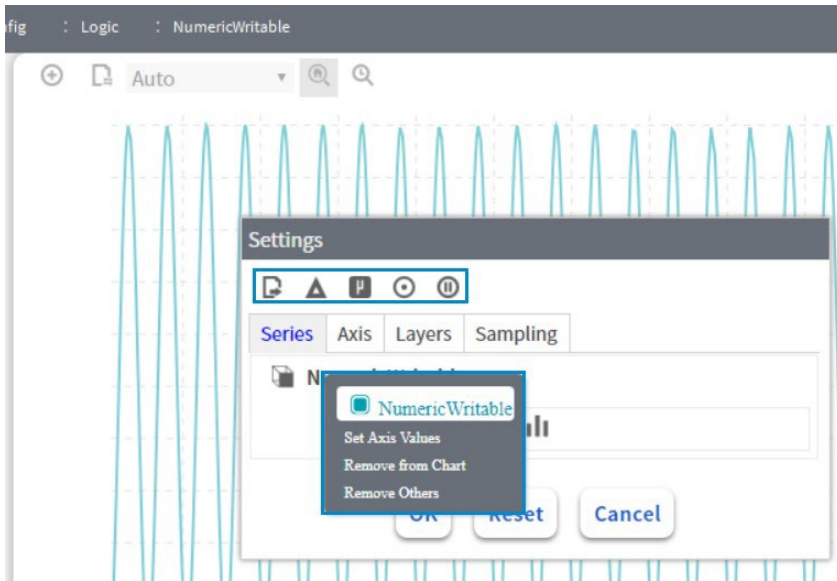


NOTE: Most options in the Command Bar provide fine tuning for viewing purposes only. Changes made with those options are of a temporary nature and are not included when the chart is saved or exported. For example, if you turn on time zoom and Delta using buttons in the command bar and then export the 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.

If you want 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 46: 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	Options	Description
 Add Series		Add components to the chart. Select one or more components via the File Chooser. Use <b>Ctrl + Click</b> to select multiple individual components or select a folder that contains multiple components.
 Save Chart		Available only when you open an existing .chart file and make changes. Save Chart — saves the file (chart or csv format) to the Station space (Files/charts/chartName.chart or Files/csv/chartName.csv).



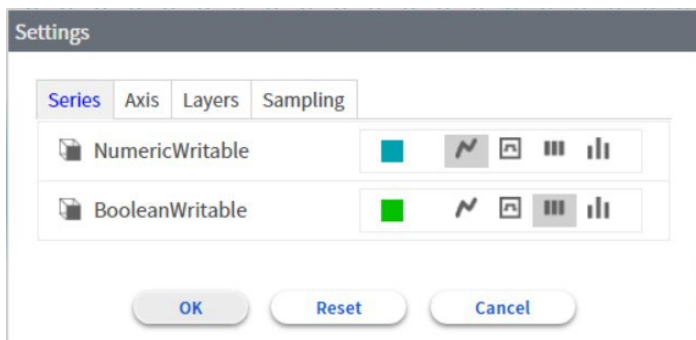
Command Bar	Options	Description
 <b>Export Current View or Object</b>	<ul style="list-style-type: none"> <li>Select Exporter</li> <li>Ord Type</li> <li>Base Ord</li> <li>Select Destination</li> <li>File Name</li> <li>View On Export</li> </ul>	<p>Available in a new chart and when you open an existing chart file.</p> <p><b>Select Exporter</b> — Choose the exported file type. Options are <b>Chart</b> (default), <b>CSV</b>, and in a browser connection <b>Print</b> is also available.</p> <p><b>Ord Type</b> — <b>Absolute</b> or <b>Relative</b> (default).</p> <p><b>NOTE:</b> In Niagara 4.6 and later, there is added support for “relativized” Ords to better accommodate Px page reuse.</p> <p><b>Base Ord</b> — Used only for chart exports with the <b>Relative Ord Type</b>. Allows you to specify a base ord to use to “relativize” all of the ords in the series for that chart.</p> <p><b>Destination</b> — (in Workbench) — <b>Station File Exports</b> file to station File space (Files/charts/chartName.chart) or (Files/csv/chartName.csv)</p>
<b>Time Range</b> 	<ul style="list-style-type: none"> <li>Auto (default)</li> <li>Time Range</li> <li>Today</li> <li>Last 24 Hours</li> <li>Yesterday</li> <li>Week To Date</li> <li>Last Week</li> <li>Last 7 Days</li> <li>Month To Date</li> <li>Last Month</li> <li>Year To Date</li> <li>Last Year</li> </ul>	<p>Specifies time range for data display. Selecting the <b>Time Range</b> option launches a window where you can enter custom <b>Start</b> and <b>End</b> times for the range. Leave the <b>Endtime</b> property blank for live data to continue plotting on the chart.</p>
 <b>Toggle HomeZoom</b>	<ul style="list-style-type: none"> <li>On (default)</li> <li>Off</li> </ul>	<p>Turns On/Off Home Zoom.</p> <p><b>On</b> — zooms to the X-axis of the primary data set.</p> <p><b>NOTE:</b> If the primary data set is numeric, it zooms on the Y-axis.</p> <p><b>Off</b> — reverts to Home Zoom.</p>
 <b>Toggle Time Zoom</b>	<ul style="list-style-type: none"> <li>On</li> <li>Off (default)</li> </ul>	<p>Turns On/Off Time Zoom.</p> <p><b>On</b> — zooms X-axis to the time period specified by the <b>Time Range</b> drop-down list.</p> <p><b>Off</b> — reverts to Home Zoom.</p>
 <b>Toggle DeltaCommand</b>	<ul style="list-style-type: none"> <li>OnOff (default)</li> </ul>	<p>Turns On/Off Delta.</p> <p><b>On</b> — plots the rate of change between points.</p>

Command Bar	Options	Description
		<i>Off</i> — resumes plotting data points.
 Toggle Sampling Command	<ul style="list-style-type: none"> <li>On</li> <li><i>Off</i> (default)</li> </ul>	Turns On/Off Sampling. <i>On</i> — sampling is enabled <i>Off</i> — turns off sampling and disables auto-sampling behavior.
 Toggle StatusColoring	<ul style="list-style-type: none"> <li>On</li> <li><i>Off</i> (default)</li> </ul>	Turns On/Off data Status Coloring. <i>On</i> — 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. <i>Off</i> — hides status coloring, data points/color bands.
 Toggle Pause	<ul style="list-style-type: none"> <li>On</li> <li><i>Off</i> (default)</li> </ul>	Turns On/Off pause in live data plotting. <i>On</i> — pauses live data plotting. No longer in live mode when paused <i>Off</i> — resumes live data plotting
 Stop	<ul style="list-style-type: none"> <li>On</li> <li><i>Off</i> (default)</li> </ul>	Visible only during data loading. Turns data chunking On/Off. <i>On</i> — stops the data chunking process, halts data coming from the server. While stopped, the button displays a red border. <i>Off</i> — reloads all of the data.

## Chart settings

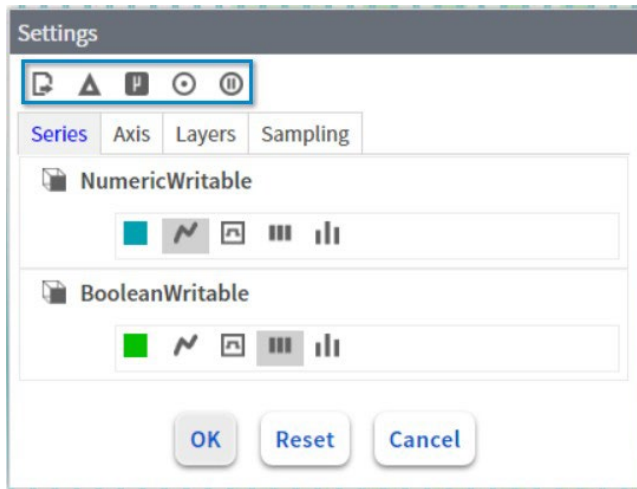
Options in the Chart view Settings window allow you to make data presentation changes that are of a persistent nature, meaning the changes are retained when the chart is exported or saved.

Figure 47: Settings Window in Workbench

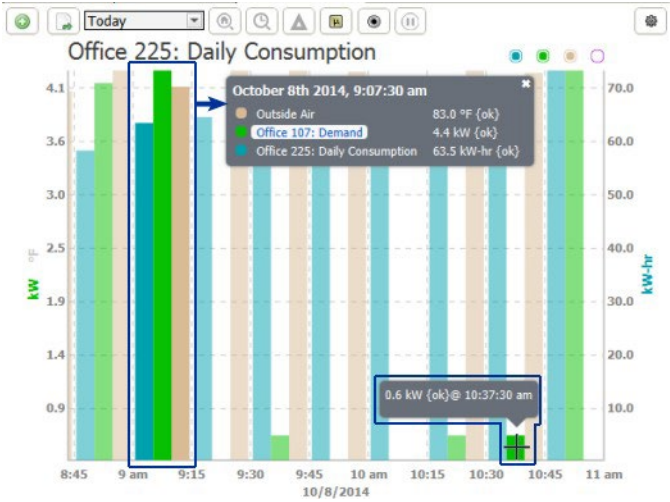


NOTE: In the Niagara 4.8 Workbench and later, if the chart width is less than 800 pixels, several of the chart commands icons are moved into the Settings window above the tabs. When the chart is resized wider than 800 pixels, those icons revert back to the Commands Bar in the chart.

Figure 48: Commands icons in the Settings window



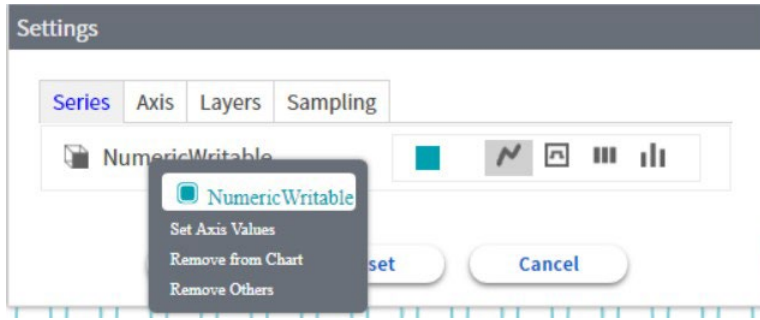
Series tab

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.
Chart type	<ul style="list-style-type: none"> <li>Line,</li> <li>Discrete line,</li> <li>Shade,</li> <li>Bar</li> </ul>	<p><b>Line</b> — plots a smooth line with interpolation. The default chart type for Numeric points and histories.</p> <p><b>Discrete line</b> — plots a “stepped” line without interpolation.</p> <p><b>Shade</b> — plots shaded areas, known as “swim lanes,” representing state change. The default chart type for Boolean and Enum points.</p> <p><b>Bar</b> — 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).</p> <p>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.</p> 

Right-click menu

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 49: Right-click menu options



### Axis tab

Settings	Options	Description
<b>Y-Axis Orientation</b>	<ul style="list-style-type: none"> <li>left</li> <li>right (default)</li> </ul>	Aligns Y-axis of primary data to either the left or right side of the chart.
<b>Data Zoom Scope</b>	<ul style="list-style-type: none"> <li>primary (default)</li> <li>all</li> </ul>	<p>Sets the Data Zoom Scope to primary or all.</p> <p>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.</p> <p>All — changes the X-axis to accommodate all available data, including live data as it is recorded.</p>
<b>Show Grid</b>	<ul style="list-style-type: none"> <li>true (default)</li> <li>false</li> </ul>	<p>Turns on/off the chart grid.</p> <p>true — the grid displays in chart behind data.</p> <p>false — the grid does not display.</p>
<b>Background Color</b>	<ul style="list-style-type: none"> <li>On</li> <li>Off (default)</li> </ul>	<p>Turns on/off the background area color for the current theme.</p> <p>On — the background area color displays in chart behind data.</p> <p>Off — the background area color does not display.</p>
<b>Chart Cursor</b>	<ul style="list-style-type: none"> <li>Crosshair (default)</li> <li>None</li> </ul>	<p>Sets the appearance of mouse pointer while positioned over a chart.</p> <p>Crosshair — the mouse pointer appears as a crosshair.</p> <p>None — turns off the mouse pointer visibility (while positioned over a chart), hiding it completely.</p>
<b>Facets Limit Mode</b>	<ul style="list-style-type: none"> <li>Off (default)</li> <li>Inclusive</li> <li>Locked</li> </ul>	<p>Configures whether the WebChart uses a point's facets for <b>Min</b> and <b>Max</b>.</p> <p>Off (default) — the WebChart ignores a point's facets for <b>Min</b> and <b>Max</b>.</p> <p>Inclusive — the WebChart includes a point's facets for <b>Min</b> and <b>Max</b>.</p> <p>Locked — forces the WebChart to use a point's facets for <b>Min</b> and <b>Max</b>.</p> <p>NOTE: In each of these settings <code>chartMin</code> and <code>chartMax</code> 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 <code>chartLimitMode</code> is supplied with the corresponding values of "Inclusive" or "Locked".</p>

Settings	Options	Description
		<p>NOTE: Previously, if you were not using a chart file to load a Web-Chart, there was no way to preset any options. In Niagara 4.6 and later, there is a <b>Default Options</b> WebProperty on a Px page which you can modify to preset WebChart default options. By default, modifications are saved to <code>file:^charts/defaultOptions.chart</code>.</p> <p>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.</p>
<b>Show Start Trend Gaps</b>	<ul style="list-style-type: none"> <li>• Yes (default)</li> <li>• No</li> </ul>	<p>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.</p> <p>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</p> <p>No — allows the dots to be connected, eliminating any such gaps.</p>
<b>Show Data Gaps</b>	<ul style="list-style-type: none"> <li>• Yes</li> <li>• No (default)</li> </ul>	<p>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).</p> <p>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.</p> <p>No — if a record has a hidden flag set or invalid values (+inf, -inf, NaN) the record's dots are connected to adjacent records.</p>

### Layers tab

Settings	Options	Description
<b>Data Popup</b>	<ul style="list-style-type: none"> <li>• On (default) Displays</li> <li>• Off Pauses</li> </ul>	<p>Enables/disables the Fixed Data popup.</p> <p>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.</p> <p>Off — suspends display of fixed data popup.</p>
<b>Data Mouseover</b>	<ul style="list-style-type: none"> <li>• On (default)</li> <li>• Off</li> </ul>	<p>Enables/disables the mouseover Data Value popup.</p> <p>On — mouse position on chart data displays the recorded component value, status, and the time for that mouse position.</p> <p>Off — suspends display of mouseover data value popup.</p>
<b>Status Coloring</b>	<ul style="list-style-type: none"> <li>• On</li> <li>• Off (default)</li> </ul>	<p>Turns On/Off data status coloring.</p> <p>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.</p> <p>Off — hides status color data points/color bands in the chart.</p>

### Sampling tab

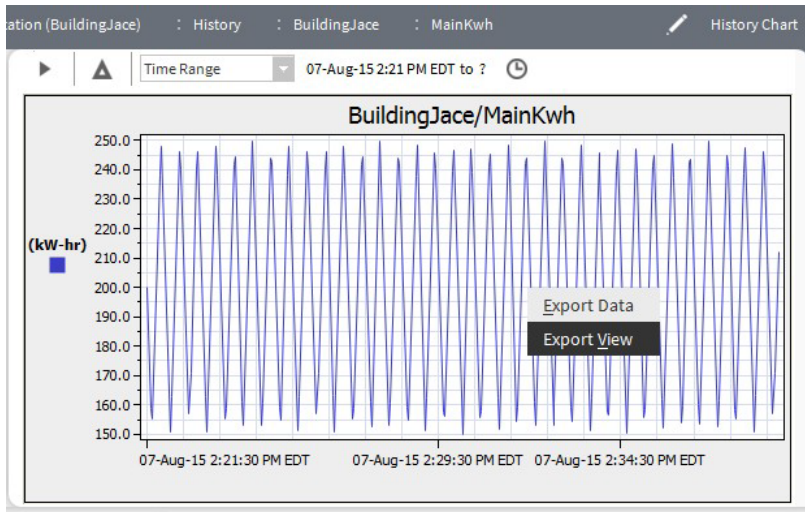
Settings	Options	Description
<b>Auto Sampling</b>	<ul style="list-style-type: none"> <li>• true (default)</li> <li>• false</li> </ul>	<p>Enables/disables automatic sampling optimizations.</p> <p>true — automatically begins sampling if the focused data set exceeds 2500.</p> <p>false — automatically stops sampling if the focused data set is below 2500.</p>
<b>Sampling Type</b>	<ul style="list-style-type: none"> <li>• Average (default)</li> </ul>	<p>Sets the Sampling type.</p>

Settings	Options	Description
	<ul style="list-style-type: none"> <li>• Min</li> <li>• Max</li> <li>• Sum</li> </ul>	<p>Average - samples average values for the selected rollup period.</p> <p>Min - samples minimum values for the selected rollup period.</p> <p>Max - samples maximum values for the selected rollup period.</p> <p>Sum - samples the total of the values in the selected rollup period.</p>
<b>Desired Period</b>	<ul style="list-style-type: none"> <li>• Best Fit (default)</li> <li>• 1 Minute</li> <li>• 15 Minutes</li> <li>• 30 Minutes</li> <li>• Hour</li> <li>• Day</li> <li>• Week</li> <li>• Month</li> <li>• Year</li> <li>• Custom</li> </ul>	<p>Configurable setting allows you to choose the desired sampling interval.</p> <p>By default, set to <i>Best Fit</i> 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.</p>
<b>Sample Size</b>	2500 (default)	<p>Specifies the number of points in the data set to sample. Range is 1–50000.</p> <p>NOTE: The default auto sampling size is configurable in the <code>system.properties</code> file.</p>
<b>Sampling</b>	<ul style="list-style-type: none"> <li>• true</li> <li>• false (default)</li> </ul>	<p>Enables/disables sampling for any size data set.</p> <p>true — turns on sampling</p> <p>false — turns off sampling</p> <p>NOTE: For performance reasons, sampling cannot be turned off once the focused data set exceeds 50,000. This threshold is configurable in the <code>system.properties</code> file.</p>
<b>Data Points</b>	Read only	Displays the maximum number of points in the data set that are available to sample.
<b>Sampling Period</b>	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.

## History Chart view

Charts created in the AX release, are available as secondary History Chart views on history records. The History Chart view uses one or more of the controls and options described in the following list.

Figure 50: History Chart view controls and options






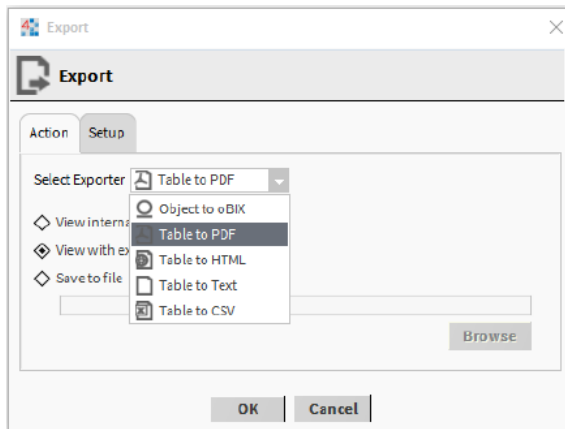
- Data parameters
  -  Delta reporting option  
This option is useful for history logging, when you want to display value changes (delta) in your report.
  -  Time range option list  
This list has a variety of predefined time range options, including an option that allows you to restrict your data presentation to a particular date and time range that you specify.
-  Live Updates  
Click the icon to start Live (on demand) History plotting. The icon changes to a “pause” icon while Live History plotting is active.
- Chart Title  
This area of the chart displays the name of the chart. This title is editable in the chart builder view.
- Y Axis  
Displays units for the vertical axis.
- X Axis  
Displays units for the horizontal axis.
- Charted Values  
The color of the line and type of line is editable in the Chart Builder view.
- Export menu  
Right-clicking on the History Chart invokes the Export window which contains two options:  
Export Data opens the Export window where you can choose to export the chart to oBIX, PDF, Text, HTML, or CSV (comma separated variable), as shown here.



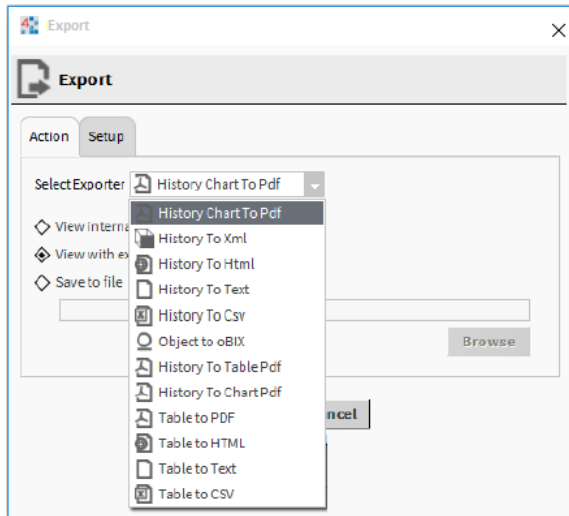
Figure 51: Export Data options



NOTE: This export function works only with charts that are using a single history. Multiple histories do not export in a usable format.

Export View opens the Export window where you can choose from a number of export options, as shown here.

Figure 52: Export View options



- Tool Tip

When you hover the mouse pointer over 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.

## Customizing the Workbench environment

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.

### Configuring the web browser whitelist

In Niagara 4.4 and later, using Workbench as a web browser is governed by the use of a whitelist which allows for added security and customizability. The whitelist specifies exactly which web addresses Workbench can navigate to. Consequently, you can no longer use Workbench to freely browse external web addresses unless you configure the whitelist to allow it.

The whitelist can be configured with allowed hostnames, domains, and subdomains, optionally filtering on protocol, port, and path as well. For more customized filtering, the regex: prefix may also be used. If desired, the web browser can be completely disabled.

- Step 1 Navigate to the `!defaults/system.properties` file.
- Step 2 Scroll to the `niagara.webbrowser.urlWhitelist` property, and enter a comma-separated list of URL patterns that you have decided are acceptable for Workbench to navigate to. Refer to the following list of configuration examples.
- Step 3 On completion, save your configuration changes and restart Workbench.

#### Web browser whitelist configuration examples

- To allow navigation only to web pages served by a particular hostname (such as your localhost), enter:  
`niagara.webbrowser.urlWhitelist=hostname`
- To allow navigation to a web server on any subdomain of a given domain (such as `domain.com`, `www.domain.com`, `subdomain.domain.com`, etc.), enter:  
`niagara.webbrowser.urlWhitelist=domain.com`
- To allow navigation to a specific subdomain (such as allowing `www.domain.com`, but not `domain.com`), enter:  
`niagara.webbrowser.urlWhitelist=www.domain.com`
- To allow navigation to the localhost but using only the specified protocol and port number. As an example using port 8088, enter:  
`niagara.webbrowser.urlWhitelist=https://localhost:8088`
- To allow navigation to any URL at `domain.com`, but filtering with a partial path such as `/public/`, enter:  
`niagara.webbrowser.urlWhitelist=domain.com/public/`
- Additional customizations are possible using regex syntax. The regex will match on any substring of the URL. For example, `regex:a` would match any URL that contains the letter "a". Similarly, to match on any file ending in `.htm` at `domain.com`, but no other files, enter:  
`niagara.webbrowser.urlWhitelist=regex:domain.com/.*/\w+\\.htm$`
- To specify multiple URL patterns, enter them in a comma-separated list such as:  
`niagara.webbrowser.urlWhitelist=localhost,niagara-central.com,bacnet.org`
- 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:  
`niagara.webbrowser.urlWhitelist=regex:`

NOTE: Be aware of any security implications or organizational policies before allowing Workbench to browse the web unrestricted.

- Prevent browsing to external URLs by leaving the whitelist empty. To configure the empty whitelist, enter: `niagara.webbrowser.urlWhitelist=`

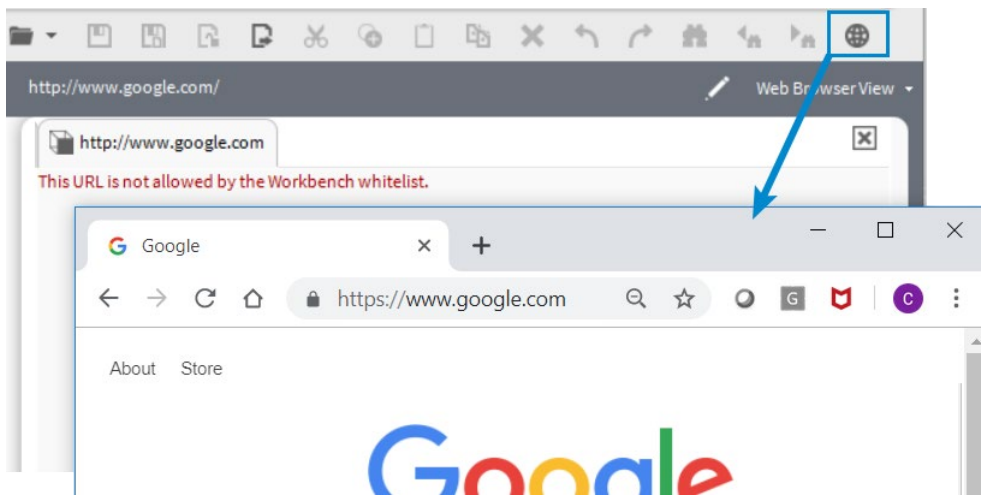
NOTE: The default whitelist includes “tridium.com” and “niagara-community.com” which are necessary to enable Cloud services such as Device Registration. If the whitelist is configured without these domains, Cloud services will not work in Workbench. You will need to use your desktop browser instead. If the system property is missing, then “tridium.com,niagara-community.com” is the default whitelist.

- To disable the web browser altogether set the `niagara.webbrowser.disabled` property. To set the property enter: `niagara.webbrowser.disabled=true`

NOTE: 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.

- For convenience, you can open all URLs in your PC’s default desktop web browser. For example, if you navigate to a URL using Workbench and find that the site is not allowed by the whitelist, click the Open In Desktop Browser command in the toolbar to access the site in your desktop web browser.

Figure 53: Toolbar command to Open In Desktop Browser



## Creating Additional Windows

Workbench allows you to create multiple fully functioning windows.

To create a duplicate of your current window, choose File:→New Window from the menu bar.

After you create multiple windows, you can then customize each individual window, as needed, to access different information, allowing you to see multiple concurrent views.

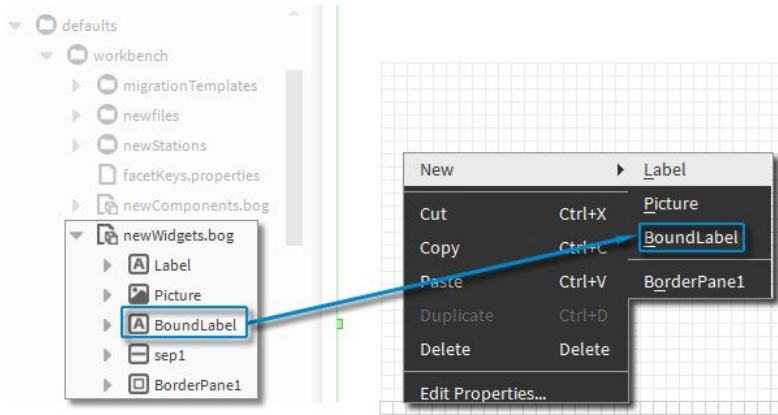
You can also copy and paste items from one window into another.

## Editing options in the “new” popup menu

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.

Adding a widget to the `newWidgets.bog` file adds the item to the PxEditor New popup submenu.

Figure 54: newWidgets.bog file



**Step 1** In the Nav tree under My File System, expand `SysHome/defaults/workbench/newWidgets.bog`.

**Step 2** Open the `kitPx` palette.

**Step 3** Drag and drop the `BoundLabel1` widget onto the `newWidgets.bog` file.

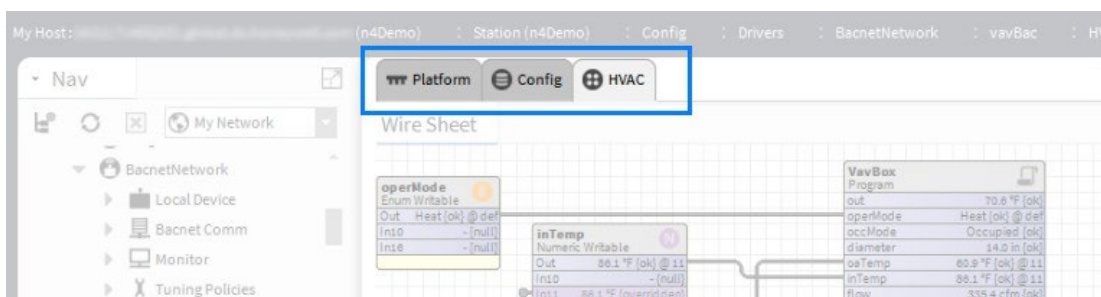
The Bound Label option is immediately available in the New popup submenu.

## Creating multiple tabs in the view pane

Workbench allows you to create multiple tabbed views within a single view pane.

As needed, you can use each tab to display a different view, component, or even host, all within the same window, as shown.

Figure 55: Multiple tabs in view



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.

Only one tab may be active at a time. In addition to simply clicking on a tab to make it active, you can select `File → Next Tab` from the menu bar to move to the next tab, moving left to right. Also, you can move to the last tab by choosing (from menu bar) `File → Last Tab`.

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

## 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.

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

- Use the keystroke combination: Click **Ctrl + 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.

## Closing a tab

Closing a tab removes the tab from the view pane.

**Step 1** To close the active tab in the view pane, use any of the following methods:

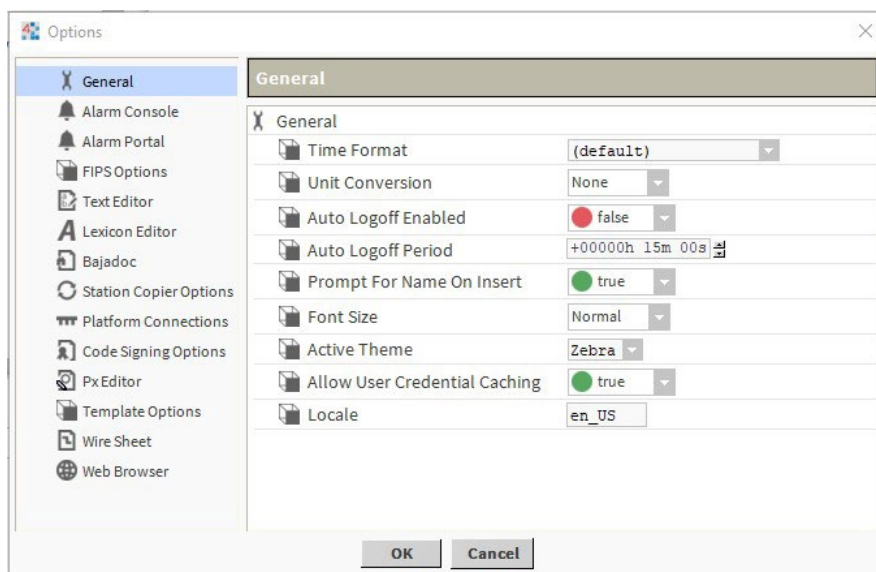
- Click **Close** icon located in the upper-right corner of the view, just below the **View** drop-down list.
- Right-click a tab and choose **Close Tab** to close that tab
- Right-click a tab and choose **Close Other Tabs** to close all tabs except that tab.
- From the menu bar, choose **File: → Close**

The currently active tab no longer appears.

## Types of Workbench 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.

Figure 56: Options window



The following options are available in the Options window:

- General
- Alarm Console
- Alarm Portal
- FIPS Options (Niagara 4.6 and later)
- 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 options include settings for a variety of Workbench display and behavior options.

Figure 57: General Workbench options

General	
Time Format	(default) ▾
Unit Conversion	None ▾
Auto Logoff Enabled	<span style="color: red;">●</span> false ▾
Auto Logoff Period	+000000h 15m 00s ⚙
Prompt For Name On Insert	<span style="color: green;">●</span> true ▾
Font Size	Normal ▾
Active Theme	Zebra ▾
Allow User Credential Caching	<span style="color: green;">●</span> 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 section “baja-UserService” as well as in the *Niagara 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 are 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.

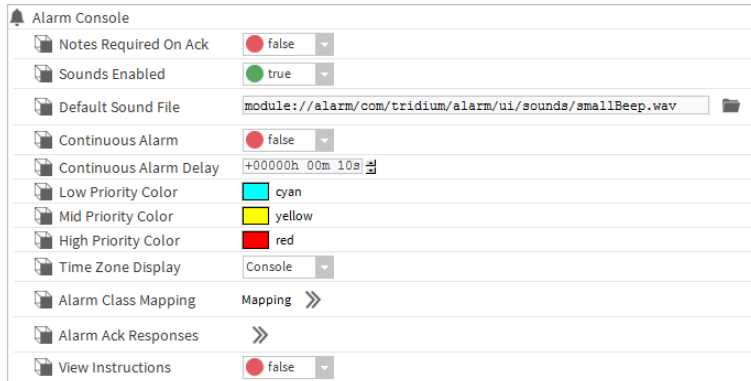
- Locale

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.

## Alarm Console

Alarm Console options allow you to customize both the appearance and behavior of the alarm console.

Figure 58: Alarm Console



Alarm Console options include the following:

Property	Value	Description
Notes Required onAck	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 <b>Continuous Alarm Delay</b> property. You can also set this value under the Alarms menu in the Workbench main menu when the Alarm Console view is active.



Property	Value	Description
Continuous Alarm Delay	hours, minutes, and seconds	When <b>Continuous Alarm</b> 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.
Alarm priority coloring	color	<p>The following properties work together in combination to produce a range of colors that are assigned to the possible range of alarm priorities (1 - 255).</p> <ul style="list-style-type: none"> <li>• <b>Low Priority Color</b> Choose a color to designate a low priority alarm.</li> <li>• <b>Mid Priority Color</b> Choose a color to designate a mid priority alarm.</li> <li>• <b>High Priority Color</b> Choose a color to designate a high priority alarm.</li> </ul> <p>The highest priority (priority 1) is assigned the color of the High Priority Color setting. The Mid-Priority and Low Priority colors are assigned likewise, to Mid and Low Priority alarms. Alarm priorities that fall between these priority levels are assigned colors on a color-scale along a path defined by the three assigned colors.</p>
Time Zone Display	drop-down list (defaults to Console)	Allows you to choose to display 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 display the Notes window when acknowledging an alarm. When the Notes Required on Ack is set to <code>true</code> , 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 <b>Notes Required On Ack</b> is set to <code>false</code> , these Alarm Ack responses are not visible.
View Instructions	<code>true</code> or <code>false</code> (defaults to <code>false</code> )	This property causes the alarm Instructions pane to display across the bottom of the Alarm Console, if this option is set to <code>true</code> . Instructions display in the pane for any single selected alarm that has associated instructions.

## Alarm Portal

Alarm Portal options allow you to customize both the appearance and behavior of the Portal Alarm Console.

Figure 59: Alarm Portal

Alarm Portal	
🔔 Alarm Portal	
📁 Tray Icon Enabled	<input checked="" type="radio"/> true
📁 Alarm Popup Enabled	<input checked="" type="radio"/> true
📁 Alarm Popup Always On Top	<input checked="" type="radio"/> true
📁 Alarm Popup Uncloseable	<input checked="" type="radio"/> true
📁 Kiosk Mode	<input type="radio"/> false
📁 Reconnect Interval	+000000h 02m 00s
📁 Default Time Range	Today

Alarm Portal options include the following:

Property	Value	Description
Tray Icon Enabled	true or false (defaults to true)	Displays an alarm icon in the system tray when the alarm portal is active, if set to true.
Alarm PopupEnabled	true or false (defaults to true)	Displays an alarm popup window when the alarm portal is active, if set to true.
Alarm Popup Always On Top	true or false (defaults to true)	Causes the alarm popup window to stay on top of other windows when the alarm portal is active, if set to true.
Alarm PopupUncloseable	true or false (defaults to true)	Makes the alarm popup window uncloseable when the alarm portal is active, if set to true.
Kiosk Mode	true or false (defaults to false)	The alarm portal opens in Kiosk Mode the next time it is started, if set to true. For related information, refer to the <i>NiagaraAX Graphics Guide</i> .
Reconnect Interval	hours, minutes, seconds	The alarm portal checks for disconnected alarm consoles. If a console is disconnected, a reconnect is attempted within the Reconnect Interval time.
Default TimeRange	time range	Allows you to choose the default time range that displays in the Portal Alarm Console pane of the Alarm Console view(Console).

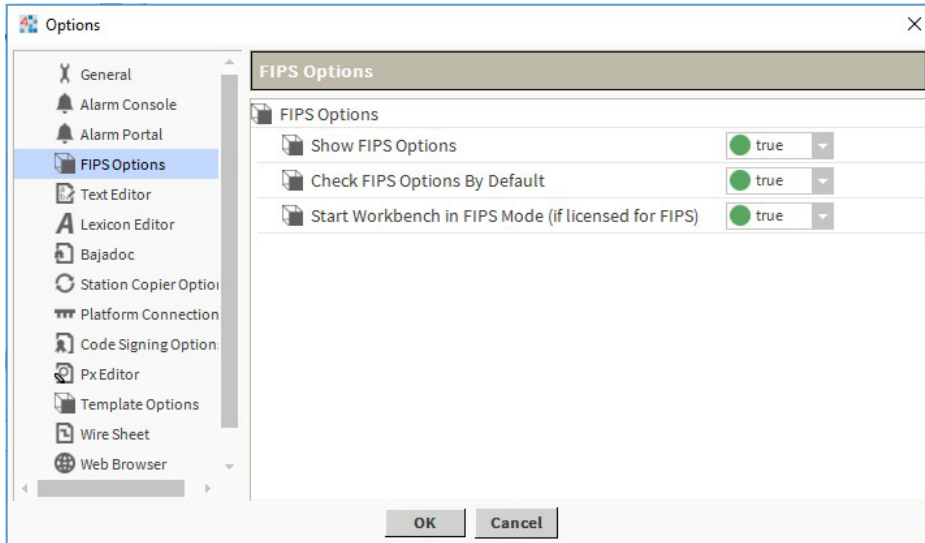
## FIPS Options

In Niagara 4.6 and later, 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 60: Workbench FIPS Options

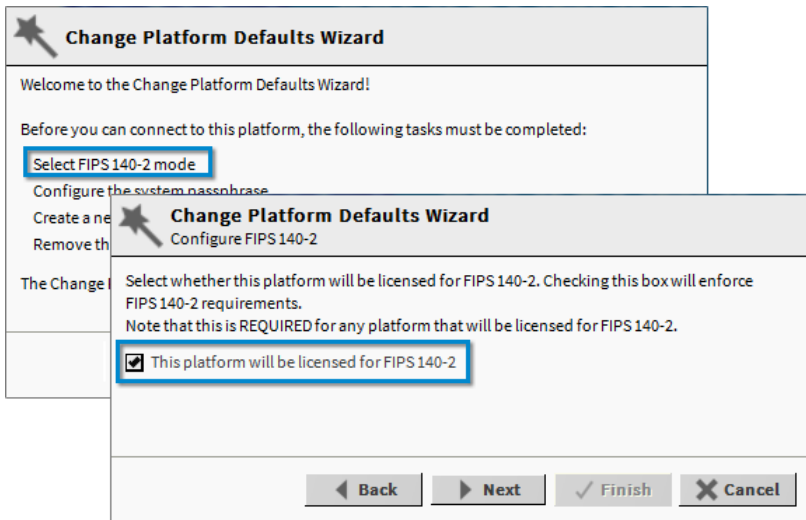


Setting Show FIPS Options to true causes certain FIPS options to be visible during the following tasks:

- Changing the default platform credentials via the Change Platform Defaults Wizard:

If the Workbench option to "Show FIPS Options" is set to "true" there is an added "Select FIPS 140-2 mode" step in the Change Platform Defaults Wizard, 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. Note that if not checked, the platform does not consider a password FIPS-compliant, even if it technically meets the requirements. Also, if both of the Workbench FIPS Options are set to "true", then by default this checkbox is visible and selected. In that situation, the wizard enforces FIPS password strength requirements by default.

Figure 61: Change Platform Defaults Wizard step to select FIPS mode



- Changing the system passphrase via the System Passphrase command in Platform Administration.

Figure 62: FIPS Option in Set System Passphrase window

Set System Passphrase

Set the passphrase used to encrypt sensitive information on platform's filesystem:

**Current Passphrase** .....

**New Passphrase** .....

**Confirm New Passphrase** .....

This host will be licensed for FIPS 140-2

OK Cancel

- Changing the platform user passwords via the User Accounts command in Platform Administration:

Figure 63: FIPS Option in Manage platform daemon users

Manage platform daemon users

**Users**

Name	Comment
platAdmin	new platform user acct

This host will be licensed for FIPS 140-2

New User Delete User Change Password

OK

- Setting the system passphrase and platform user passwords during Commissioning.

Figure 64: FIPS Option in Commissioning

**Commissioning**

This wizard combines steps for configuring a host to run stations. Please check below for each type of configuration change you wish to make:

This platform will be licensed for FIPS 140-2

Request or install software licenses

Set enabled runtime profiles

Install a station from the local computer

Install lexicons to support additional languages

Install/upgrade modules

Install/upgrade core software from distribution files

Sync with my local system date and time

Configure TCP/IP network settings

Configure system passphrase

Configure additional platform daemon users

Clear All Check All

Back Next Finish Cancel

**NOTE:** To install a FIPS license to a particular host, the Workbench FIPS Options described above must be set to true.

## FIPS Compliant Passwords in Workbench

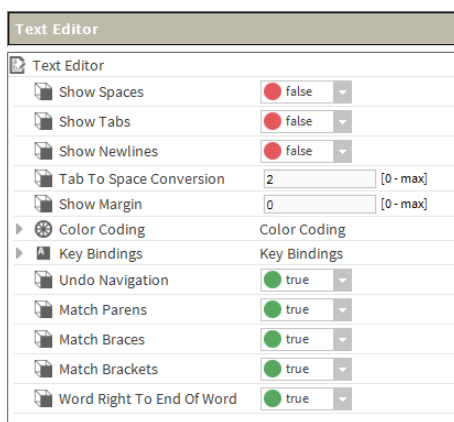
Workbench running in FIPS mode will also enforce strong passwords for operations such as exporting certificates, setting passwords on certificates, and logging in to stations.

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.

## 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.

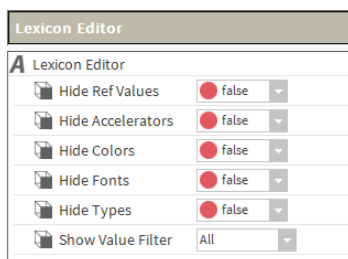
Figure 65: Text Editor



## 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.

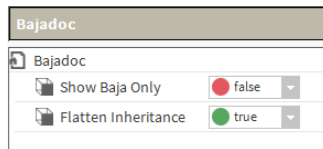
Figure 66: Lexicon Editor



## Bajadoc

Baja reference documentation includes both Java API details as well as Baja slot documentation.

Figure 67: Bajadoc

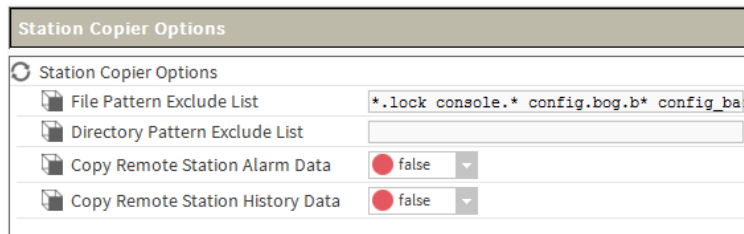


Property	Value	Description
Show Baja Only	true or false (defaults to false)	When set to <code>true</code> displays only the reference documentation for slots (Properties, Actions, and Topics). When set to <code>false</code> , documentation on the Java constructors, methods and fields is also displayed.
Flatten Inheritance	true or false (defaults to false)	Allows you to flatten the inheritance hierarchy into a single set of documentation. When set to <code>false</code> only the Java members and Baja slots declared in the specified class are displayed.  When set to <code>true</code> all Java members and Baja slots inherited from super classes are also shown.

## 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).

Figure 68: Station Copier Options



- 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)

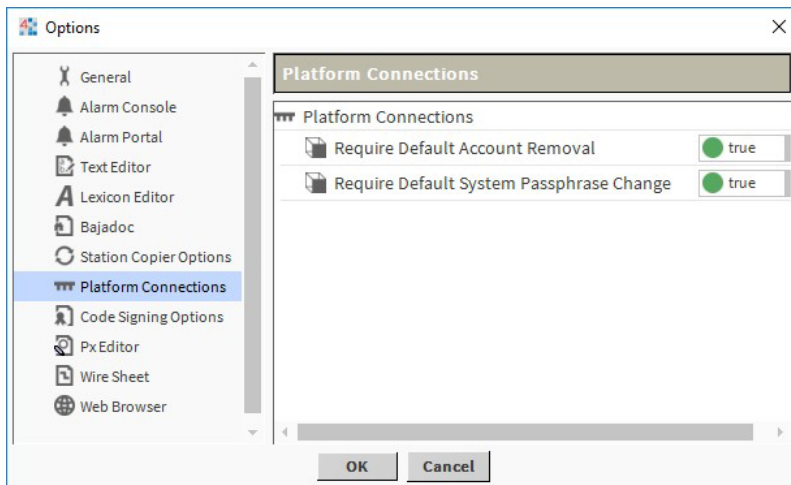
These properties allow you to copy (`true`) or not copy (`false`) certain station data (from the Remote Station to the Workbench).

**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*.

## Platform Connections

In Niagara 4.4 and later, 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.

Figure 69: 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.

## Option properties

Name	Value	Description
Require Default Account Removal	true (default) or false	<p>true configures the system to require you to delete the default platform account after you connect to the platform for the first time.</p> <p>false allows the default admin account to remain, pending its deletion by another process.</p> <p>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.</p>
Require Default System PassphraseChange	true (default) or false	<p>The system passphrase protects access to the platform.</p> <p>true configures the system to require you to create a new, strong passphrase after you connect to the platform for the first time.</p> <p>false preserves the default passphrase pending its change by another process.</p> <p>A strong passphrase protects a platform from unauthorized access.</p>



## 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).

Figure 70: 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.

### Signing program objects

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.

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 to 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.

### 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.

Prerequisites: You are working in Workbench running on a Supervisor or engineering PC.

Step 1 Click Tools→Certificate Management.

The **Certificate Management** view opens.

Step 2 Click the New button at the bottom of the view. The **Generate Self Signed Certificate** window opens.

Figure 71: Generate Self Signed Certificate window

Step 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-digit code (must be US or us, not USA).

Choose **Code Signing** for **Certificate Usage**.

The OK button activates when all required information is provided.

Step 4 To create the certificate, click OK. The Private Key Password window opens.


Figure 72: Private Key Password window

Step 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.

Step 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.

### 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.

Prerequisites: You are using Workbench running on a PC.

Step 1 If necessary, navigate to the **Certificate Management** view and select the code-signing certificate.

The view opens to the User Key Store.

Step 2 Select the code-signing certificate and click the Cert Request button at the bottom of the view.

Step 3 Confirm that the certificate properties are correct and click OK.

The Certificate Manager prompts you for the private key password.

Step 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`.

Step 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`.

Step 6 To confirm completion, click OK.

Step 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).

## 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 Niagara 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.

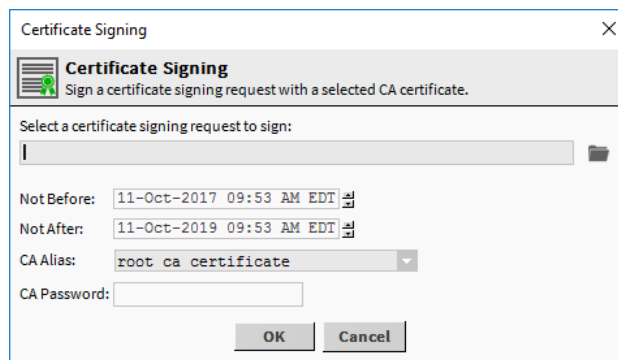
### Prerequisites:

- 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.

**Step 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.

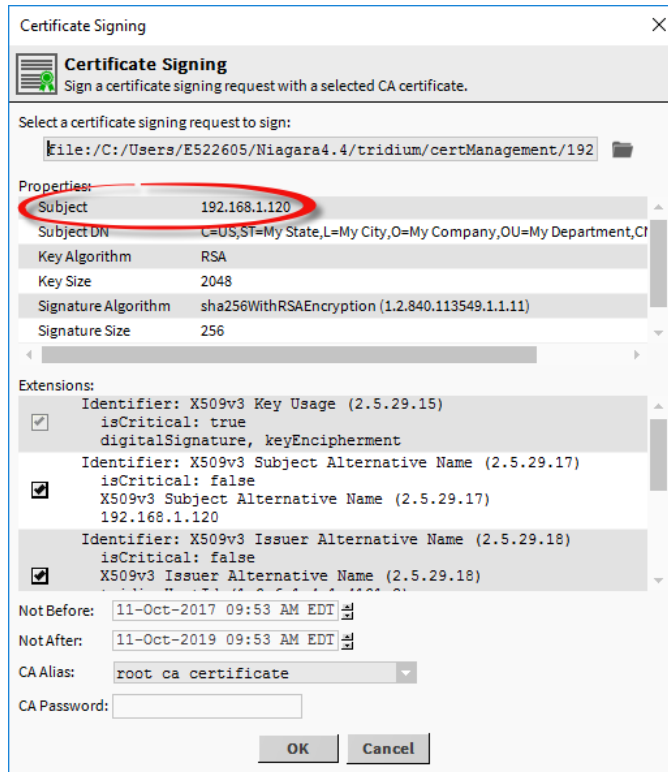
Figure 73: Certificate signing window



**Step 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.

Figure 74: Certificate signing window



Step 3 Confirm that this is the correct CSR by checking the Subject.

Step 4 Select the date on which the certificate becomes effective (**Not Before**) and the date after which it expires (**Not After**).

Step 5 For **CA Alias**, use the drop-down list to select the certificate (root or intermediate) whose private key will sign this certificate.

Step 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.

Step 7 To complete the signing, click Save.

Step 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.

Repeat this procedure for each CSR.

### 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.

Prerequisites: You have the signed `.pem` files. The focus is on the User Key Store in the appropriate stores location (Workbench or platform/station).

Step 1 Click **Import**.

Step 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.

Step 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.

Step 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.

Step 5 Using the operating system, delete the `.pem` file(s) from the secure Workbench computer.

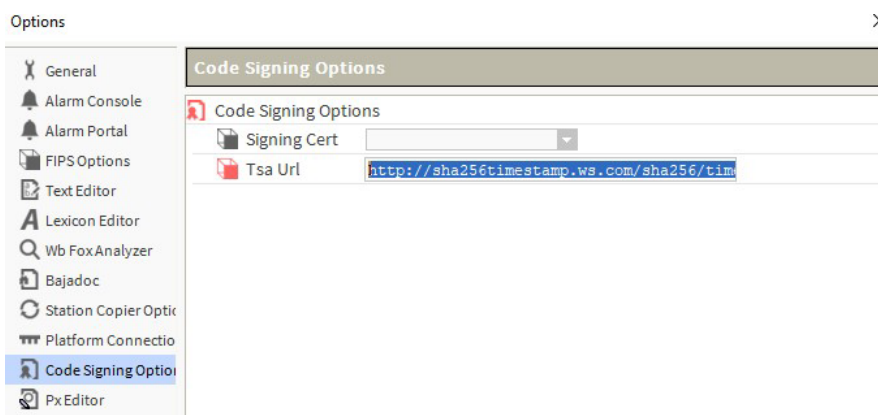
## Configuring Workbench to sign program objects

To begin signing program objects, the code-signing certificate you created must be selected as the signing tool.

Prerequisites: The code-signing certificate exists.

Step 1 In Workbench, click **Tools**→**Options**, and click **Code Signing Options**. The **Code Signing Options** property sheet opens.

Figure 75: CodeSigningOptionsproperty sheet



Step 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.

Step 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.

Step 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.

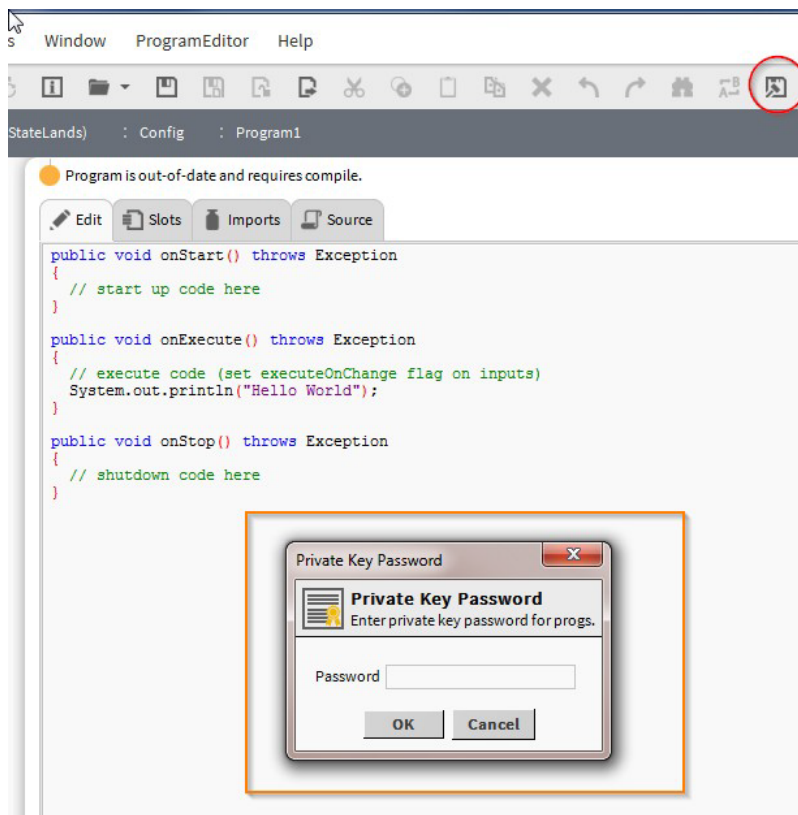
## Supplying the private key password

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.

### Step 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.

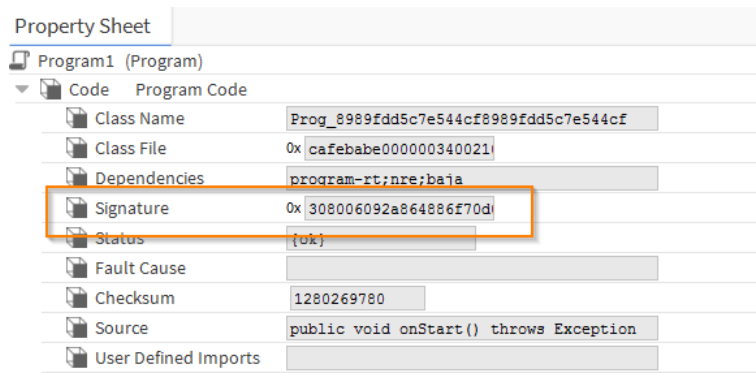
Figure 76: Private key password



Step 2 Enter the password you created when generating the certificate.

Step 3 Verify that the new signature property on Code is displayed.

Figure 77: Verify new signature property



## Approving an exception

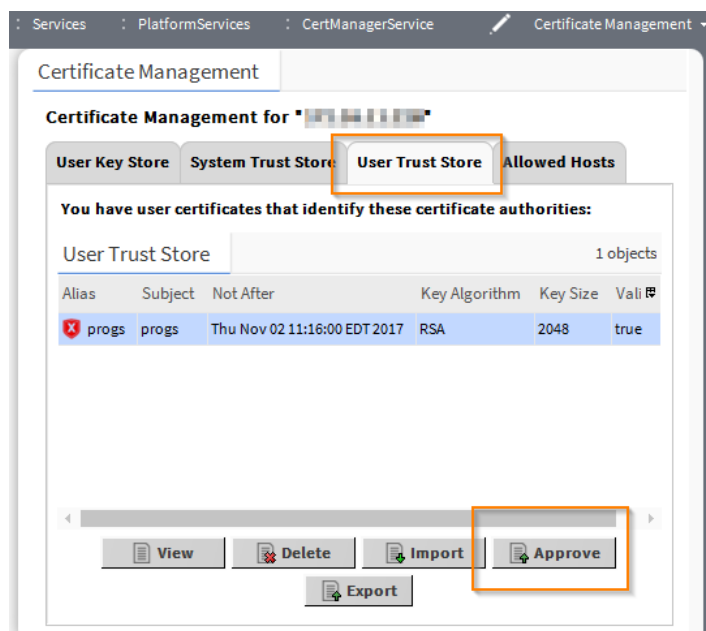
The station verifies the signature on each program object to ensure that the certificate chain is trusted. If the root CA certificate used to sign the code-signing certificate is in the station's **User Trust Store**, the code runs without further intervention. If the code signing-certificate is self-signed, or the root CA certificate is missing, the station reports an error, after which you may approve an exception.

Prerequisites: Your program object has been signed by a self-signed code-signing certificate, or the code-signing certificate is not trusted.

Step 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**.

Figure 78: Certificate Management



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.

Step 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**.



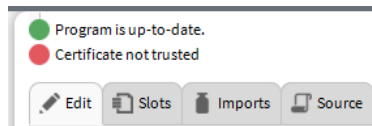
## Approving an exception in an Editor

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.

Prerequisites: 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**.

Step 1 Open a program in the Program Editor.

Figure 79: 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.

Step 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**.

## Installing a certificate

If your System Trust Store already contains the root CA certificate of 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 TrustStore of all platform/stations that serve as system clients. To do this, use an Install Certificate provisioning job. This can be useful before running a signed provisioning robot on several stations.

Prerequisites:

- The BatchJobService is available under the Services node and ProvisioningNwExt is available under your NiagaraNetwork in the Nav tree.
- The root CA or signed intermediate certificate is available on a thumb drive.
- The provisioningNiagara palette is open.

Step 1 Open the platform/station stores on a Supervisor (or engineering) computer and click the User Trust Store tab.

NOTE: Make sure you are in the platform/station stores. You cannot complete this procedure if you import the certificate into the Workbench User Trust Store of your Supervisor or engineering computer.

Step 2 Click the Import button, navigate to the location on the thumb drive that contains the root CA certificate and click Open.

Step 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.

Step 4 Navigate to the location in the station where you manage provisioning jobs.

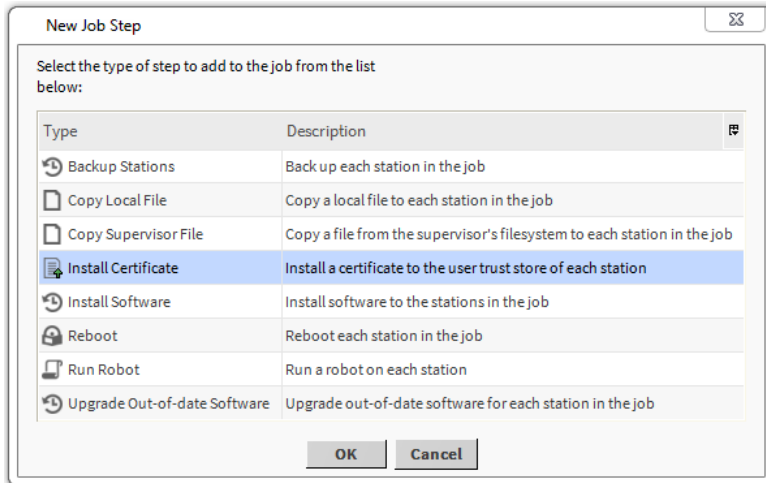
Step 5 Drag a NiagaraNetworkJobPrototype component to this location and name the component something like, “Root CA certificate provisioning.”

Step 6 Double-click the new component.

The Niagara Network Prototype View opens.

Step 7 In the Steps to run for each station pane, click the plus icon. The New Job Step window opens.

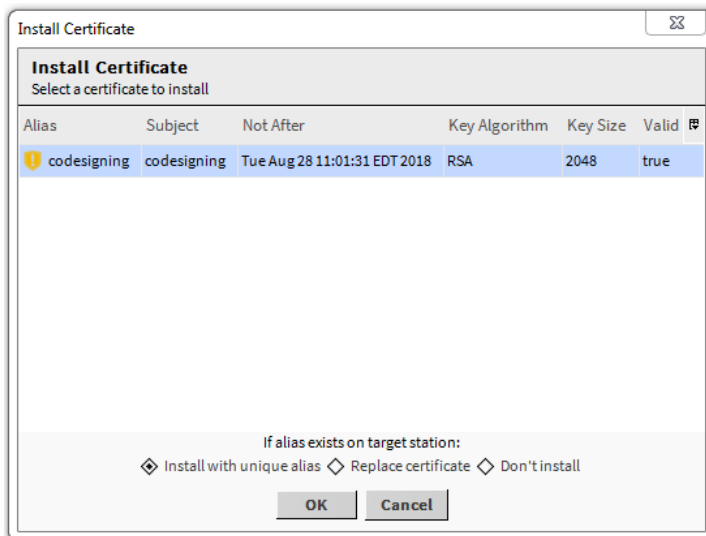
Figure 80: New Job Step window



Step 8 Select **Install Certificate** and click OK.

The Install Certificate window opens.

Figure 81: Install Certificate window



Step 9 To complete the installation, select the root CA certificate and click OK.

Step 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 TrustStores of the other clients.

## Recompiling and signing program objects

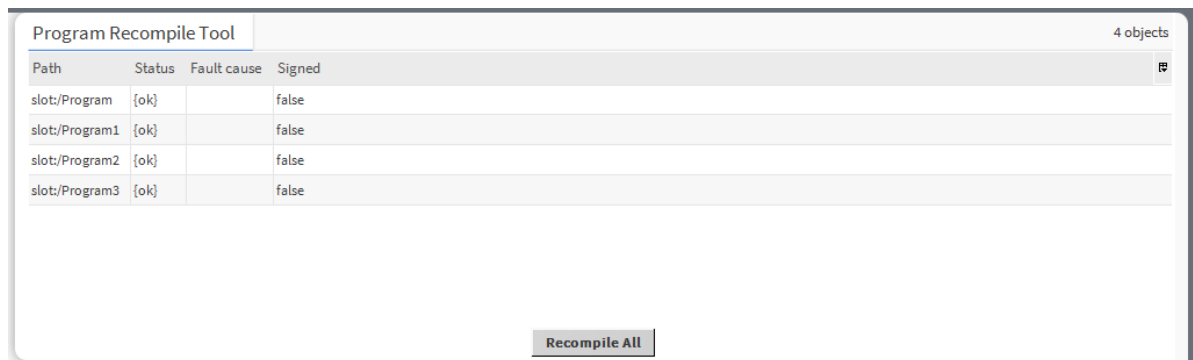
To sign all existing unsigned program objects, use the Program Recompile Tool view on the Program Service. This recompiles all program objects on the station, and, if code signing is configured, signs them using the code-signing certificate.

Step 1 Access the Program Service.

Step 2 Click Program Recompile.

The Program Recompile Tool view opens.

Figure 82: Program Recompile Tool



This view lists every program object in the station.

Step 3 Click Recompile All.

The system recompiles and signs every program object using the currently selected 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.

### 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.
```

## 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.

When the migrator encounters the first program object that needs signing, it prompts you for your code-signing certificate password.

## Code-signing troubleshooting

This topic summarizes common errors when configuring code signing.

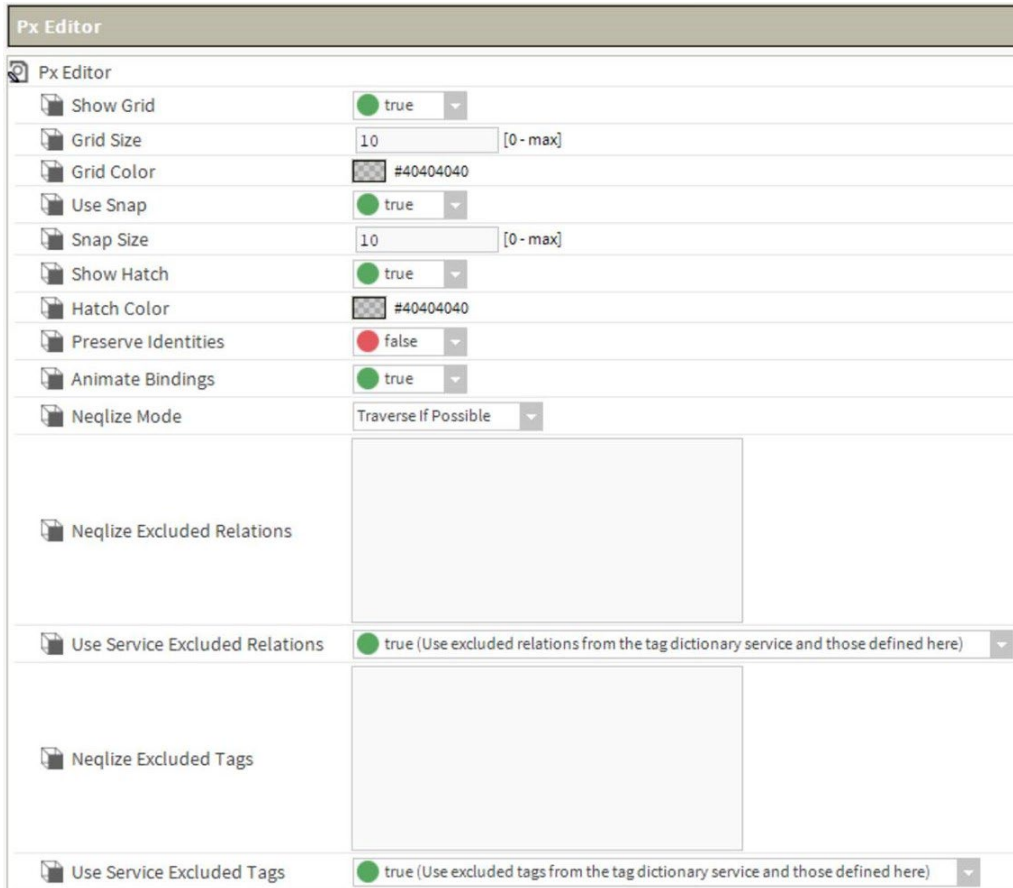
### Code signing is not working.

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.

## Px Editor

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 83: Px Editor options



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.

Property	Value	Description
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 (defaultsto 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 colorfield to display the Color Choose window. Use the Color Chooser to set the color that you want to assign to the Px editor hatching.
Preserve Identities	true or false (defaults to false)	If set to true, this property allows you to explicitly turn on support for encoding all names and handles on a Px page.
Animate Bindings	true or false (defaults to true)	This option, true by default, allows the Px Editor to display live binding data for widgets in Px Edit mode. If you set this option to false, then no data animation occurs in the Edit mode, although animation does occur, as expected, in View mode.
Neqlize Mode	Traverse if possible (default), Traverse only, Select only	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: at tempts to find a traverse query only. Select only: attempts to find a select query only. For details, see About tag-based Px bindings.
Neqlize Excluded Relations	(empty by default)	User-entered values for relation pattern filters used to excluderelations when converting slot path Ords to traverse NEQL query Oords.
Use Service Excluded Relations	true (default), false	When "true", the user values for relation pattern filters will be appended to the TagDictionaryService values. Otherwise, the user values will be used exclusively and the TagDictionaryService values will be ignored.
Neqlize Excluded Tags	(empty by default)	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	When "true", the user values for tag pattern filters will be appended to the TagDictionaryService values. Otherwise, the user values will be used exclusively and the TagDictionaryService values will be ignored.

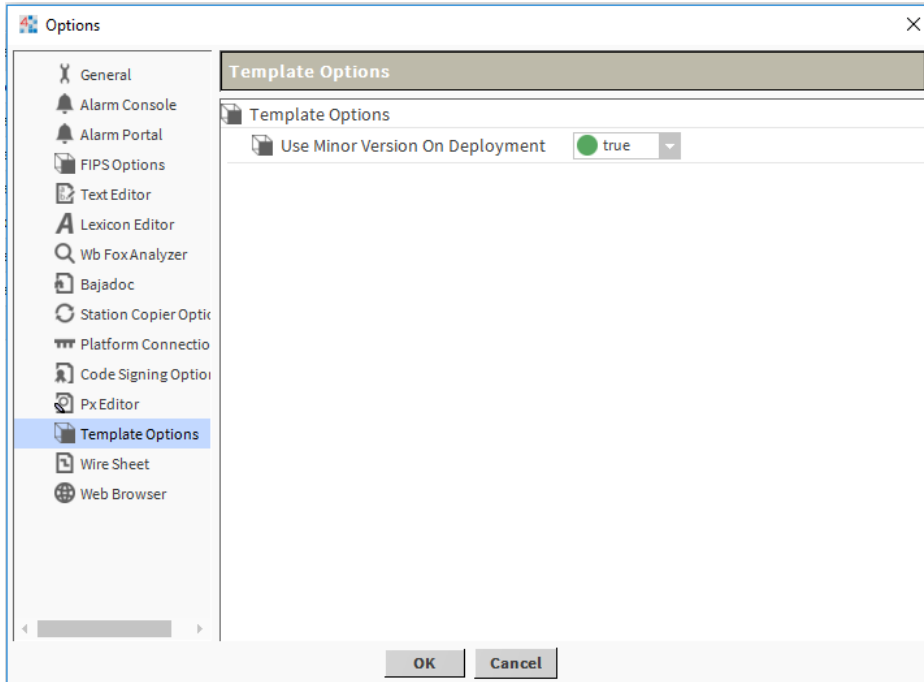
## Template Options

This option is used 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 4.8 have dependencies on minor version of Niagara 4.8 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.

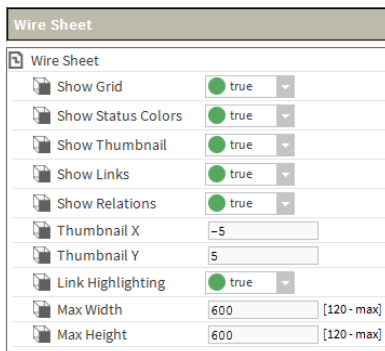
Figure 84: Workbench Template Options



## Wire Sheet

Wire sheet options allow you to customize the appearance of the Wire Sheet view.

Figure 85: Wire Sheet



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.
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 (defaultsto -5)	A field is provided for setting the X axis for the default position of the thumbnail view.
Thumbnail Y	number (defaults to 5)	A field is provided for setting 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.

## Web Browser

This option allows you to customize the web browser by enabling or disabling the Web Development Tools.

## Loading splash screen

From Niagara 4.4 onwards, a loading splash screen appears while Workbench is loading. The loading splash screen appears as shown in the image below.

Figure 86: Replacing Loading Splash Screen



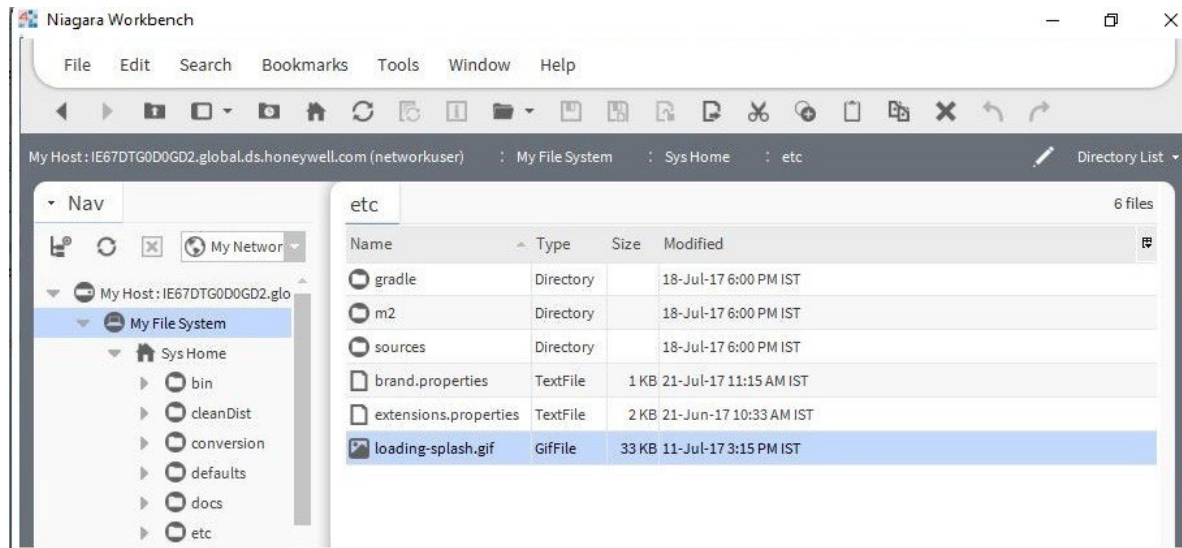
Starting in Niagara 4.4, you can replace the default splash screen image with alternate image.

**Step 1** Go to My File System→Sys Home→etc in the Nav tree.



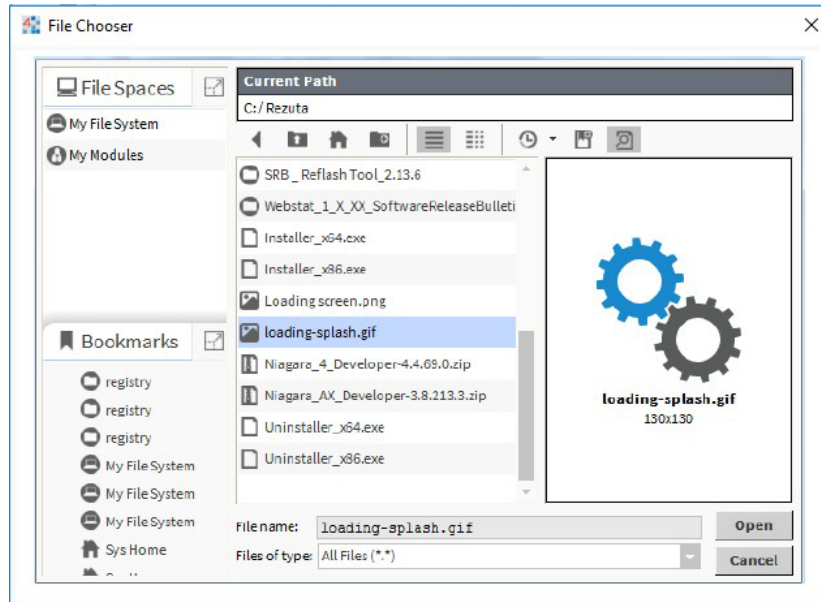
## Getting Started with Niagara

Figure 87: Workbench



- Step 2 Delete the existing loading-splash.gif file.
- Step 3 Right click on the screen and select 'Copy From' option from the list.
- Step 4 A File Chooser window appears. Choose the correct location from local drive where the desired gif file is located.
- Step 5 Select a new .gif file and click Open button.

Figure 88: File Chooser window



NOTE: If you wish to add new .gif file, it must have the name "loading-splash.gif".

- Step 6 Delete the .gif file, if you do not want a loading splash screen to appear.

## About Workbench themes

Workbench themes options allow you to customize the appearance of the Workbench display. Generally speaking, a theme is a predefined collection of design elements that determine the way an application appears to the user. Within the Niagara Framework, a theme is a module that controls the appearance of Workbench on the local computer by defining fonts and colors, as well as the icons, used in the display. Choosing a different theme affects only the Workbench appearance, there is no other impact.

You can customize your Workbench display by selecting the theme that you prefer.

### Types of themes

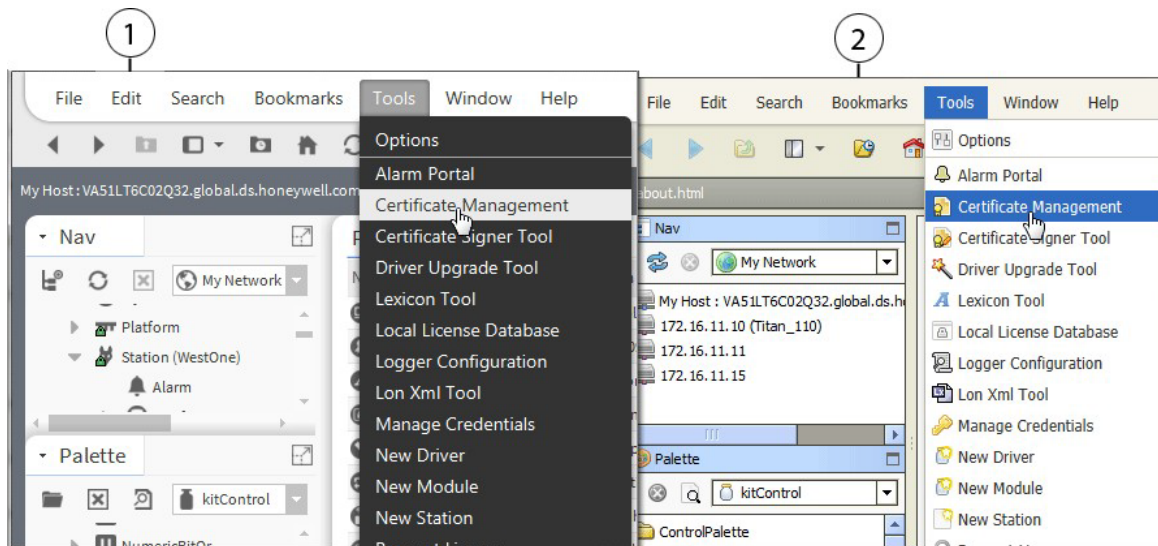
Workbench provides standard built-in themes as part of the default application environment.

### Built-in themes

Following are three Workbench themes listed and illustrated below:

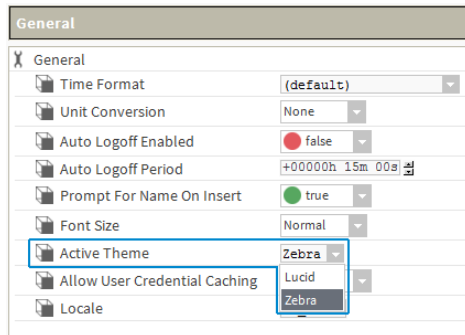
- Zebra (shown below as theme 1 on the left)  
This theme has a low contrast, grayscale coloring and is the default theme.
- Lucid (shown below as theme 2 on the right)  
Displays a blue and gray color scheme.

Figure 89: Workbench Differences between built-in themes: 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. In order for the theme change to take effect, you must exit Workbench and restart it.

Figure 90: 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 will not cause those changes to be saved.

# CHAPTER 3 DATA AND CONTROL MODEL

## Topics covered in this chapter

- ◆ Wire Sheet object management
- ◆ About control points
- ◆ About point extensions
- ◆ About control triggers
- ◆ About point status
- ◆ About writable points
- ◆ About composites

In any Niagara station (Supervisor or controller), all real-time data are normalized within the station database as points, a special group of components. Each point represents one data item.

## Wire Sheet object management

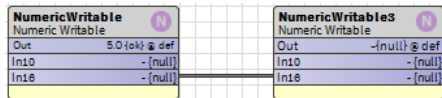
An object on a Wire Sheet view of a data model is a point component that represents a piece of data.

### Basic object linking

Object links define the direction of data flow.

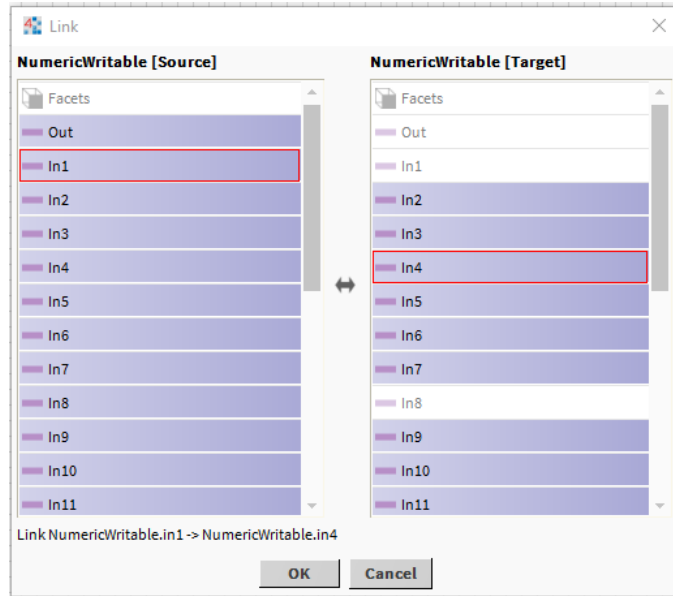
**Step 1** Move the mouse over the **In** or **Out** node of a wire sheet object (glyph) until the area is highlighted and the mouse pointer changes to a white arrow.

**Step 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.

Figure 91: Link window



**Step 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.

## Continuous object linking

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.

**Step 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.

**Step 2** To deactivate the link state, release the Shift key and click anywhere or touch any key.

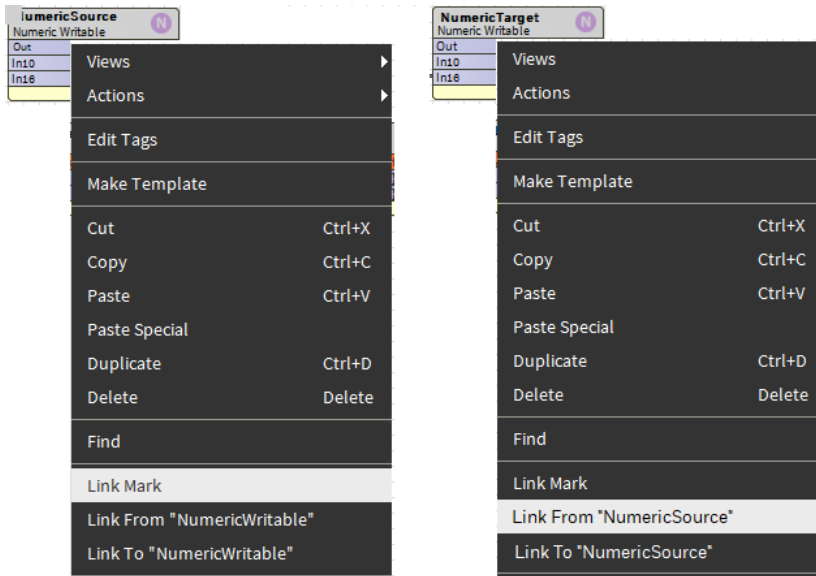
## Creating multiple links at the same time

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.

**Step 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.

Figure 92: Link Mark or Link From command



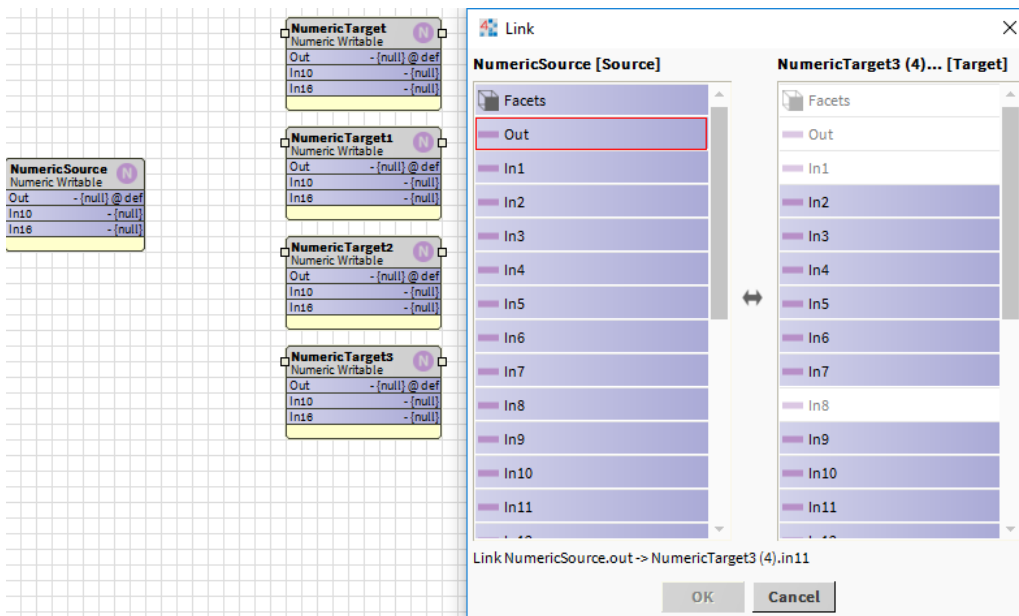
**Step 2** To define one or more selected objects as a link source or link target, select the Link From command.

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.

**Step 3** 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.

Figure 93: Link window



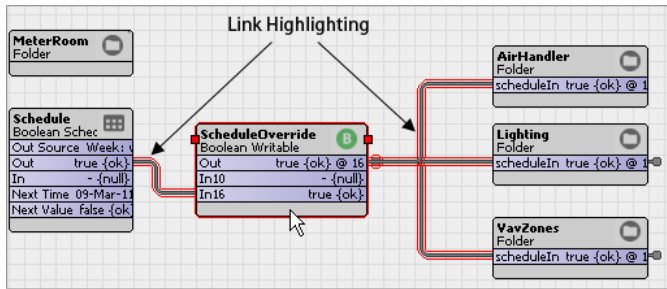
## Viewing links

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.

Step 1 Select a component on the Wire Sheet.

All links associated with the selected object display with a colored outline around them.

Figure 94: Link highlighting option



NOTE: Highlighted links do not mean that the LINK is selected.

Step 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).

Step 3 To customize the colors, edit the `system.properties` file (in the Nav tree `lib` folder, under `My-Files` → `Sys Home`).

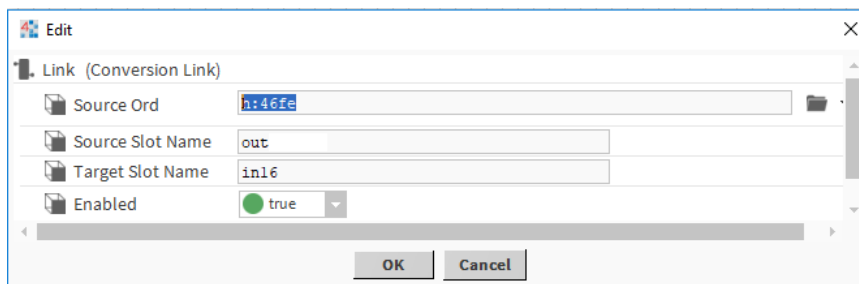
You specify each color using the standard hexadecimal notation used for HTML color display.

## 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

Step 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.

Figure 95: Edit window



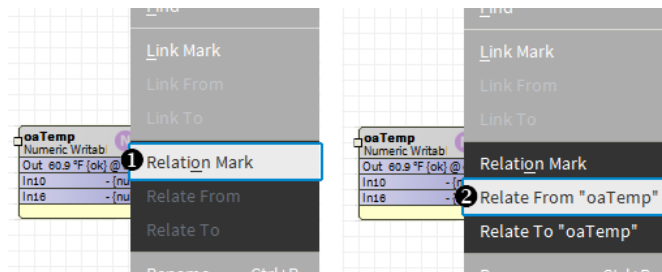
Step 2 Use the property fields to change any of the following link property values: Source Ord, Source Slot Name, Target Slot Name, or Enabled and click OK.

## Organizing objects in a hierarchy

You add relations between 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.

**Step 1** To define one or more selected objects as a relation source or relation target, select the **Relate Mark** command.

Figure 96: RelateMark command



**Step 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.

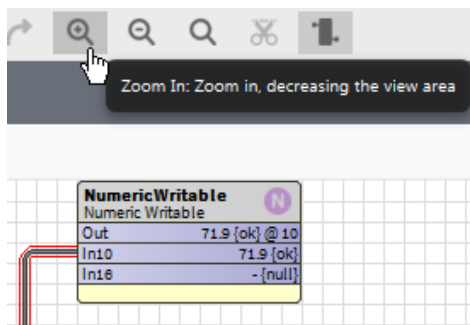
**Step 3** Select the **Relate To** command to define one or more selected objects as a relation source or relation target.

This command opens the Relation window with the marked object as the target object.

## Zoom controls

Complex wire sheets can be difficult to manage. The zoom controls let you magnify and reduce the screen images as needed.

Figure 97: 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.

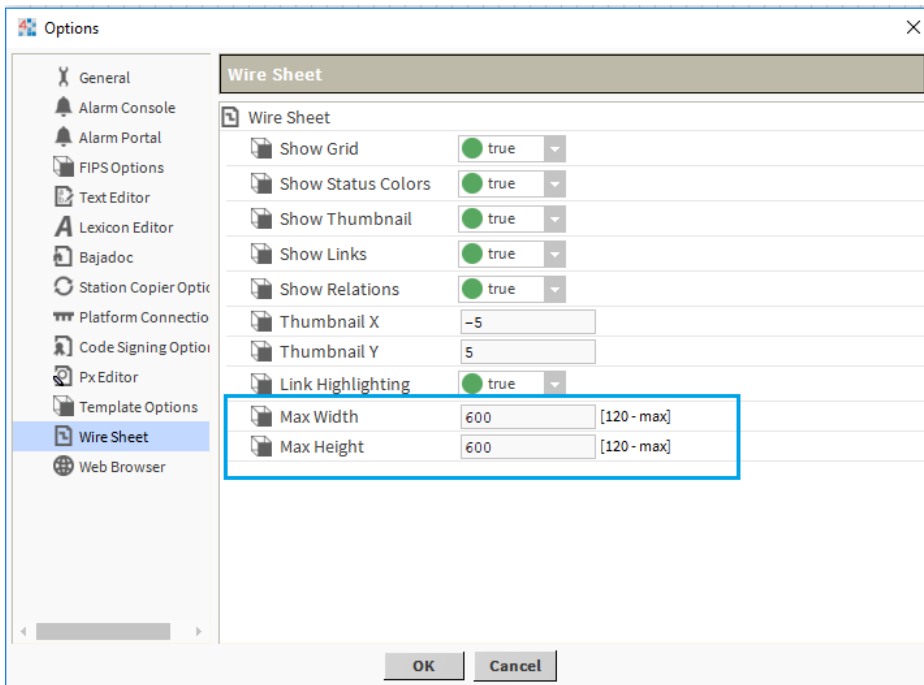


## Keeping all objects within view

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.

- Step 1 From the Workbench menu bar, click Tools→ Options.  
The Options window opens.
- Step 2 In the left pane of the Options window, click the Wire Sheet option.

Figure 98: Wire Sheet option

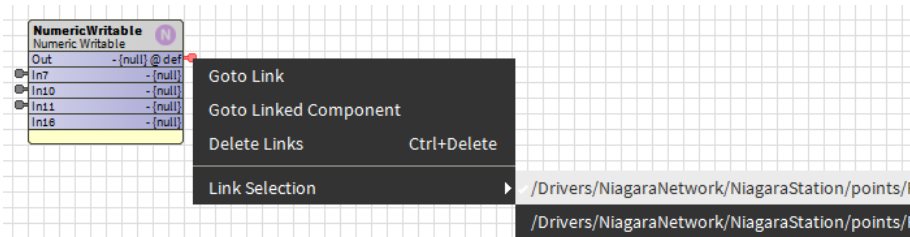


- Step 3 Enter a desired maximum value (in pixels) in the **Max Width** and **Max Height** fields and click OK.

## Link navigation

Link knobs provide easy navigation.

Figure 99: Link Selection menu



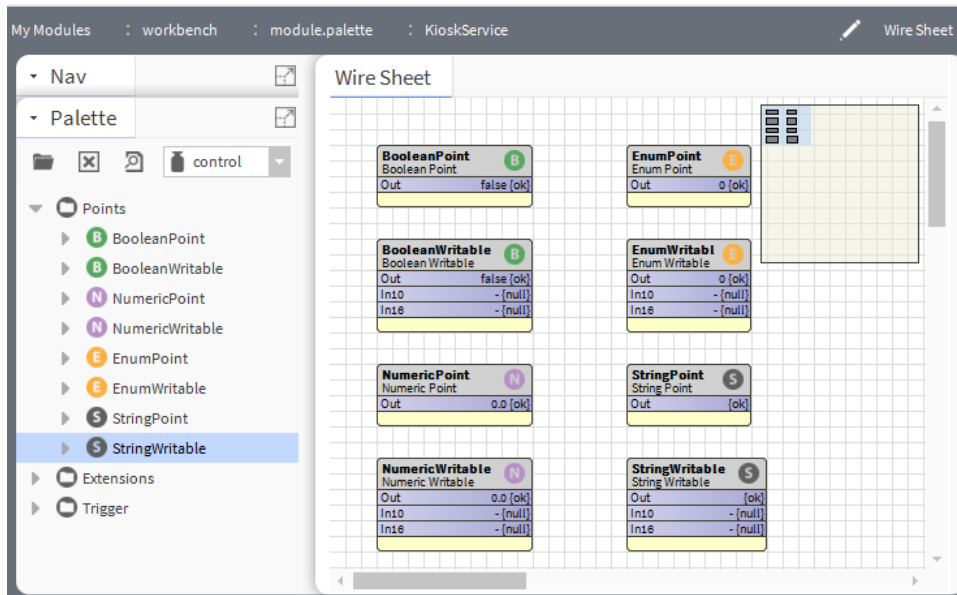
- Off-view linking  
Double-clicking on the knob hyperlink at the opposite end of the link, displays the Wire Sheet view with the linked knob highlighted.
- Goto Link command  
Right-clicking on the knob displays a GoToLink command on the menu. Selecting this command hyperlinks to the component on the 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.

## About control points

Control points are the foundation for all points in the station, including all proxy points.

Figure 100: Simple control points



The framework supports eight simple control point components. Each reflects a combination of a data (value) category and a point type. You can find these points in `Points` folder of the control palette.

Boolean Category	Numeric Category	Enum Category	String Category
BooleanPoint	NumericPoint	EnumPoint	StringPoint
BooleanWritable	NumericWritable	EnumWritable	StringWritable

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

- Boolean, which represents a binary value with only two states, such as off or on.
- Numeric, which 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, which represents an enumerated state (more than two), such as a multi-speed fan with states off, slow, and fast. Enums are often called multi-states or discretes. States typically derive from established integer value/state name pairs.
- String, which represents one or more ASCII characters (and if alpha-numeric), often with some literal meaning.

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).

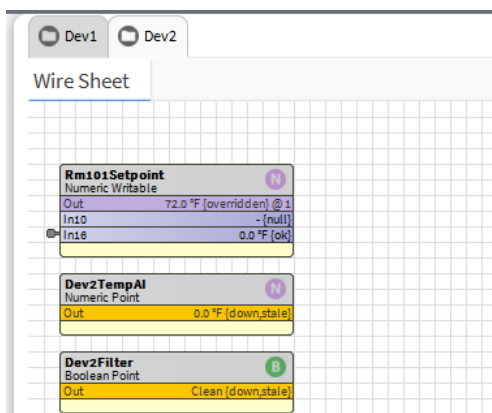
Other point components are found in both the kitControl palette. Briefly, these components include:

- 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. See the *Niagara KitControl Guide*.

## About point properties

Each point has input properties and a single output property (`Out`). A point's `Out` property provides real-time information.

Figure 101: 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.

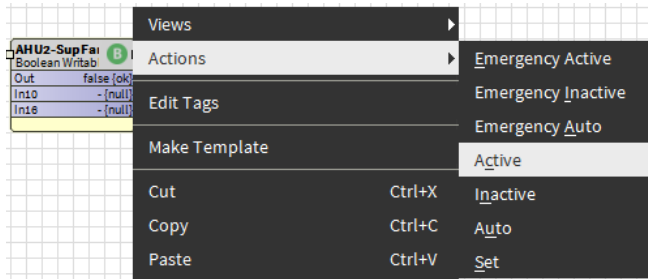
- The currently active priority level (from a 16-level priority scheme) for writable control points only.

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`, for example: `Off {ok} @ 1`.

## About point actions

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

Figure 102: Actions available on right-click menu



An action is a slot that defines a behavior. Some other control objects and extensions also have actions. In the case of the four writable control points, default actions include the ability to:

- Override the point at priority levels 8 (override) and 1 (emergency override), where control can be independently set or “auto’ed” at either level. A level 8 override can be for a defined (or custom) length duration, as specified in the action’s window.
- Set the value of the point’s **Fallback** property.

Often, you modify a writable point’s default actions.

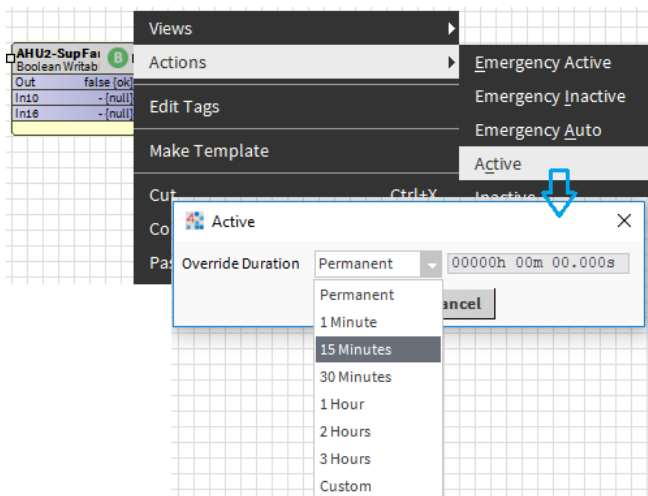
## About override actions

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.

A manual (level 8) override action to a point (not auto) prompts for an override duration, see the following image.

NOTE: Emergency overrides (level 1) do not have durations—these overrides are permanent (until auto’ed).

Figure 103: Override action duration (argument) window



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 duration in hours, minutes, and seconds.

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

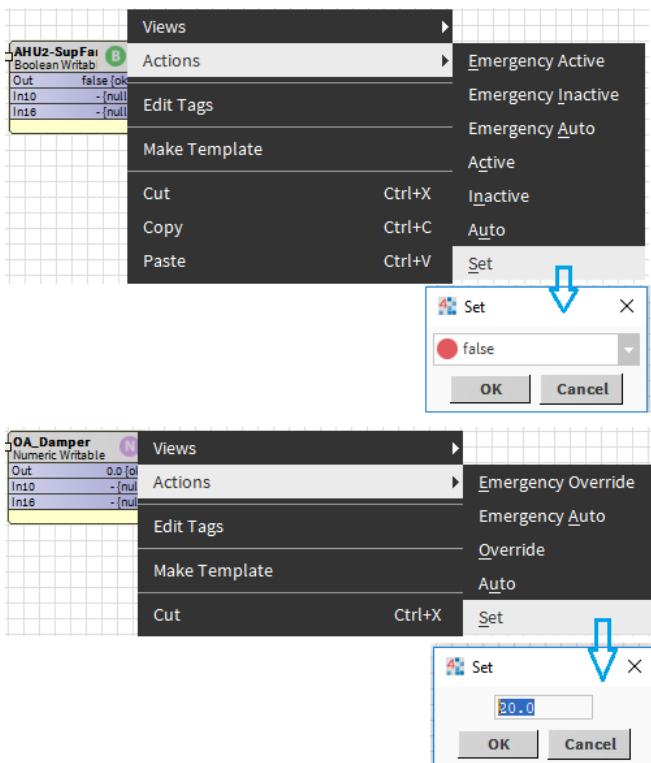
After clicking OK, the override action is issued to the point—if this is the highest effective priority level, the writable point operates under this control. If this is a timed override, the action is automatically auto'ed upon expiration of the override period.

## About set (Fallback) action

Whenever a writable point has a null or invalid value at inputs In1-In16 (note this means that both override levels are currently auto'ed), the Out slot is set to the value of the **Fallback** property.

By default, an operator-level user can change the **Fallback** property, using the Set action. This shows a window that displays the current Fallback value.

Figure 104: 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 from the point's property sheet.

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

NOTE: 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*.

## 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.

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*.

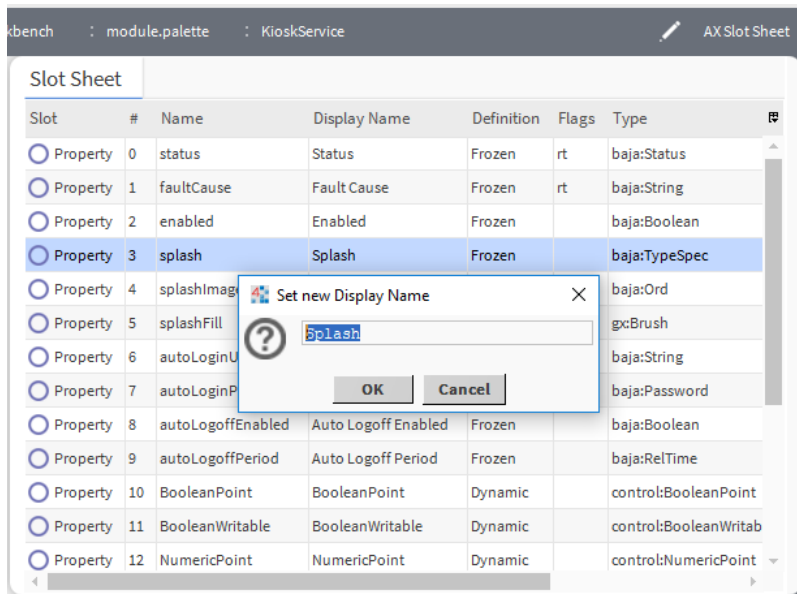
You can also modify the display name of the same action in multiple points in a single operation, using the (default) Batch Editor view of the station's ProgramService (Config→Services→ProgramService). This is one of many examples of using the Batch Editor.

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 in listed in bold.

Figure 105: 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.

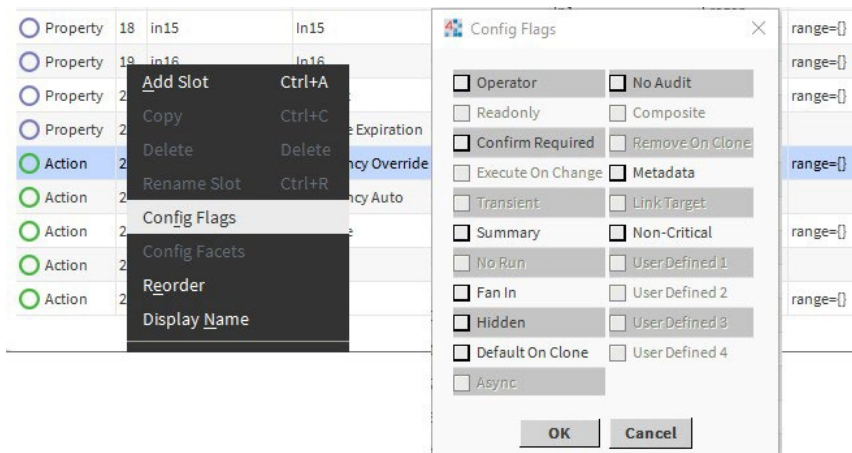
## Action access

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 106: 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 it 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.)

## 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.

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.

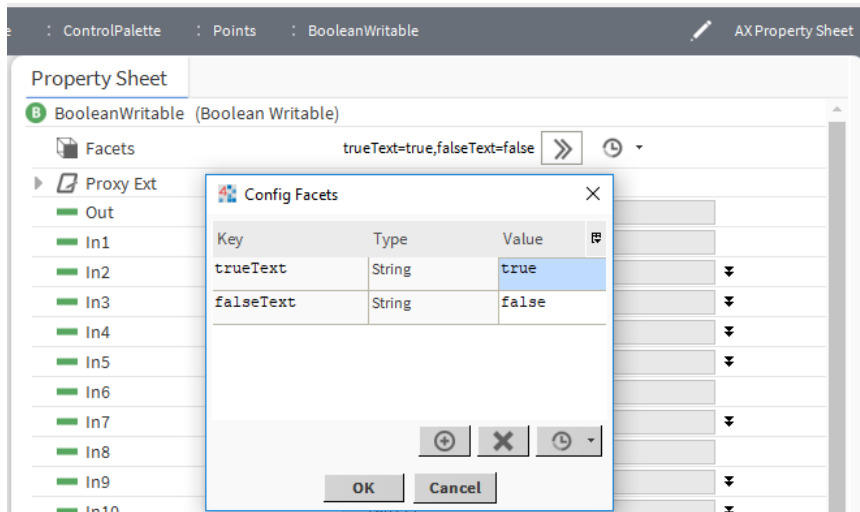
**NOTE:** Besides control points, various other components have facets too. For example, many kitControl and schedule components have facets. Details about point facets apply to these components too, unless especially noted.



## 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.

Figure 107: 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.

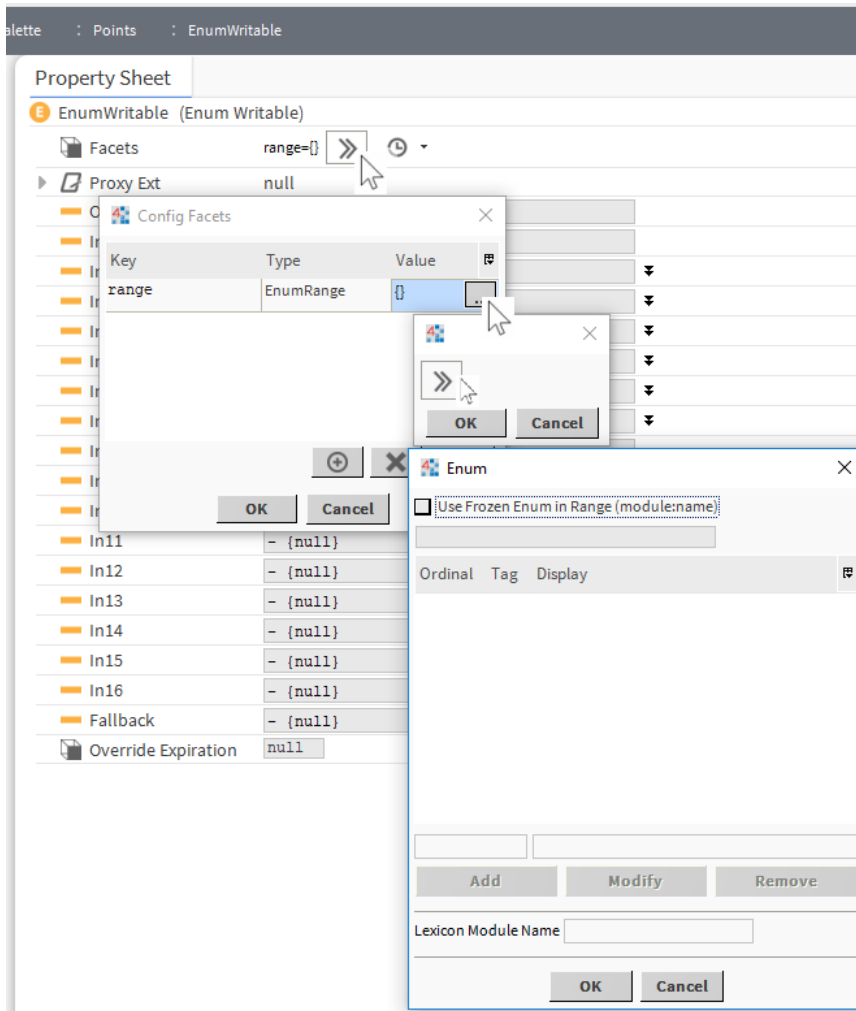
NOTE: For string-type points (StringPoint, StringWritable), facets typically have little practical application. By default, the Facets slot is empty for string-type points.

## 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.

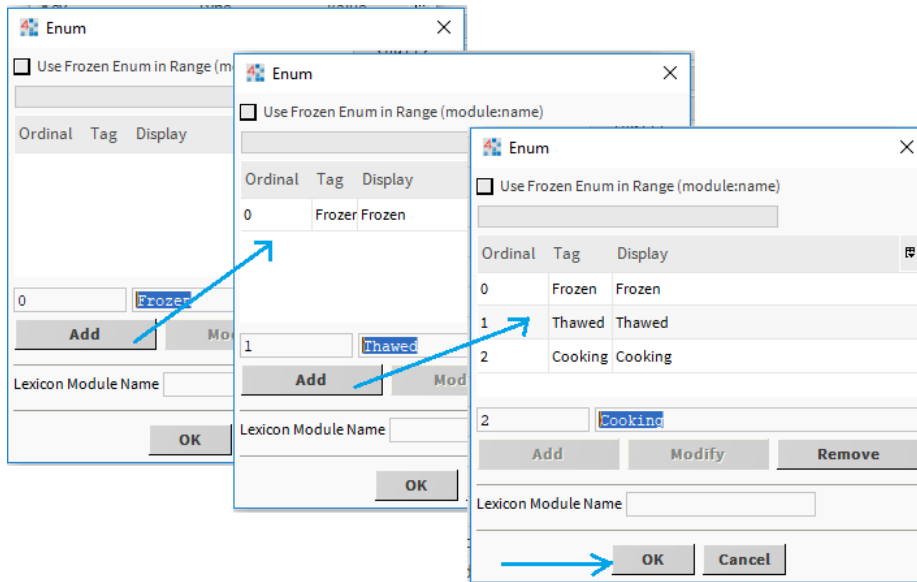
If you add an enum point from the control palette, its Facets slot has a blank range entry. Until you edit this facet and supply the ordinal-tag values, it can display only integer values. As shown, a special Enum window appears when you edit range facets.

Figure 108: 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.

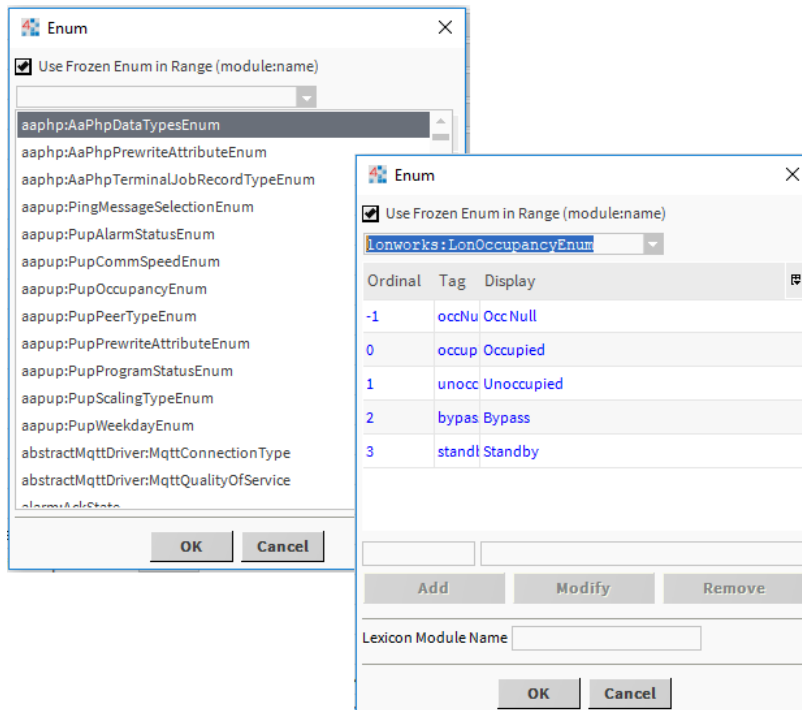
Figure 109: Type unique integer value in Ordinal box and associated text in Tag box



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 110: 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 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.

### Effect of facets on point actions

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.

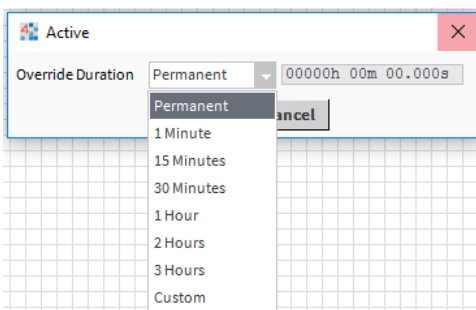
- Maximum override duration (for any writable point type)

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.

### 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.

Figure 111: Default override action menu for writable control point

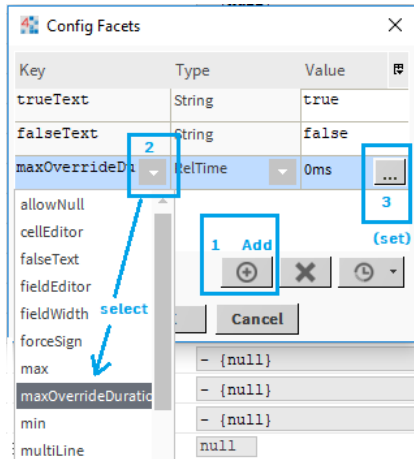


If needed, change this by adding a `maxOverrideDuration` facet (choosing type `baja:RelTime`), with specified duration time, to either or both:

- 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 112: Config Facets editor when picking maxOverrideDuration

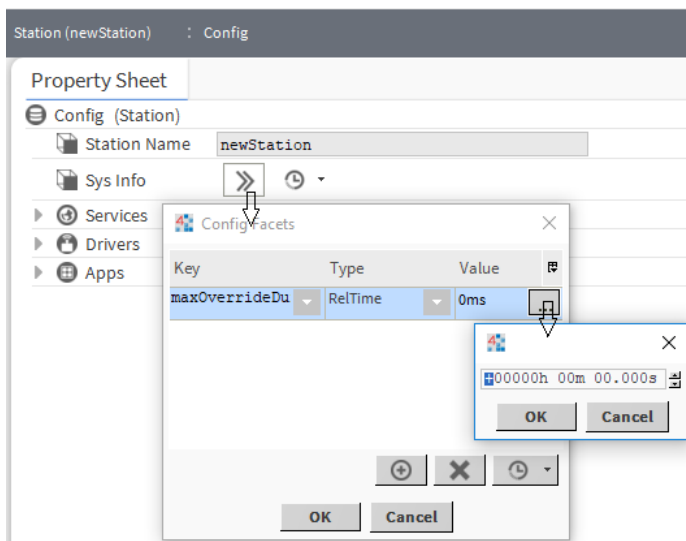


When a writable point is limited by a maxOverrideDuration facet, its action menu adjusts to show the allowable range.

### 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 113: Global maxOverrideDuration facet added to Sys Info property of station's Config component

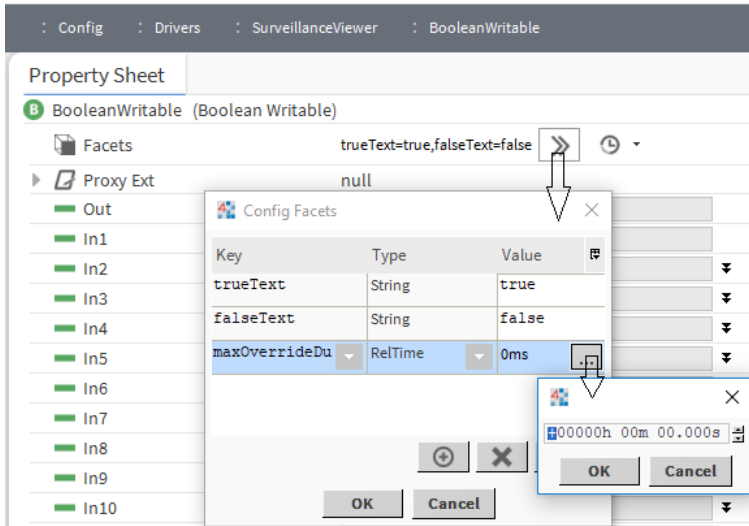


Adding this facet on the Sys Info property acts a global limit (station-wide) to a manual override action to all control points that do not have their own maxOverrideDuration facet.

### 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 114: Added `maxOverrideDuration` facet added at point level (overrides global setting)

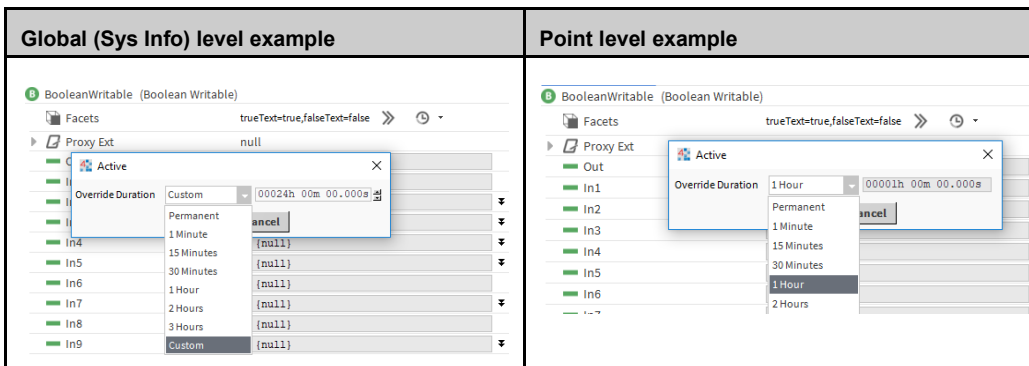


As shown, this `maxOverrideDuration` facet can be added along with any other facets in use by the control point. The example BooleanWritable point above already had configured facets for `trueText` and `falseText`.

### Action menu examples

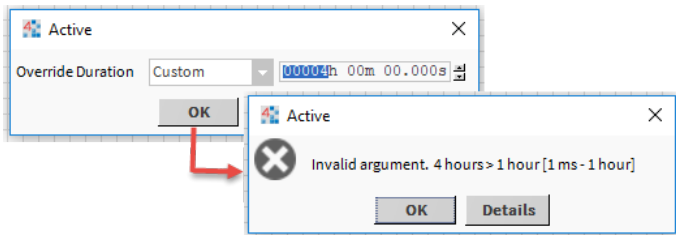
Example action menus from writable points with a `maxOverrideDuration` facet in effect are shown.

Figure 115: Example override action menus affected by `maxOverrideDuration` facet



Note 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 116: Custom override attempt over maxOverrideDuration limit produces error window



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

## About point extensions

As needed, you can add one or more extensions to a point, each as a child of that point. Extensions are a way to 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:

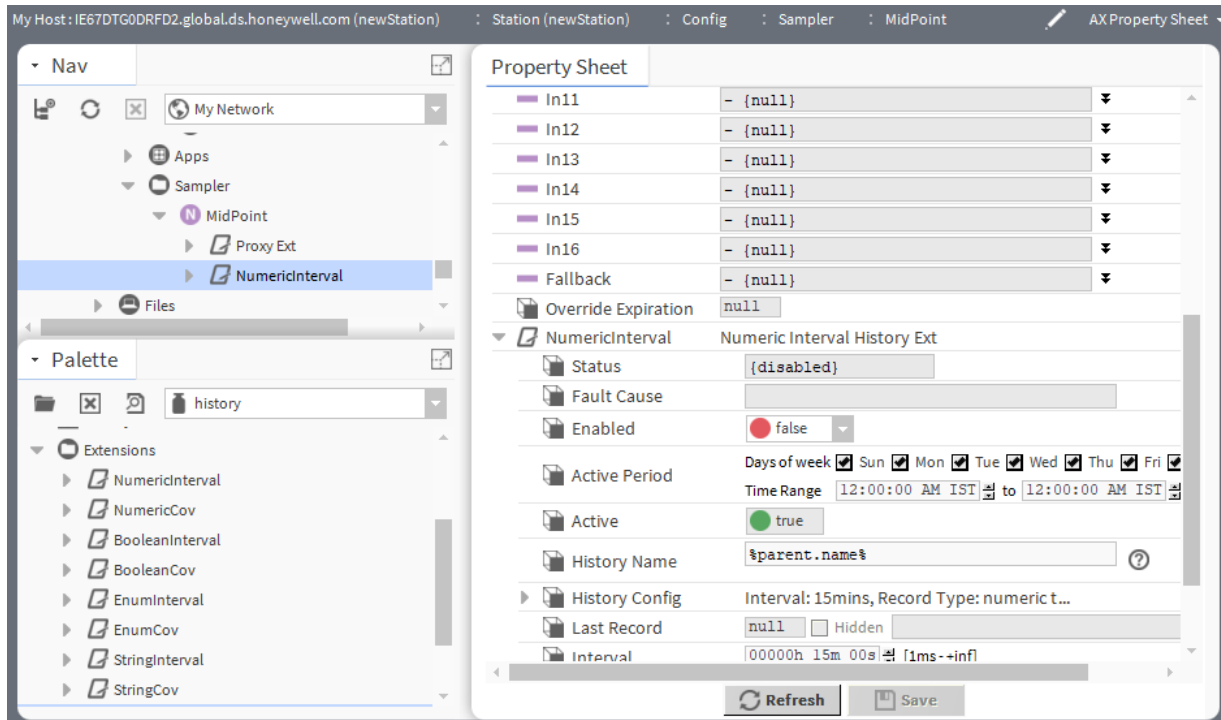
- Control extensions come from the control palette.
- Alarm extensions come from the alarm and kitControl palettes.
- history extensions come from the history palette.

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 as shown below.

Figure 117: Extension expanded in a point's property sheet



If a point has multiple extensions, they are processed in the same top-to-bottom order that they appear listed in that point's property sheet. You can re-order extensions in a point—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 originates, 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 simply null (NullProxyExt)—an empty placeholder. The station itself originates the point's default Out value.

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

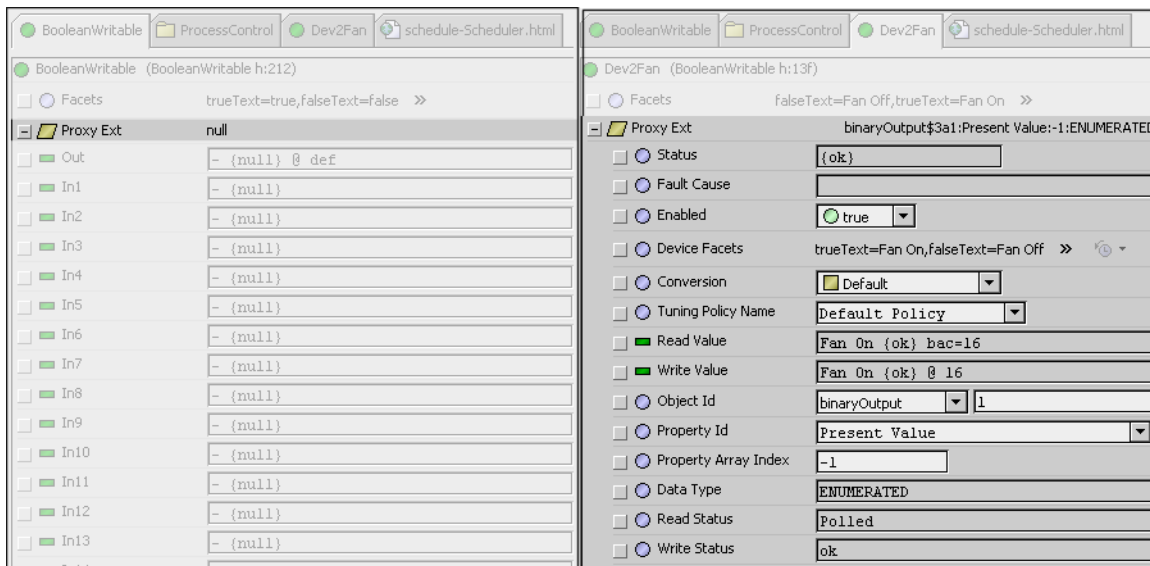
- <DriverType>

For any proxy point, meaning any of the 8 point types you create using the Points extension under a device represented in any of the network types, the proxy extension is <DriverType>ProxyExt. For example, a BooleanWritable proxy point under the Points container of a Bacnet Device 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.

The following image compares property sheet views of the proxy extension properties for two Boolean-Writable points, on the left a control point, on the right a BACnet proxy point.

Figure 118: Proxy extension for control point (null) and a Bacnet proxy point.

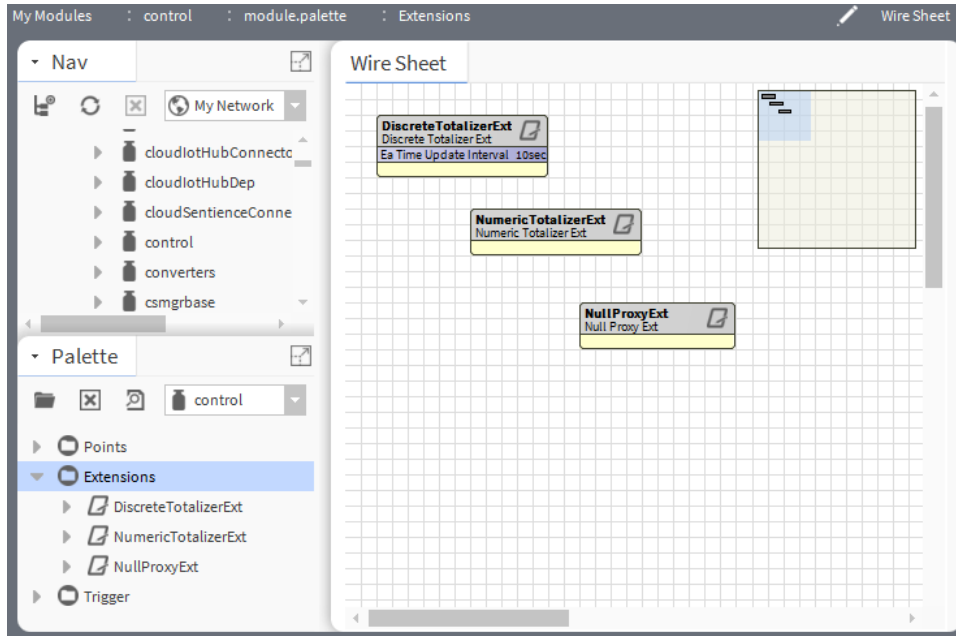


For more details, refer to the *Niagara Drivers Guide*.

## 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.

Figure 119: 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.

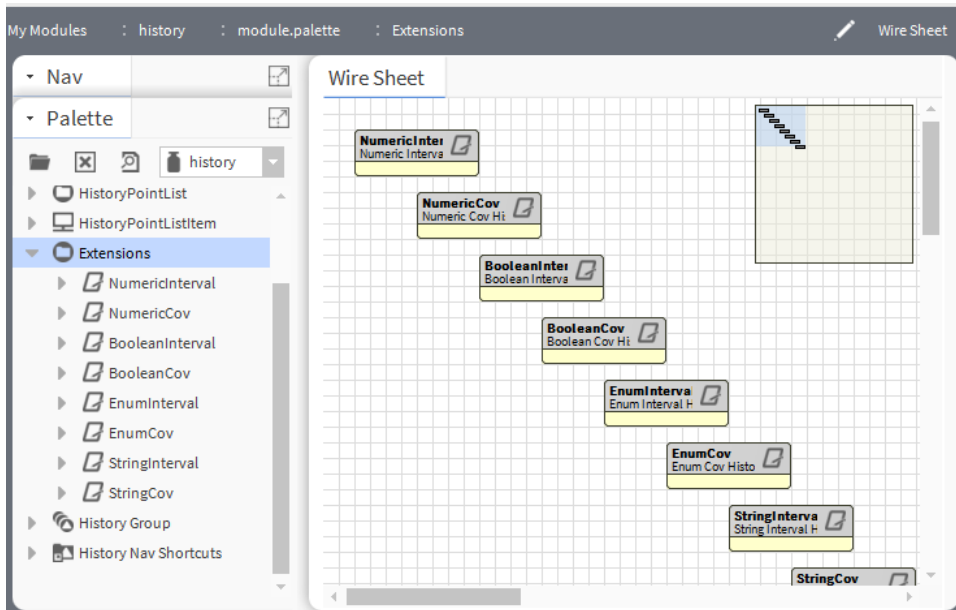
## About history extensions

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.

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 120: History extensions



Each history extension has various properties in two major groups:

- Collector properties determine how or when data is 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"	
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.

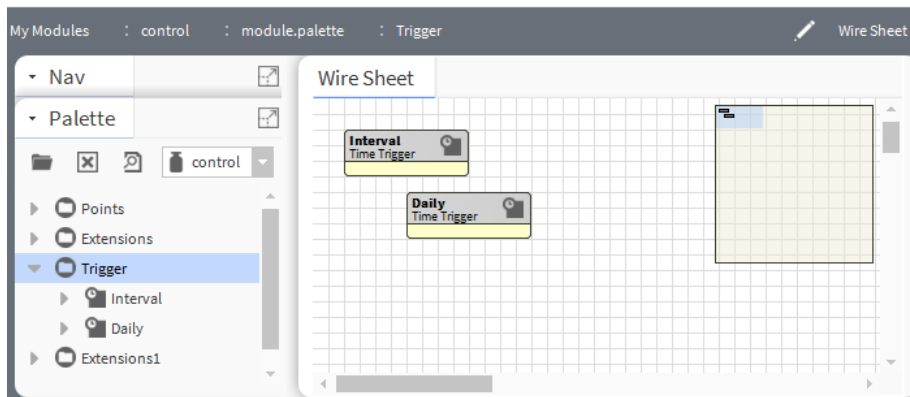
History extension type	Applies to point types		General description
	(read-only)	Writable	
StringCov	StringPoint	StringWritable	Collects upon each change of string value or status.
	—	any object with single String Out	

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.

## About control triggers

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

Figure 121: 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 needed.
- **Daily**  
Fires once a day at a specific time and day of 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.

## How triggers are used

To use a trigger, you typically link it to a selected action of a point extension (e.g. 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 `ResetElapsedActiveTime` action of a `DiscreteTotalizerExt` extension of a `BooleanWritable` point. In this case, that point's `DiscreteTotalizerExt` would only show runtime accumulated during the current day.

## About related objects

Other objects with trigger functions are found in various palettes. The most closely related is the `Trigger-Schedule` object, found in the `schedule` palette.

## About point status

Along with point value, point status is available at point Out. Status reflects status flags, 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 `baja` lexicon, using Workbench's Lexicon Editor. For a default English installation, to change default status appearance settings, edit the `en: baja` lexicon.

## Types of status flags

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  <b>65.0 °F</b>	Point currently has a value in an alarm range, as defined by property in its alarm extension.
fault	black text, orange background  <b>208.8 °F</b>	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  <b>72.0 °F</b>	Current point control is from an action, meaning a user-invoked command at either priority level 8 (override) or priority 1 (emergency).
disabled	gray text, light gray background  <b>79.0 °F</b>	Originates from a proxy point only. Point (or its parent device or network) has been manually disabled (property <code>enabled = false</code> ).
down	black text, yellow background  <b>68.4 °F</b>	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  <b>73.4 °F</b>	Originates from a proxy point only. Driver communications have not received a requested response for this data item within the configured times (Tuning period).

Type	Default Colors, Example	Meaning
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 Niagara KitControl Guide 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.

## Priority of status indication

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*.

## How status flags are set

Status flags are set differently depending on the type of point or control object. The following sections explain:

- Simple control point status
- Propagate Flags status option (linked Math and Logic objects)

NOTE: For more details on proxy point status, see *Niagara Drivers Guide*.



## Simple control point status

For simple control points (NullProxyExt) the following status flags are the only ones set and cleared:

- alarm  
Point is currently in an alarm condition as defined in its alarm extension.
- unackedAlarm  
Point has alarm extension assigned to an alarm class requiring acknowledgment, but the last alarm event has not yet been acknowledged. Point may/may not be in alarm.
- override  
Writable points only, during a user-initiated action at priority levels 8 (override) or 1 (emergency). The override flag clears when the action times out, or when a user issues an auto action at that same priority level.
- null  
Writable point has null or otherwise invalid value at In1—In16, plus null configured as Fallback value.

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

Figure 122: Status with simple control points

The screenshot displays the Niagara 4 configuration interface. On the left, a navigation pane shows a network hierarchy with 'My Network' selected. Underneath, several control points are listed, including 'OffnormalAlgorithm'. The main area is divided into three panels:

- Wire Sheet:** Shows two control points:
  - BW\_Fan:** Boolean Writable. Status: **Out: false [fault,down,sts]**. Inputs: In10: -(null), In16: -(null).
  - NW\_Setpt:** Numeric Writable. Status: **Out: 81.5 [alarm,overridden]**. Inputs: In10: -(null), In16: -(null).
- Property Sheet:** Configured for 'Offnormal Algorithm (Out Of Range Algorithm)'.
 

High Limit	80.0
Low Limit	60.0
Deadband	1.0 [0.0--inf]
High Limit Text	
Low Limit Text	

 At the bottom, there are checkboxes for 'Limit Enable', 'Low Limit Enable' (checked), and 'High Limit Enable' (checked).

## 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.

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

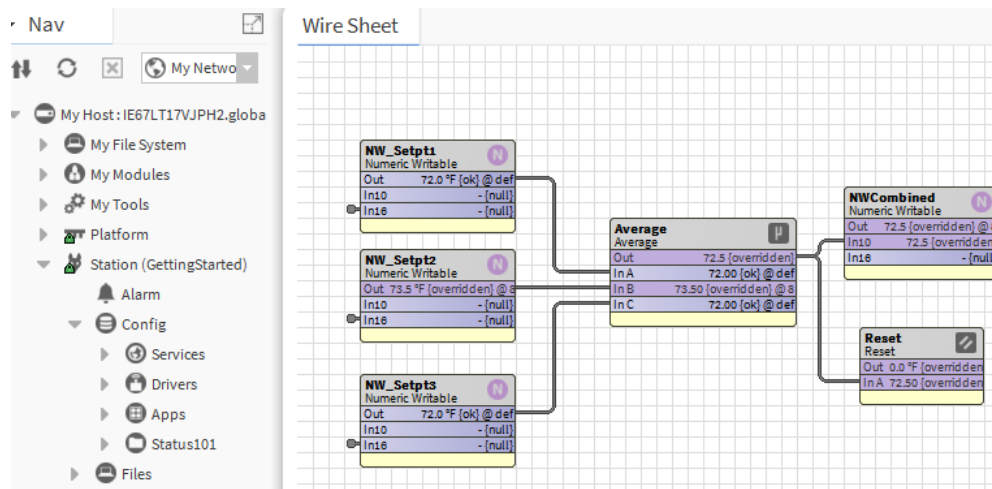
- disabled
- fault
- down
- alarm
- overridden

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 123: 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.

## 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.

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.

## About writable points

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.

## About the priority scheme

Each writable point uses a 16-level priority scheme, with corresponding inputs In1—In16, plus a **Fallback** property. Level 1 is the highest priority, and level 16 is the lowest.

## 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).

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.

## Priority linking rules

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 BooleanWritable, level 6 is also unavailable.

Priority levels 1 and 8 are reserved for actions (emergency and override). Priority level 6 in a BooleanWritable 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). That input could be linked to multiple priority type outputs, including those with duplicate priority levels and/or levels also used for object commands (emergency and manual).

## Priority level conventions

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.

## About the priority scheme

Each writable point uses a 16-level priority scheme, with corresponding inputs In1—In16, plus a “Fallback” property. Level 1 is the highest priority, and level 16 is the lowest.

The following topics further describe the priority scheme:

- Priority input scan
- Priority linking rules
- Priority level conventions

## About minimum On and Off times

Each BooleanWritable 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 BooleanWritable 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 minimum timer is triggered by a state change transition to active or inactive. This results in the new state value being stored 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 BooleanWritable point controls a fan, with related properties set as follows:

Property	Value	Description
Min Active Time	00000h 01m 30s	Specifies that once started, the fan must run at least 90seconds.
Min Inactive Time	00000h 03m 5s	Specifies that once stopped, the fan must remain stopped at 3minutes, 5 seconds.

Starting with the fan stopped at schedule level (priority 16), if a user gives it a manual override on (priority level 8), the fan will run for 90 seconds at priority level 6 (a higher level). After this period, the fan continues running at the override 8 level for the duration of the override.

During the initial 90 seconds, a different override action (off or auto) will be ineffective—as the higher priority level 6 remains in control.

Once stopped, the point's minimum off time will keep the fan off at priority level 6 for the specified duration (in this example, 3 minutes and 5 seconds). During this period, only an emergency command or input change at In2—In5 can effect further change.

## 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.

NOTE: 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 Note above.

### Some composite examples

A few simple examples of composites are as follows:

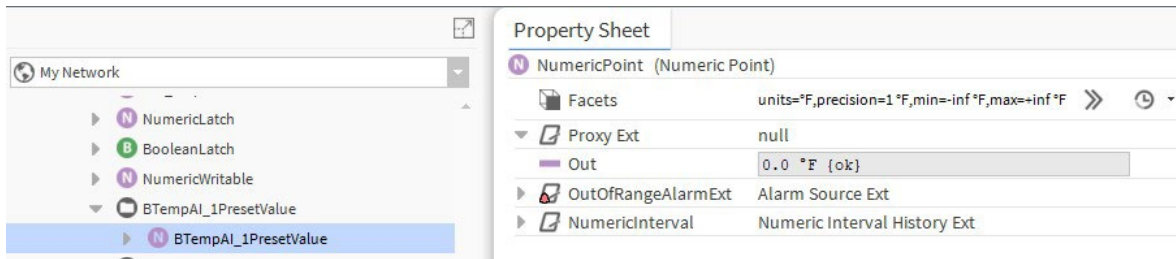
- Point-level composite
- Folder-level composite

## Point-level composite

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

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

Figure 124: Property sheet of proxy point with two extensions

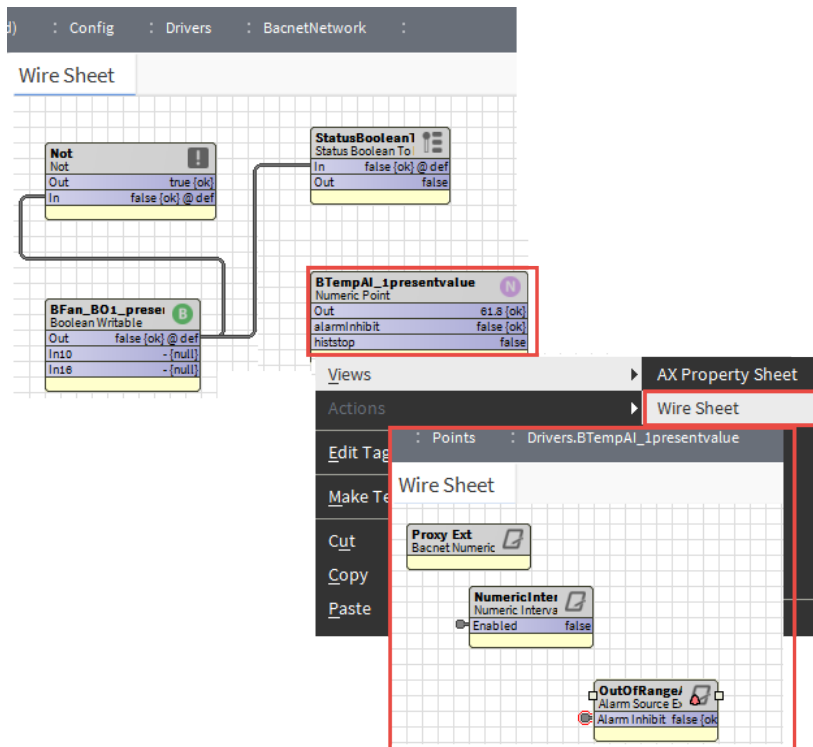


In this example, when another system-related BooleanWritable (representing the system fan) has a false (Off) status, it is desired that this temperature point be:

- 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 kit-Control objects needed. Without making a composite, the necessary objects and links may appear.

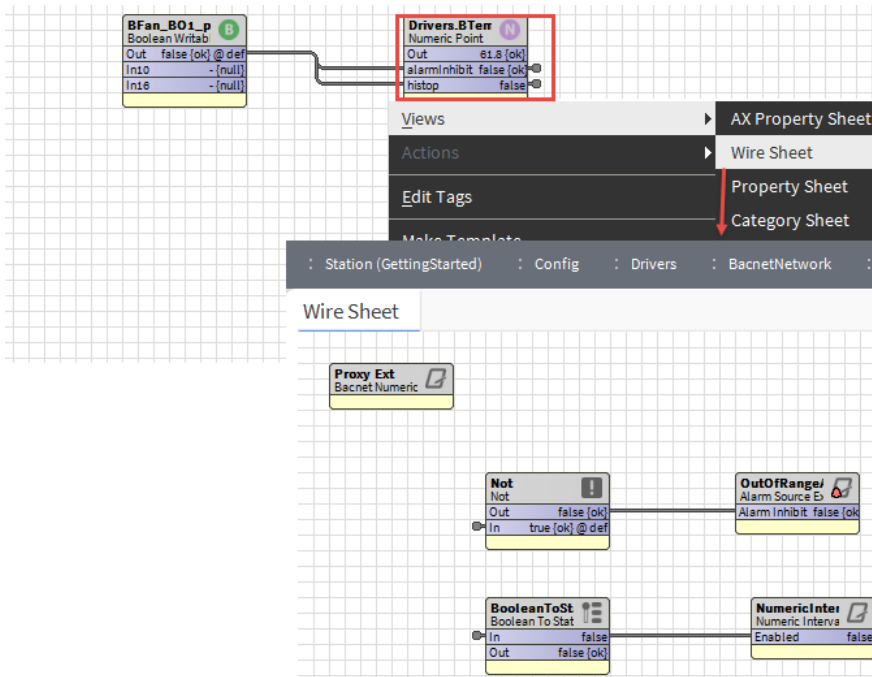
Figure 125: Proxy point example without composite



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.

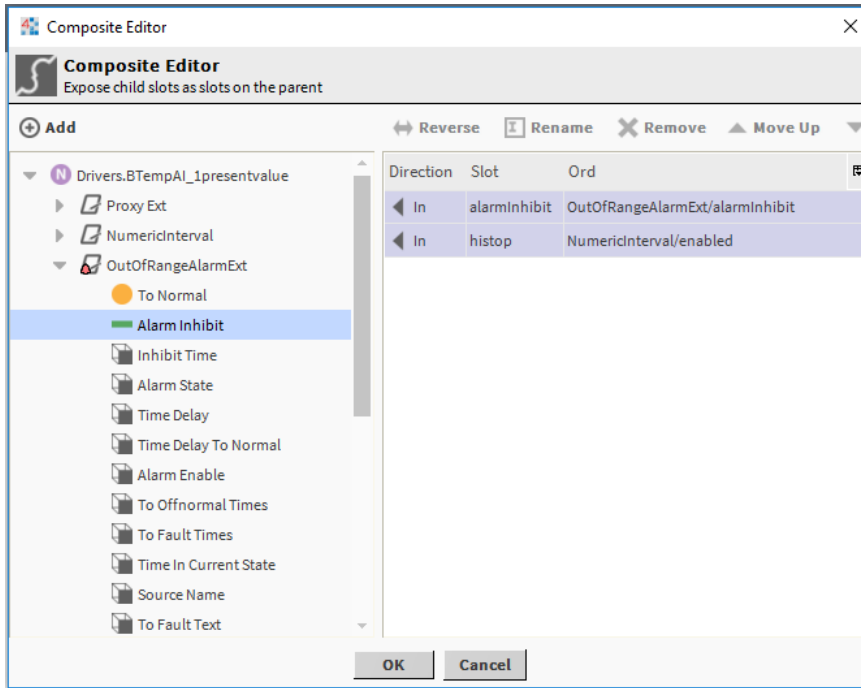
Figure 126: Proxy point as composited container





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

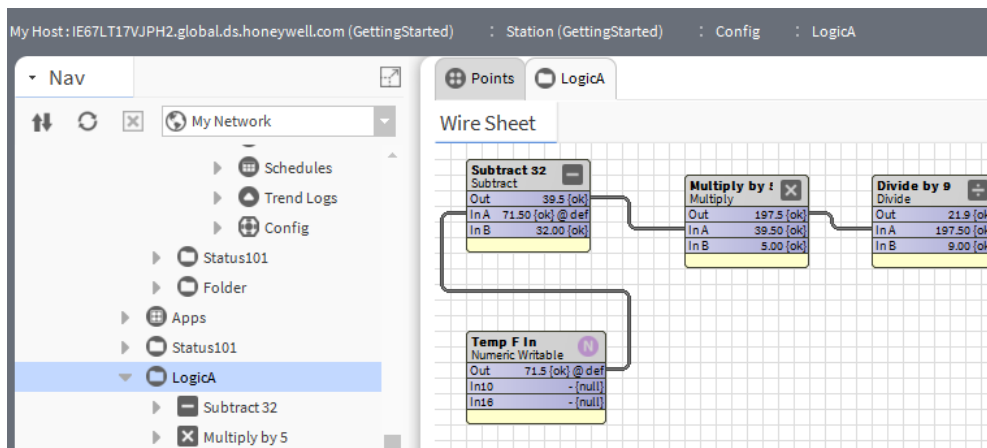
Figure 127: Composite Editor for example NumericPoint



## Folder-level composite

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

Figure 128: 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.

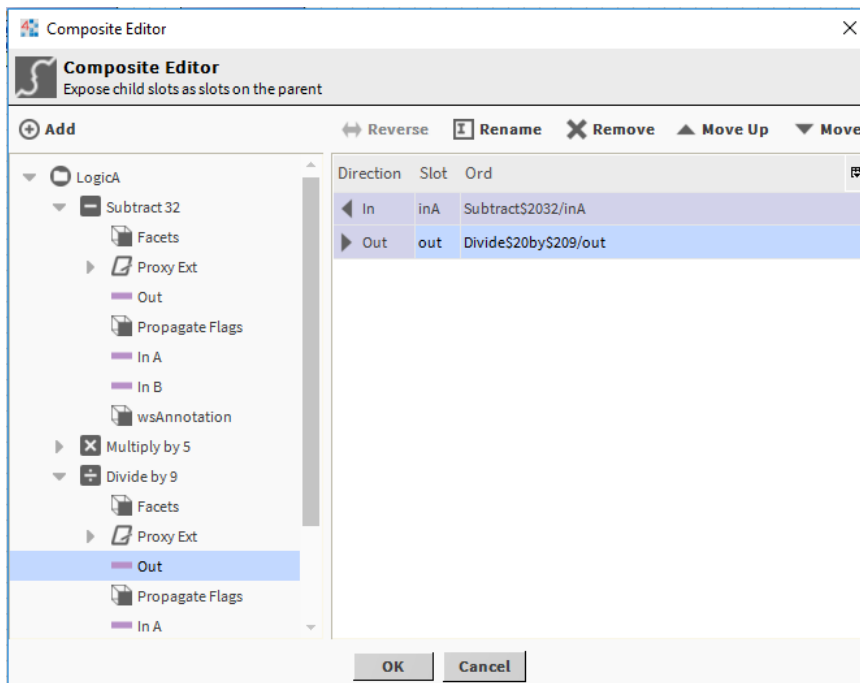
If this application was needed later, you could copy all ttree linked objects again and insert them in another (perhaps already crowded) wire sheet. However, the middle Multiply object reveals an intermediary result that is distracting.

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.

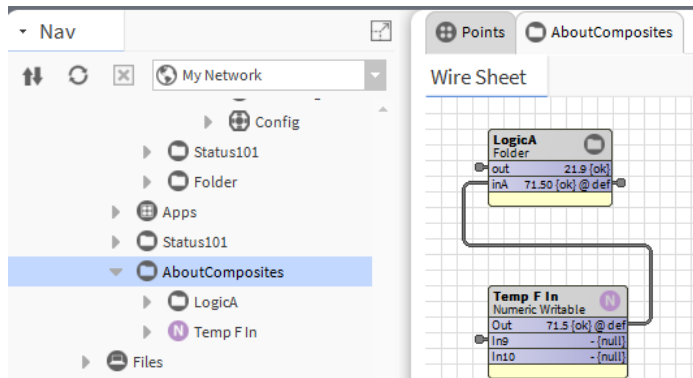
Figure 129: Launching and using the Composite Editor



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 130: 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.

## 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 pins 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.

# **CHAPTER 4 ABOUT WORKBENCH TOOLS**

## **Topics covered in this chapter**

- ◆ Alarm portal
- ◆ Security management tools
- ◆ Driver Upgrade Tool
- ◆ Embedded Device Font Tool
- ◆ Jar Signer Tool
- ◆ Kerberos Configuration Tool
- ◆ Lexicon Tool
- ◆ Local License Database
- ◆ Logger Configuration tool
- ◆ Lon Xml Tool
- ◆ Manage Credentials
- ◆ Module Info View
- ◆ NDIO to NRIO Conversion Tool
- ◆ New Driver Wizard
- ◆ New Module Wizard
- ◆ New Station wizard
- ◆ Request License
- ◆ Resource Estimator
- ◆ Time Zone Database Tool
- ◆ Todo List
- ◆ Workbench Fox Analyzer
- ◆ Workbench Job Service
- ◆ Workbench Library Service
- ◆ Workbench Service Manager

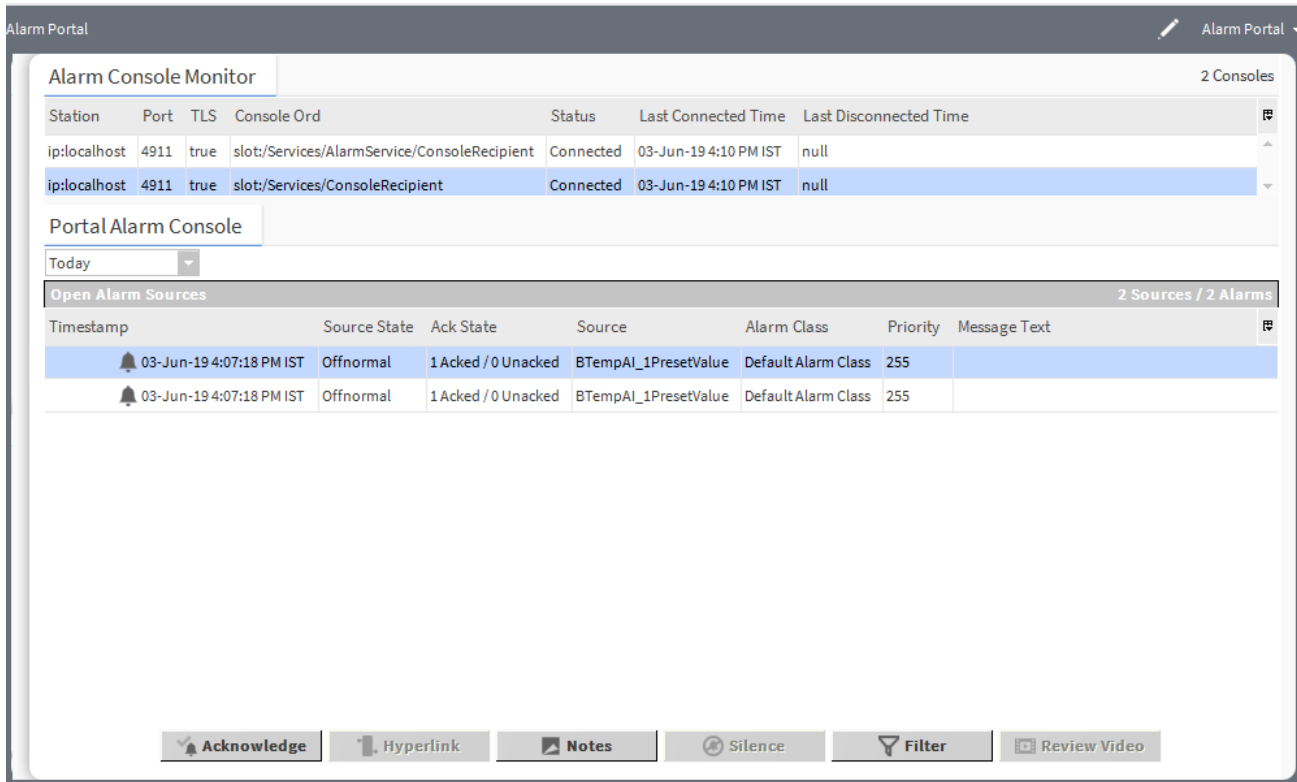
This section describes some of the standard tools available in Workbench. These tools provide views that are designed to facilitate specific tasks – from managing credentials to monitoring alarms. All of the tools described are available from the Tools menu. However, navigational links to some tool also appear in the Nav side bar and in the Nav Container View.

When you first open Workbench, the My Tools node will not be present in the Nav tree. Therefore, if you configure a tool, such as the BACnet service tool, all the configuration data are lost when you close the current session. Tools, such as the Alarm Portal, BACnet service, Lonworks service, and the Workbench Job Service are available when you do not have a station open. This is provided as a convenience and may only be useful in a limited number of situations. For example, you would probably use the Job Service, BACnet service, and Lonworks Service under the Services node of a specific station.

## **Alarm portal**

The Alarm Portal tool allows you to view and acknowledge alarms from many different stations using a single viewer (portal).

Figure 131: Alarm 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.

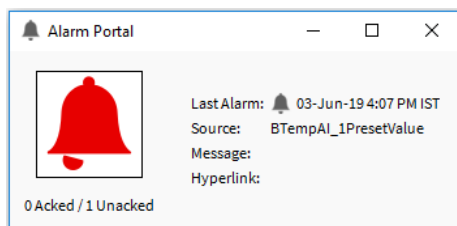
To add alarm consoles from different stations, right click in the Alarm Console Monitor (top pane) and choosing the Add Alarm Console menu selection to initiate the Add Alarm Console wizard.

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 132: Alarm portal window



To set options for the alarm portal, click the Tools→Options menu.

## Security management tools

Two 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. For more details, see *Niagara Station Security Guide*.

## Driver Upgrade Tool

Use this tool to upgrade the driver of remote station from the supervisor.

Figure 133: Driver Upgrade Tool

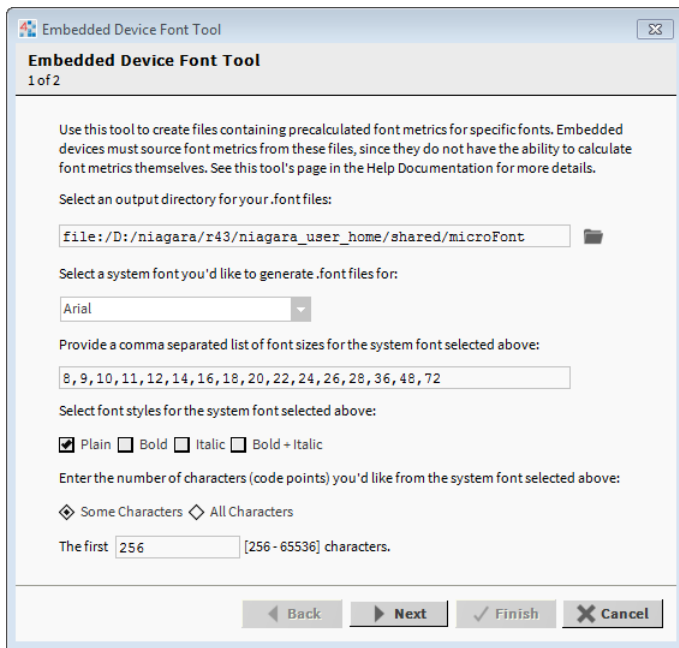
Host	Username	Password	Port
localhost	Pacrim1\H159983	••••••••	3011
Platform Connection	admin	••••••••	1911
Station Connection			

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.

## 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.

Figure 134: Embedded Device Font Tool



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 below in the image.

Figure 135: Java Virtual Machine

Architecture	armie-v7
Enabled Runtime Profiles	rt,ux,wb
Java Virtual Machine	oracle-jre-compact3-qnx-arm (Oracle Corporation 1.8.0.91.3)
Niagara Stations Enabled	enabled
Number of CPUs	1
Current CPU Usage	4%

- 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 you'll be using to view Hx/Px pages in a web browser.

You can install the .font files you have generated to this location on the embedded device:

```
My File System/User Home/shared/microFont
```

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" />
</defs>
```

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

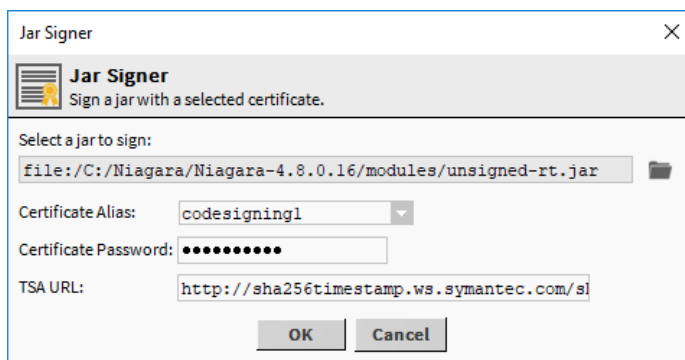
The framework will search for .font files in three locations in the following order:

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).

## Jar Signer Tool

Available in Niagara 4.8 and later, 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 unsigned \*.jar file using a code signing certificate.

Figure 136: 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).

### 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, starting in Niagara 4.8, if you attempt to install an unsigned module, it will cause a signature verification warning.

Prerequisites:

- You are running Niagara 4.8 or later version of Workbench.
- You have already created a code signing certificate. For details, see *Niagara Station Security Guide*).

- Step 1 In Workbench, click Tools→Jar Signer Tool to open the Jar Signer window.
- Step 2 Beside the field labeled **Select a jar to sign**, click the folder icon to the right.
- Step 3 In the file chooser window navigate to the unsigned jar file and click to select it, and click OK.
- Step 4 In the Alias field, click and select your code signing certificate.
- Step 5 Enter the Certificate Password.
- Step 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.

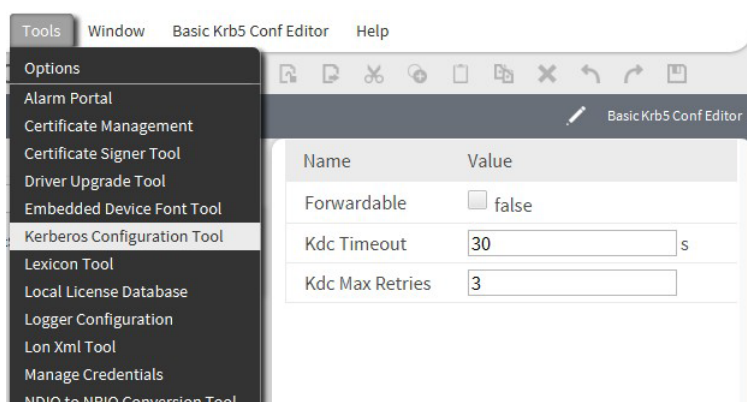
Step 7     Navigate to the desired location and click Save.

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

## Kerberos Configuration Tool

The Kerberos Configuration Tool provides an editor view, Basic Krb5 Conf Editor, which allows you 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 137: Basic Krb5 Conf Editor view



Additionally, the Kerberos Configuration Tool includes the Advanced Krb5 Conf Editor located under the Views dropdown 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*.

## Lexicon Tool

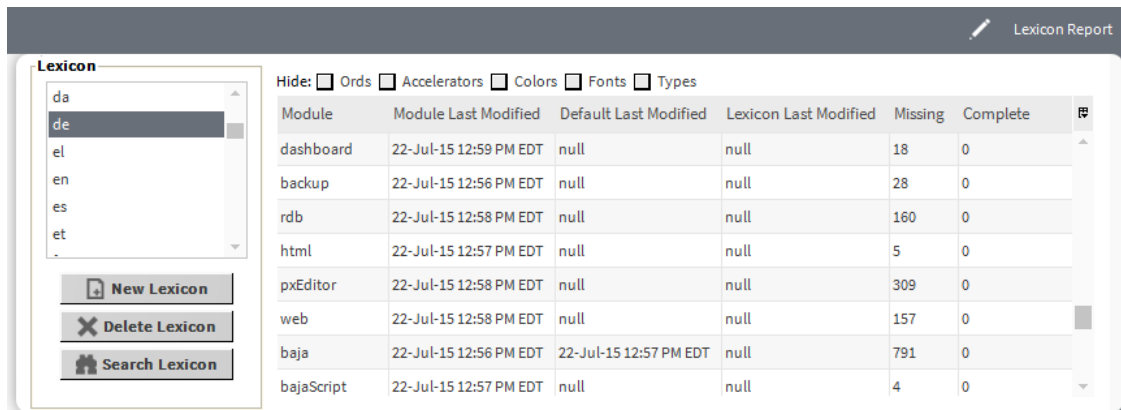
This tool controls the text of the user interface. You can use it to translate the user interface. For more detailed information, see the Niagara Lexicon Guide.

For details on installing (edited) file-based lexicons and (new/edited) module-based lexicons in remote JACE platforms, see 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.

Figure 138: Lexicon report view



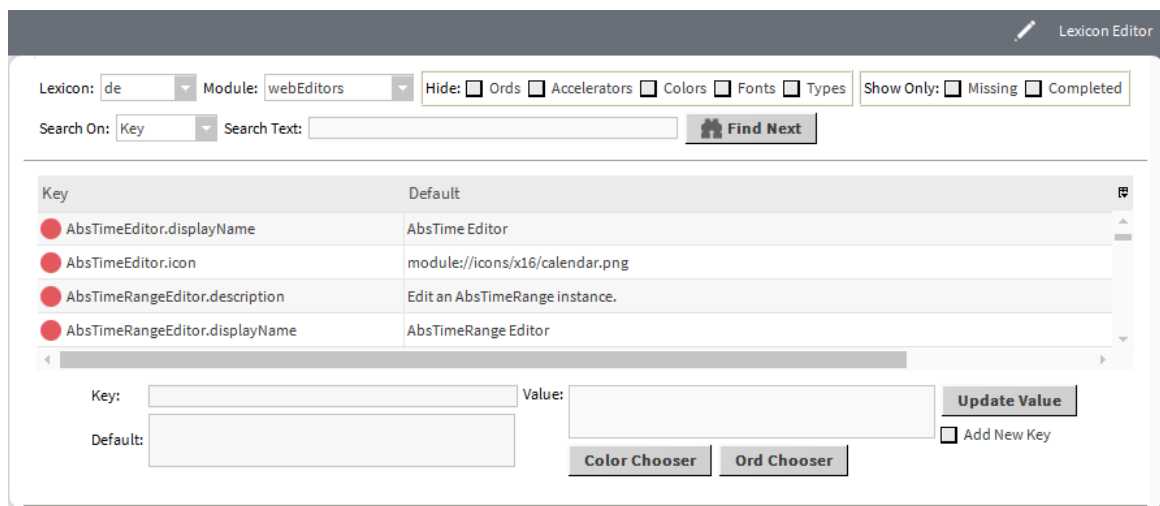
Check boxes at the top of report view allow you to hide values that could affect missing status counts.

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.

## 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.

Figure 139: 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). Starting in AX-3.7 the Value field 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 Niagara toolbar.

For detailed information on using the lexicon editor view, see the *Niagara Lexicon Guide*.

## Lexicon Module Builder

The Lexicon Module Builder view (shown below) 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 140: Lexicon module builder view

Source	Lexicon	File	Path
<input checked="" type="checkbox"/>	file	en	control.lexicon

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.

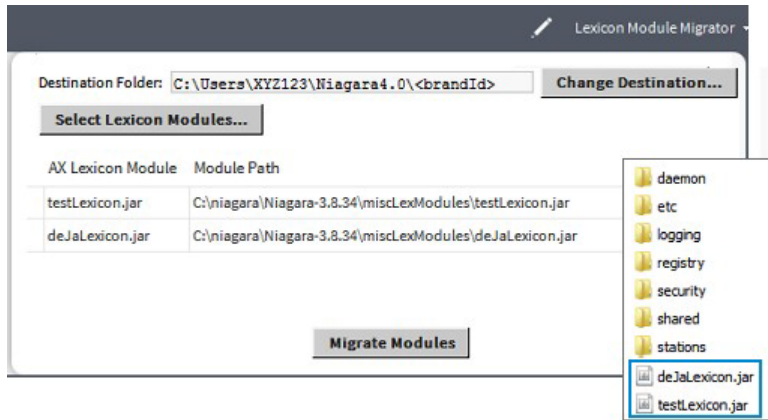
On completion of the build, you will see a confirmation message indicating that the module was constructed and placed in the !modules folder.

For detailed information on using the lexicon module builder view, see the *Niagara Lexicon Guide*.

## Lexicon Module Migrator

The Lexicon Module Migrator view (shown) allows you to migrate AX-3.8 lexicon modules to the Niagara 4.0 installation.

Figure 141: Lexicon Module Migrator view and migrated lexicon modules in user home folder



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.

NOTE: For any modules that are new in Niagara 4.0, such as webEditors, etc., you need to build new lexicon modules using the Lexicon Module Builder view.

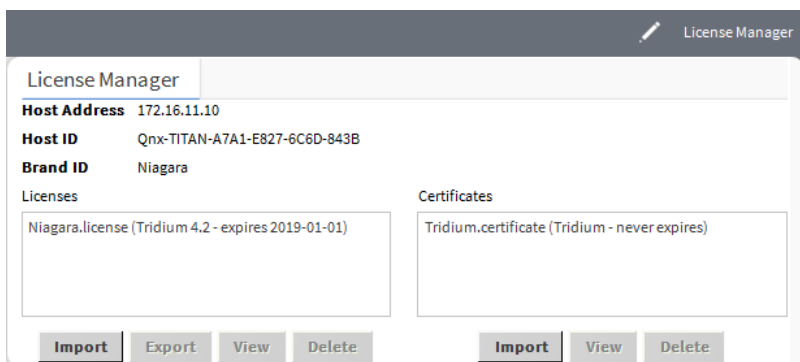
For detailed information on using the Lexicon Module Migrator, see the *Niagara Lexicon Guide*.

## Local License Database

The WorkbenchLicense Manager view is available, when you select Local License Database from the main menu.

The WorkbenchLicense Manager view lets you browse and manage the contents of your local license database.

Figure 142: Workbench 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  
Always available, this allows you to add license file(s) from a local license file or license archive (.lar) file.
- Export  
Always available, this allows you to save all licenses (or any selected licenses) locally, as a license archive file.
- Delete  
This allows you to delete licenses from your Niagara local license database.
- Sync Online

This is typically available if you have Internet connectivity. This lets you update 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*.

## 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.

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.

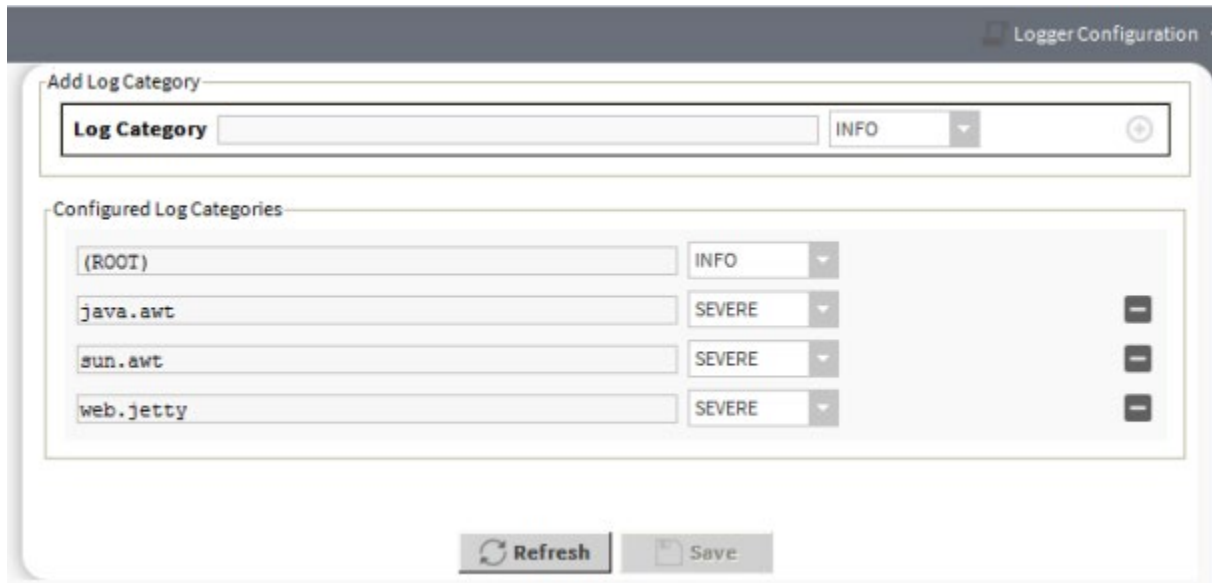
Logging configuration settings for local or Workbench logs are stored in the `!logging/logging.properties` file which exists in two locations, your Workbench User Home and the daemon User Home, and are maintained by the framework.

The Logger Configuration tool has seven severity levels which differ from the number of log levels used in NiagaraAX. Log levels used are mapped as shown here:

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 143: 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 144: logSetup

Index | logSetup | DEFAULT-alarm Sample [t]

The Log alarm has been set to default Log level

Log Configuration											
Log	Level	OFF	SEVERE	WARNING	INFO	CONFIG	FINE	FINE	FINE	ALL	DEFAULT
DEFAULT	WARNING	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Applicable
alarm	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
alarm.database	INFO	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
alarm.dataRecovery	INFO	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
backup	INFO	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
bacnet.link.ethernet	INFO	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
baja	FINE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
box	FINE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
box.ord	FINE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
box.reg	FINE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>


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):

## Configuring settings for local Workbench logs


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

NOTE: The procedures to configure log settings of a station are identical except that you must access the Logger Configuration view via the DebugService in the connected station.

### Adding a new log category

- 1 In Workbench, select Tools→Logger Configuration. The Logger Configuration view appears.
- 2 Click in the **Add Log Category** field and enter the new category name. Or, enter only the first letter of a name to see a list of existing logs (whose names start with that letter) to choose from.
- 3 Click the severity level dropdown list and select the desired log level.
- 4 Click  (add) to add the new category.
- 5 Click Save.

### Removing an existing log category

- 1 In Niagara, select Tools→Logger Configuration.
- 2 The Logger Configuration view appears.
- 3 Click  (remove), located to the right of the log category you wish to remove. The log category is immediately removed.
- 4 Click Save.

### Changing the severity level of an existing log category

- 1 In Workbench, select Tools→Logger Configuration. The Logger Configuration view appears.
- 2 Click the current severity level and select the desired level from the dropdown list.
- 3 Click Save.



## Viewing logged data

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

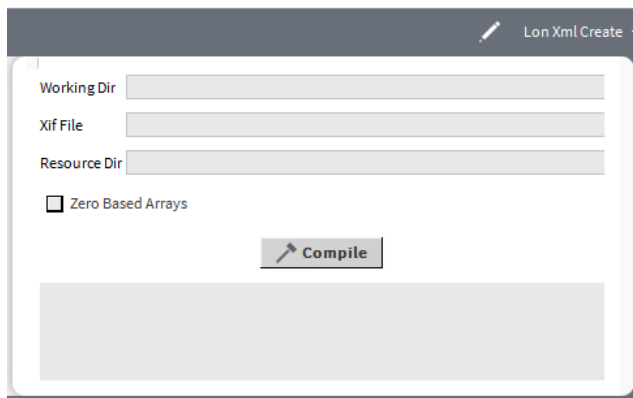
To view logged data for Workbench:

- Open the Niagara console started with Workbench.

## 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 (.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 145: Lon Xml Create tool view



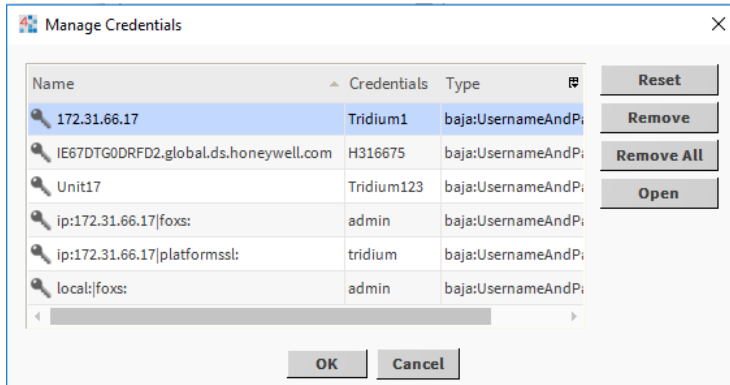
Refer to the *Lonworks Guide* for details about using this view and about the following:

- Need for custom Lon Xml files
- Lon Xml file overview
- Lon Xml creation
- Storing .Inml files
- Differential temperatures and Inml file edits

## Manage Credentials

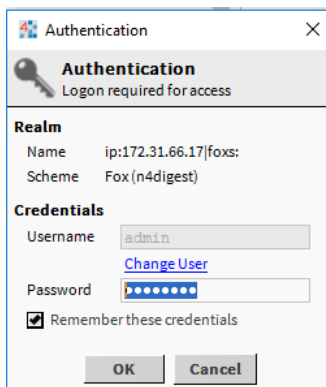
The Manage Credentials tool, or Credentials Manager provides a window to access and open any previously remembered (cached) connections from your Workbench, including both platform and station connections.

Figure 146: 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* (checkbox) enabled.

Figure 147: Authentication (Remember credentials)

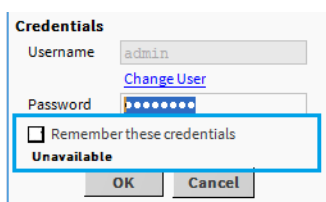


The available caching of credentials is a convenience feature, such that you can simply open the platform or station later by entering only the IP address, or simply clicking on that host's dimmed platform or station in the Nav tree, or going to the credentials manager. The login authentication window 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 checkbox 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.

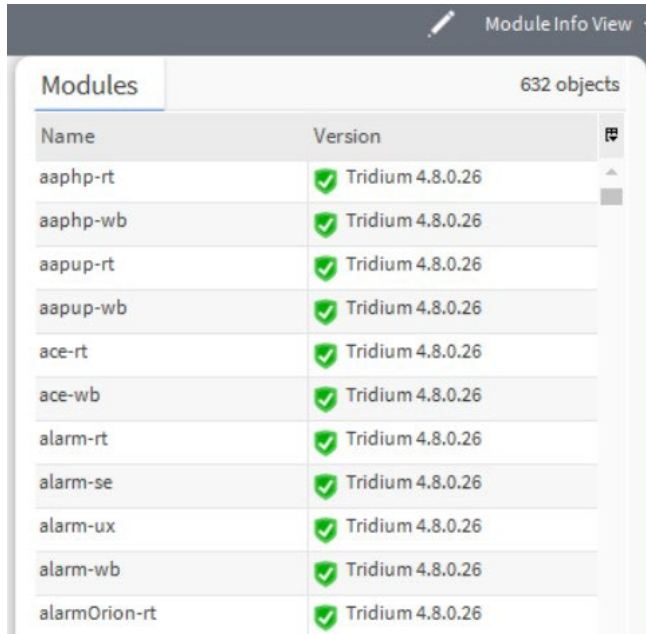
Figure 148: Credentials checkbox



## Module Info View

Available in Niagara 4.8 and later, 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.

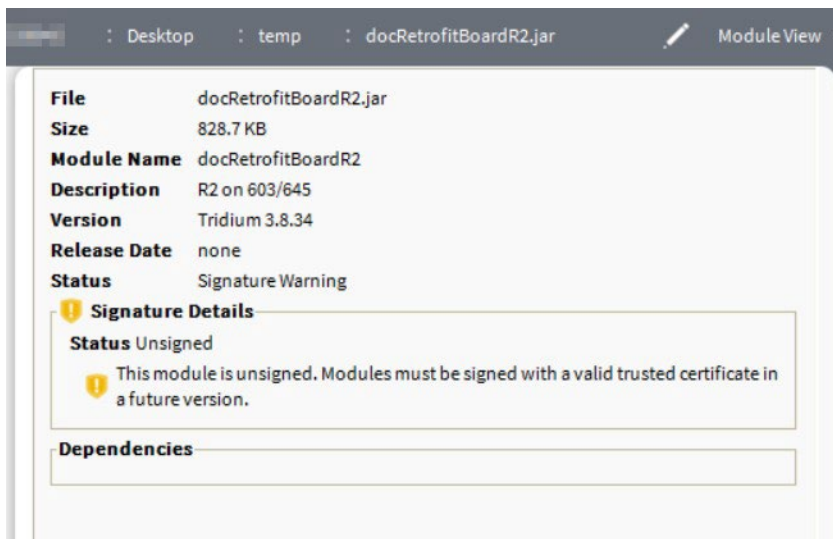
Figure 149: Module Info View shows signature status for modules installed on local machine



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 150: Module View showing details for a module that is not yet installed



File: docRetrofitBoardR2.jar  
 Size: 828.7 KB  
 Module Name: docRetrofitBoardR2  
 Description: R2 on 603/645  
 Version: Tridium 3.8.34  
 Release Date: none  
 Status: Signature Warning

**Signature Details**  
 Status: Unsigned  
 This module is unsigned. Modules must be signed with a valid trusted certificate in a future version.

**Dependencies**

## NDIO to NRIO Conversion Tool

In Niagara 4.2 and later, 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 which only supports 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 Niagara 4.3 and later 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.

Note that in prior releases (Niagara 4.2/AX-3.8U2 and earlier), 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

## 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 which does not support NDIO.

Prerequisites:

- 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.

NOTE: 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.

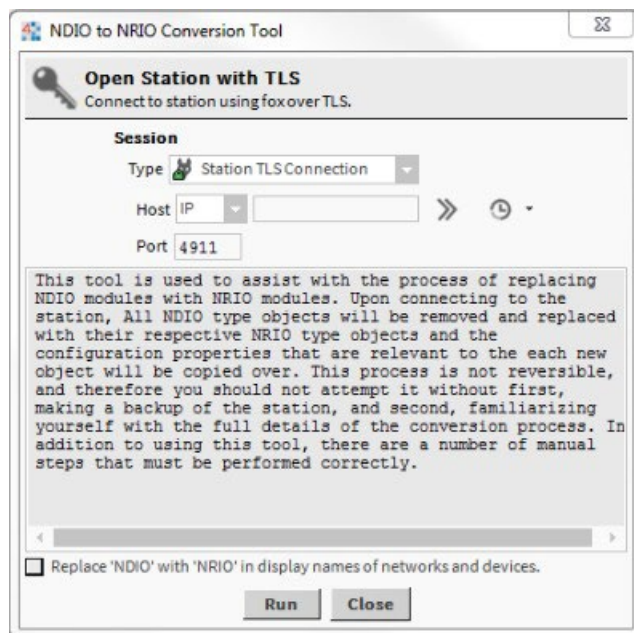
Step 1 Backup the JACE station that contains the NDIO module(s) to be converted.

Step 2 Confirm that both the NRIO and NDIO modules are installed on the JACE

Step 3 Click Tools→ NDIO to NRIO Conversion Tool.

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

Figure 151: NDIO to NRIO Conversion Tool window



Step 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.

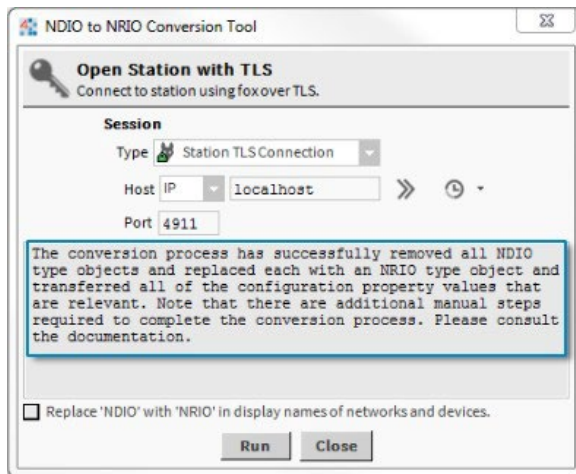
Step 5 Click Run.

- 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.

Figure 152: Conversion successful



Step 6 Click Close.

Step 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.

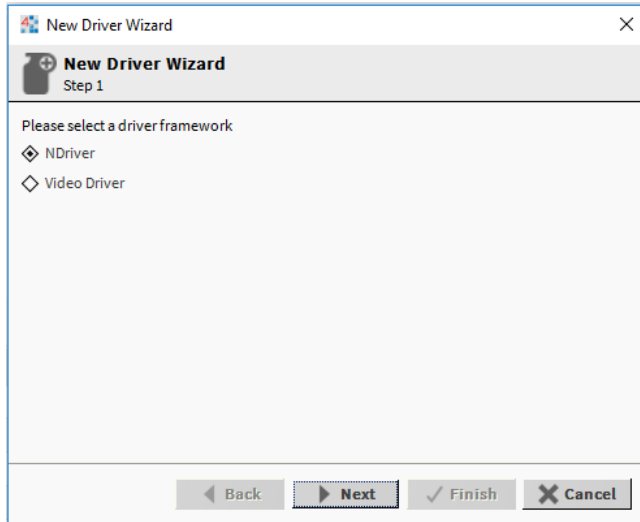
Step 8 Open the NRIO Device Manager for one of the new NRIO Networks. Use the Discover and Match functions to match the converted NRIO devices to the connected ones. Repeat this step for each network.

The conversion process is now complete.

## 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.

Figure 153: New Driver Wizard

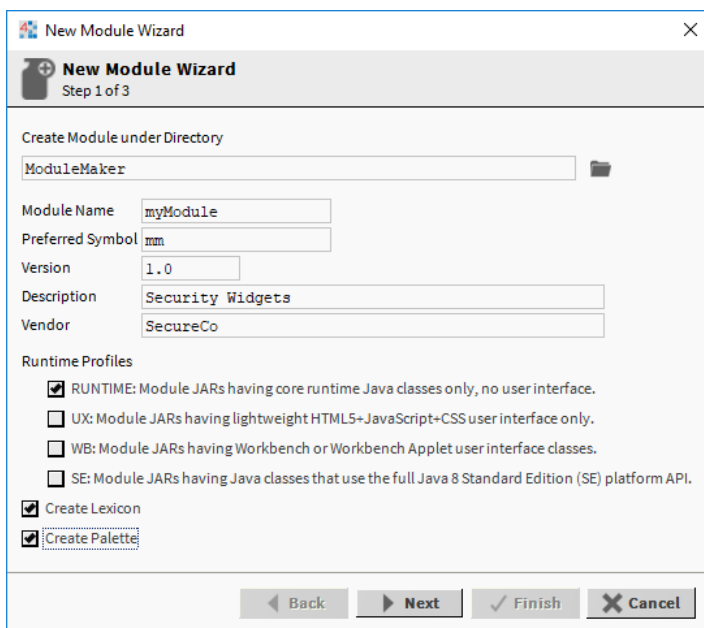


This wizard is provided as a convenient way to perform some of the steps necessary for creating a driver module. The four wizard windows allow driver developers to specify some basic driver module options and finish with the wizard creating a set of folders and files under a station directory that you assign during the wizard process. For details, see the Niagara Developer Guide.

## New Module Wizard

The New Module Wizard is available when you select Tools→ New Module. The wizard presents a series of windows that help you create and save a new module.

Figure 154: 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.

## New Station wizard

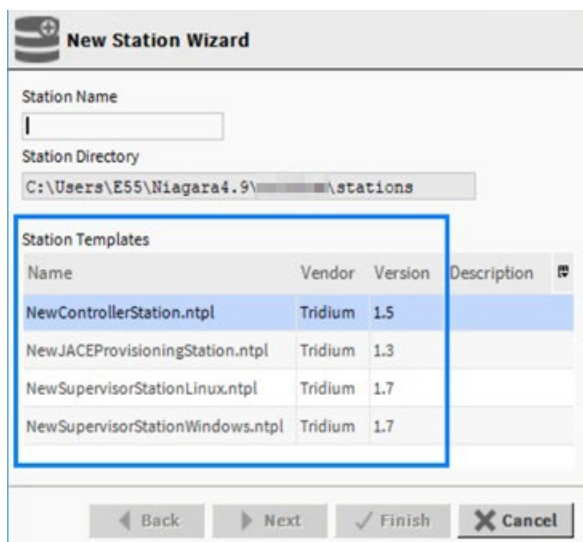
The New Station Wizard available in the Tools menu, uses templates to define the starting contents of a new station.

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.

NOTE: In the Niagara 4.6 Workbench (and later), 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.

NOTE: 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.

Figure 155: New Station Wizard



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 Niagara 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 (your Niagara User Home) and not in the User Home of the local Niagara 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 Niagara 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 156: 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.

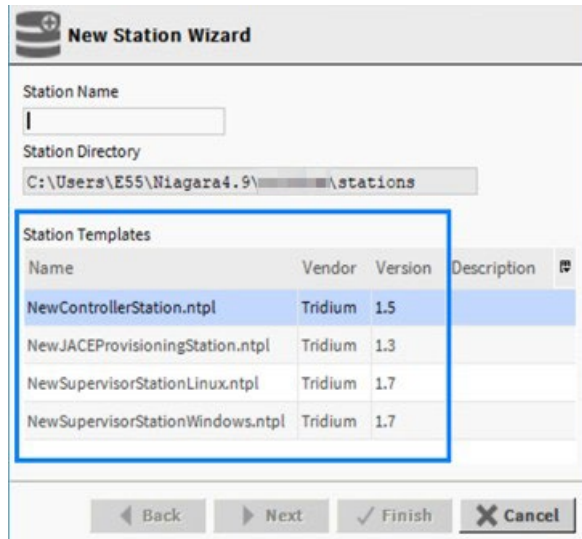
## Creating a new station

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

NOTE: If the platform that you are running is licensed for FIPS, then the New Station tool will create a FIPS-compliant station.

Step 1 In Workbench, select Tools→New Station. The New Station Wizard opens.

Figure 157: New Station Wizard



Step 2 In the New Station Wizard do the following:

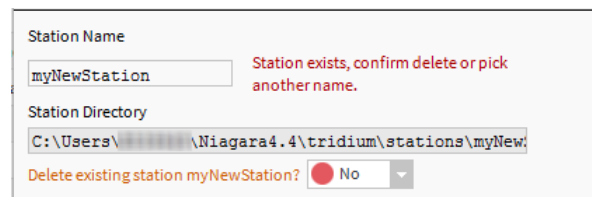
a. Enter the **Station Name**

The station name field is case-sensitive and must begin with a letter. Best practice is to keep the station name short, use a station display name, if a longer name with spaces or other characters is required.

NOTE: The **Station Name** text that you enter is automatically appended to the read-only **Station Directory** field to create a directory of the same name.

Also, if you enter a duplicate station name you are prompted about whether to delete the existing station, as shown below. You can either delete the existing station or enter a different station name.

Figure 158: Station name



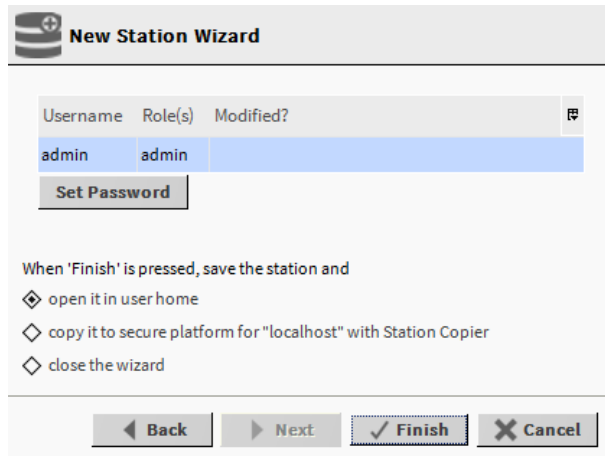
b. Click to select a **Station Template** type.

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

c. Click Next.

The second window opens to set the admin user password, and allows you to choose an action to take once the station creation is completed.

Figure 159: Station name: second window



Step 3 Click Set Password button to enter a new password in the fields.

The Set Password opens.

Step 4 Enter the new password in the fields and confirm.

NOTE: Your password must contain the following: at least 10 characters, at least 1 digit, at least 1 lowercase character, and at least 1 uppercase character.

If the new station template selected in Step 1 contains any exposed configurable properties, such as the ports, etc., this dialog will present those properties allowing you to modify the values.

Step 5 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 PropertySheet view of the new station `config.bog` file displays.

### Using the NewSupervisorStationLinux template

Niagara 4.2 and later provide a Linux Supervisor station template. This procedure describes the steps to create a new Linux Supervisor station.

Prerequisites:

- Niagara 4.2 or later installed
- The resulting station's Web Service is configured for HTTP on port 8080 and HTTPS on port 8443. You will need to add a rule in your network firewall to allow data transmission on these ports.

Step 1 In Workbench, click Tools→New Station.

Step 2 In the New Station wizard, click the NewSupervisorStationLinux.ntpl template and click OK.

Figure 160: NewSupervisorStationLinux template

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

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

Step 3 In the new station's Services node, expand WebServices and confirm configuration of the following port settings:

- HTTP = 8080
- HTTPS = 8443

Step 4 In the system 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 will make it appear to a remote client that the web server is running on port 80, but all requests sent to port 80 are redirected to port 8080 where the Niagara Web Server is listening for them. Similarly, requests sent to port 443 are redirected to port 8443.

## 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.

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

Step 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.

Figure 161: Template View

The screenshot shows the 'Template View' interface 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:

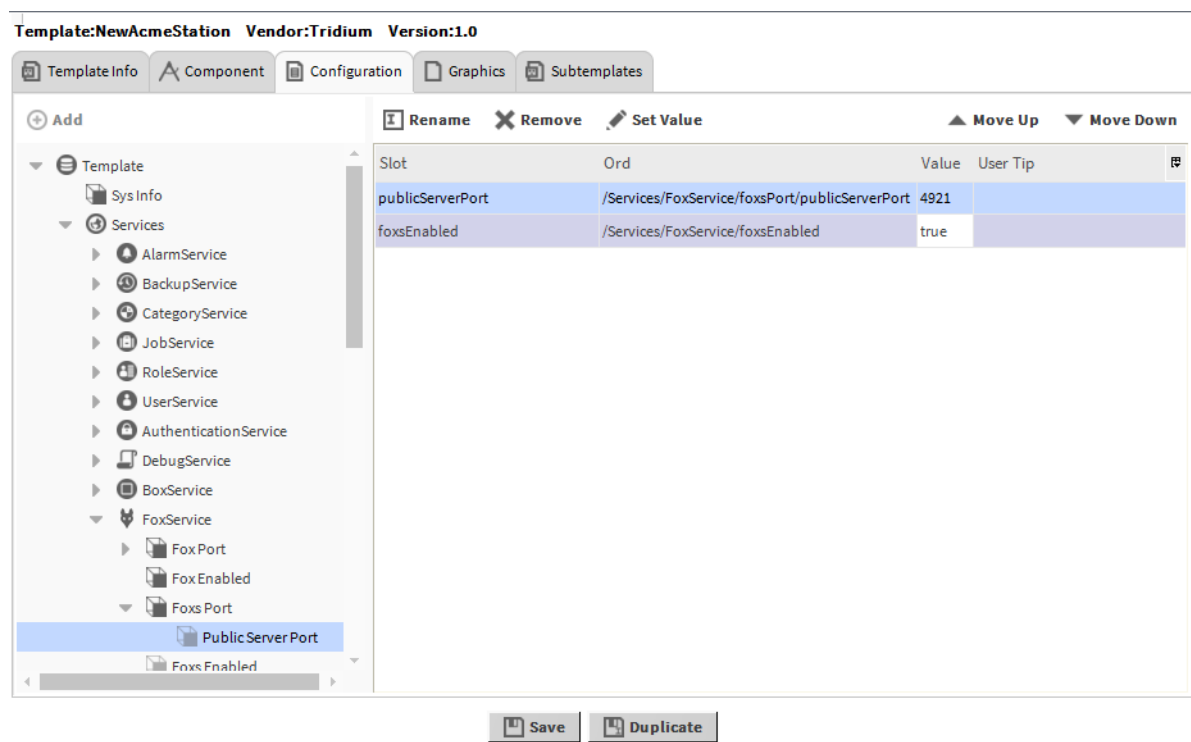
- Filename:** NewAcmeStation
- Title:** NewAcmeStation
- Vendor:** Tridium
- Version:** 1.0
- State:** Draft (dropdown menu)
- Description:** Acme new station template
- Info:** Use this template to create generic Acme station
- Icon:** NO ICON SELECTED

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

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

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

Figure 162: Configuration tab



a. In the left pane of the Configuration tab, expand Services→FoxService→Foxxs 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 **Foxxs Port**.

Step 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.

Step 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 in the next image.

Figure 163: New Station Wizard

**New Station Wizard**

Station Name

Station Directory

Station Templates

Name	Vendor	Version	Description
NewControllerStation.ntpl	Tridium	1.2	
NewJACEProvisioningStation.ntpl	Tridium	1.0	
NewSupervisorStationLinux.ntpl	Tridium	1.4	
NewSupervisorStationWindows.ntpl	Tridium	1.4	

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*.

## Request License

The Request License option on the Tools menu simply 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.

Figure 164: License request form in browser

**niagara**  
licensing

Request/Bind License

License Details

Host Id\* :

License Key\* :

Requester Details

Name\* :

Company\* :

E-mail\* :

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 Niagara. In that case, you could substitute (type in) the host ID for the other PC in this form, along with other pertinent information.

## 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.

Figure 165: Resource Estimator view

Additional Multiplier	Medium (30%)				
Device Networks	<input type="text" value="0"/>	x100,000 + Multiplier		0.000 kRU	
Devices	<input type="text" value="0"/>	x5000 + Multiplier		0.000 kRU	
Proxy Points	<input type="text" value="0"/>	x250 + Multiplier		0.000 kRU	
Program Components	<input type="text" value="0"/>	x3000 + Multiplier		0.000 kRU	
Histories	Count * (250 + Size/10)				
Config. 1	Count <input type="text" value="0"/>	Capacity	Record Count <input type="text" value="0"/>	Size <input type="text" value="0"/>	0.000 kRU
Config. 2	Count <input type="text" value="0"/>	Capacity	Record Count <input type="text" value="0"/>	Size <input type="text" value="0"/>	0.000 kRU
Config. 3	Count <input type="text" value="0"/>	Capacity	Record Count <input type="text" value="0"/>	Size <input type="text" value="0"/>	0.000 kRU
Config. 4	Count <input type="text" value="0"/>	Capacity	Record Count <input type="text" value="0"/>	Size <input type="text" value="0"/>	0.000 kRU
Config. 5	Count <input type="text" value="0"/>	Capacity	Record Count <input type="text" value="0"/>	Size <input type="text" value="0"/>	0.000 kRU
AlarmDb Capacity	<input type="text" value="0"/>	x1			0.000 kRU
<b>Total</b>					<b>0.000 kRU</b>

## Time Zone Database Tool

The Time Zone Database Tool, available in the Tools menu, provides several ways to explore the local timezones.jar on the Workbench host. This jar file is the historical time zone database.

If the Workbench host is running a station (e.g. is a Supervisor), this is also the time zone database used by that station. Using 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.

NOTE: For additional time zone details, see the *Niagara Platform Guide*.

Figure 166: Time Zone Database Tool

Time Zone Database Tool

Manifest Information | Time Zone Database | Daylight Saving Time Calculator | Time Zone Calculator

Version 2018e

Zones Provided

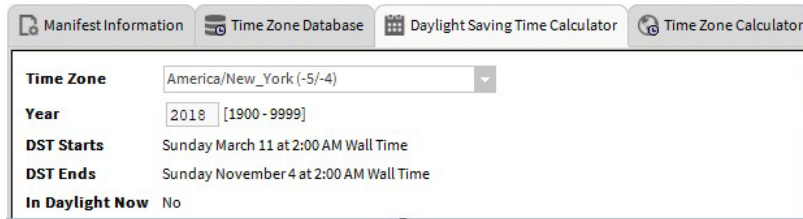
Africa | **Americas** | Antarctica | Asia | Atlantic | Australia | Canada | Europe | Etc | Indian | Other | Pacific | US

- America/Adak
- America/Anchorage
- America/Anguilla
- America/Antigua
- America/Araguaina
- America/Argentina/Buenos\_Aires
- America/Argentina/Catamarca
- America/Argentina/ComodRivadavia
- America/Argentina/Cordoba

## Daylight Savings Time Calculator

This tab in the Time Zone Database Tool shows when the DST changeover occurs for any year.

Figure 167: Daylight Savings Time Calculator tab

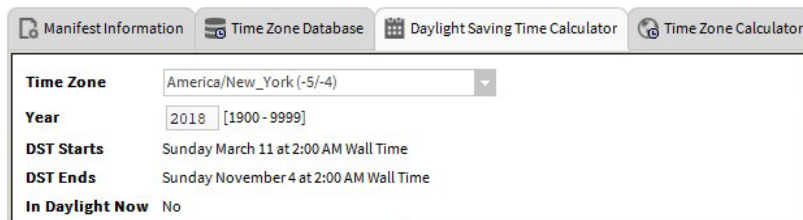


You can select a time zone and type in a year in the Year field. It will update the DST Starts and DST Ends time fields.

## Time Zone Calculator

This tab in the Time Zone Database Tool lets you compare the time between any two time zones.

Figure 168: Time Zone Calculator tab

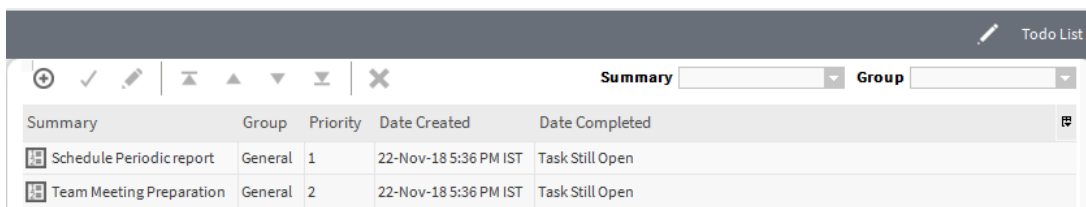


To use, select a source time zone and a target time zone, and specify a source time (initial default is the current time).

## 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.

Figure 169: Todo List view



Summary	Group	Priority	Date Created	Date Completed
Schedule Periodic report	General	1	22-Nov-18 5:36 PM IST	Task Still Open
Team Meeting Preparation	General	2	22-Nov-18 5:36 PM IST	Task Still Open

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.

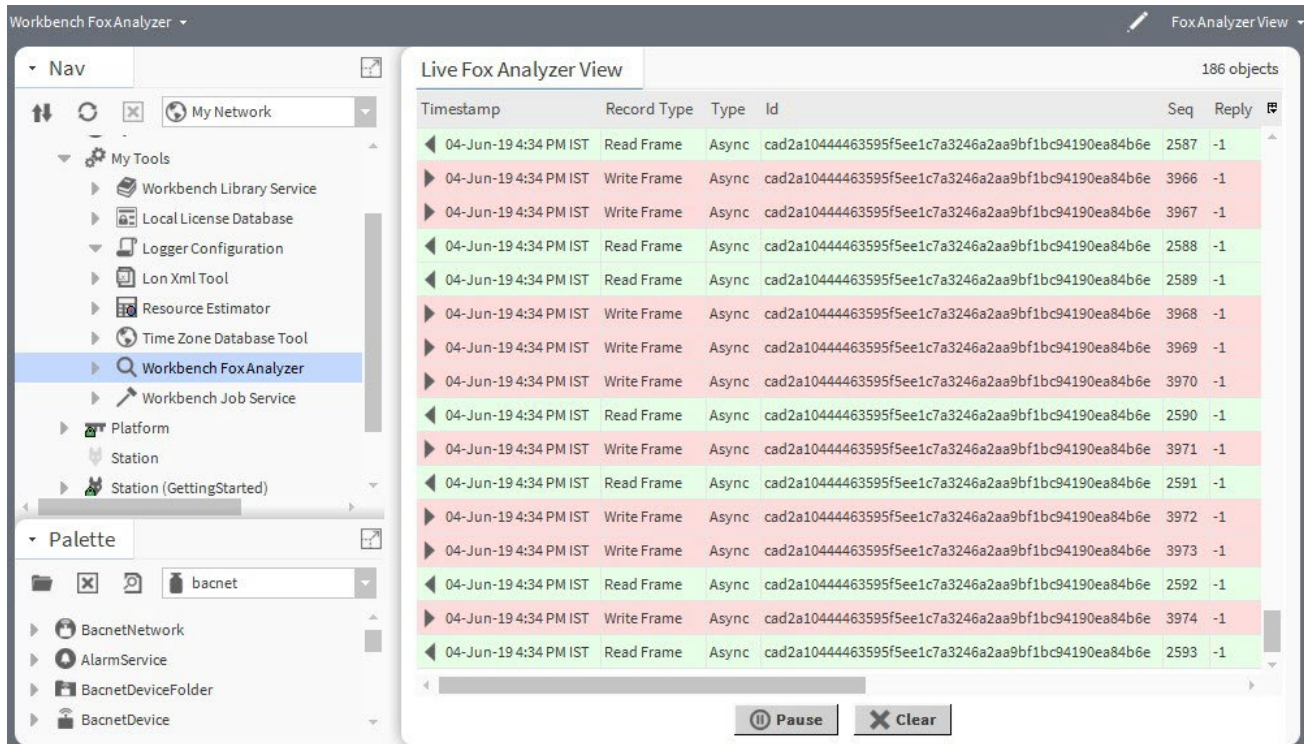


## Workbench Fox Analyzer

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.

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 section “Fox Station Analyzer”.

Figure 170: Fox Analyzer View

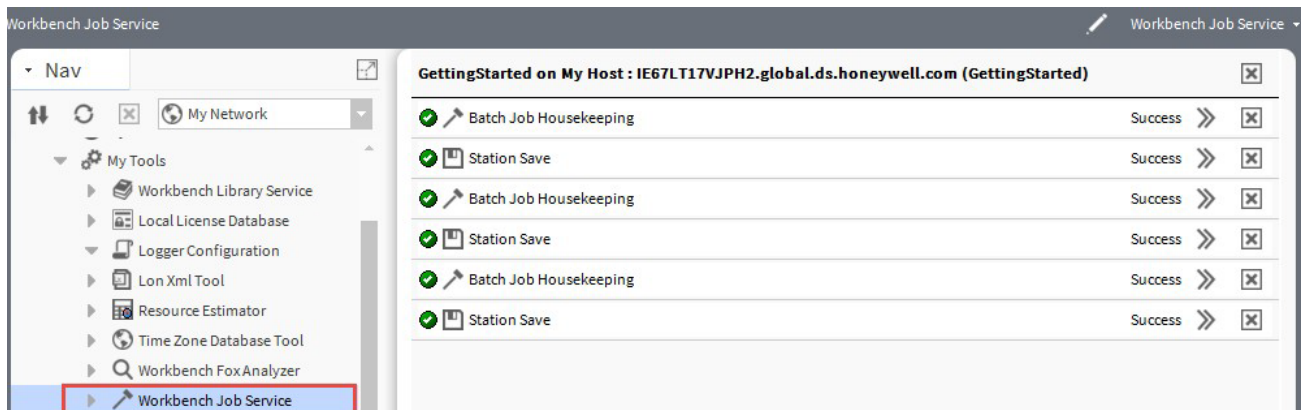


Option Name	Value	Description
View Log Limit	read-only	Displays Fox events in a color-coded table. The number indicates to the maximum number of entries displayed in the table. If the limit is exceeded, then the oldest entries are removed from the table.
Station Name Pattern	read-only	This property is used to filter 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.

## 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.

Figure 171: 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 side bar).

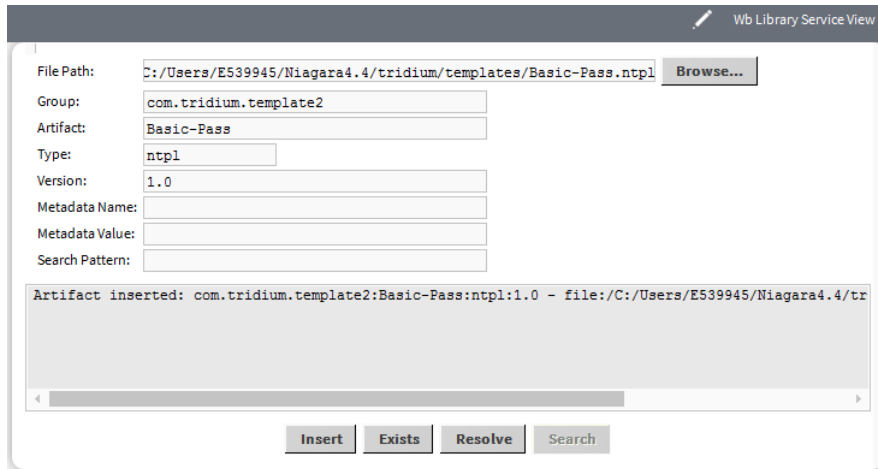
## Workbench Library Service

In Niagara 4.4 and later, 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 172: Wb Library Service View shows



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` file-path, 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.

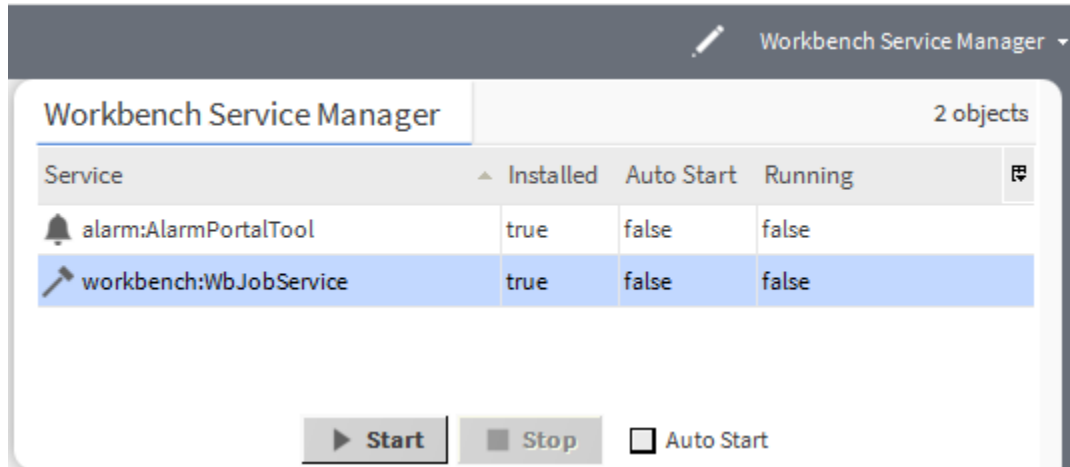
Entering a Group value activates the following buttons (at bottom):

- 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)

## 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.

Figure 173: 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.

# **CHAPTER 5 COMPONENT GUIDES**

## **Topics covered in this chapter**

- ◆ Components in alarmRdb module
- ◆ Components in backup module
- ◆ Components in baja module
- ◆ Components in chart module
- ◆ Components in control module
- ◆ Components in file module
- ◆ Components in help module
- ◆ Components in net module
- ◆ Components in program module
- ◆ Components in saml module
- ◆ Components in web module
- ◆ Components in workbench module

The Component Guides provide summary information on common 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 alarmRdb module

- rdbAlarmService

## Components in backup module

This module provides the BackupService component.

### backup-BackupService

The BackupService 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.

Also, the backup dist contains the zipped nre-config for that host (including license and certificates 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 JACE platform in zipped "dist" file format. This allows for a complete image restoration.

NOTE: 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 (AX-3.7U1), this includes unique key ring files used for client password encryption.

Not included in a backup is runtime data that is actively managed by the station, such as the alarm and history databases. This 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 Niagara release directory. If you have not previously made station backups, this directory is automatically created.

- Browser access (Wb Web Profile) — !backups

A `niagara\wbapplet\backups` folder under your Windows user profile location, e.g.:

Windows 7: `C:\Users\John\niagara\wbapplet\backups`

Windows XP: `C:\Documents and Settings\John\niagara\wbapplet\backups`

If you have not previously made station backups, this directory is automatically created.

The default name for a backup file uses a format of: `backup_stationName_YYMMDD_HHMM.dist`

For example, `backup_demo_130412_1429.dist` for a backup made of station "demo" on April 12, 2019 at 2:29 pm.

## Restoring a backup

To restore a backup dist from Workbench, you open a platform connection to the JACE, then use the platform Distribution File Installer to install a 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 (typically JACE) 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:

Property	Value	Description
Enabled	true or false	Activates ( <i>true</i> ) and deactivates ( <i>false</i> ) use of the network, device, point and component.
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.*; config.bog.b*; config_backup*
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 ExcludeFiles	string of names (each delimited by a semi-colon (;))	Specifies file types to exclude from the backup dist, when the station is stopped on the source host. either by name or extension. By default, the following files are excluded: *.lock; *backup*; console.*; config.bog.b*; config_backup*  NOTE: History (*.hdb) and alarm (*.adb) files are backed up, unlike with a running backup.
Offline ExcludeDirectories	string of Ords with Ord Chooser control	Specifies station subdirectories to exclude from the backup dist, when the station is stopped on the source host. Directories are specified using relative Ord syntax. An Ord chooser control provides a Directory Chooser dialog in which you can select station subfolders. By default, no directories are excluded, unlike with a running backup.

## baja-FoxBackupJob

This component appears as a child of the JobService and displays the following properties relative to the save job:

Property	Value	Description
Job State	read-only	States the backup job is in currently: unknown, running, canceling, canceled, success, failed.
Progress	read-only	Shows the percentage of progress toward completing the job.
Start Time	read-only	Displays the time that the job started.

Property	Value	Description
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.

## Components in baja module

- 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

## 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.

The New Station wizard installs two default authentication schemes:

- **DigestScheme** provides SCRAM-SHA256 (Salted Challenge Response Authentication Mechanism) technology for connecting Niagara 4 entities. Several messages are passed back and forth to prove the client knows the password.
- **AXDigestScheme** provides compatibility with stations running a previous software version.

Schemes available in the ldap palette include:

- **LdapScheme**
- **KerberosScheme**

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.

## baja-AuthenticationSchemeFolder

This component is a special container designed to store authentication schemes used in the station User-Service. 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.

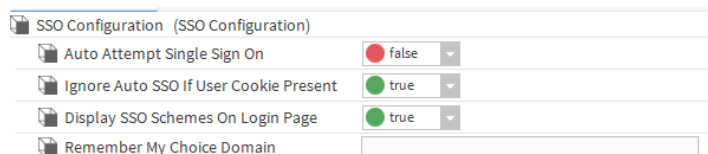
- FoxAndWebSchemes contains the DigestScheme and AXDigestScheme
- WebServicesSchemes contains the HTTPBasicScheme

## 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.

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 174: SSO properties



## SSOConfiguration properties

Name	Value	Description
Auto Attempt Single Sign On	true, false (default)	<p>When set to <code>true</code>, 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 <code>auto-SSO</code> to <code>true</code>. In order to set this to true, there must be exactly one SSO scheme available.</p> <p>When multiple SSO schemes are present in the station this setting is automatically <code>false</code> and read only.</p>
Ignore Auto SSO If User Cookie Present	true (default), false	<p>When set to <code>true</code>, 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 <code>false</code>, this has no effect.</p> <p>This is useful if you have certain users who need to login as station users rather than SSO users, such as admin users.</p>
Display SSO Schemes On LoginPage	true (default), false	<p>When set to <code>true</code>, 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.</p> <p>When using multiple SSO schemes, it is a good idea to configure the <b>Login Button Text</b> for each with a meaningful label. For example, OpenAM SSO Login.</p>
Remember My Choice Domain	text string, null (default)	<p>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.</p> <p>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.</p> <p>For example, if stations all follow the pattern <code>controller1.my-Domain.com</code>, <code>controller2.myDomain.com</code>, etc..., entering <code>my-Domain.com</code> will cause a cookie for this domain to be set on all of the stations.</p> <p>This is especially useful in an environment where <b>Auto Attempt Single Sign On</b> is set to <code>false</code>.</p>

## baja-DigestScheme

This is an authentication scheme that uses SCRAM-SHA256 (Salted Challenge Response Authentication mechanism). One of the default schemes, this component is located in the **baja** palette.

When using the DigestScheme, the password is never sent across the wire. Instead, the client sends proof that they know the password.

## baja-GlobalPasswordConfiguration

These properties configure password requirements for a particular authentication scheme. You access them by expanding **Station**→**Config**→**Services**→**Authentication**→**Authentication Schemes** and double-clicking one of the schemes.

Figure 175: Global Password Configuration properties

Global Password Configuration	
Global Password Configuration	Global Password Configuration
▼ Password Strength	Password Strength
Minimum Length	10
Minimum Lower Case	1
Minimum Upper Case	1
Minimum Digits	1
Minimum Special	0
Expiration Interval	+ 365 d 0 h 0 m 0 s
Warning Period	+ 30 d 0 h 0 m 0 s
Password History Length	0

### Scheme properties

Property	Value	Description
Password Strength	several sub-properties	See "Password strength properties," in the <i>Niagara Station Security Guide</i> .
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.

### Password strength properties

Property	Value	Description
Minimum Length	number	Indicates the total number of characters required.
Minimum Lower Case	number	Indicates the minimum number of lower case letters required.
Minimum Upper Case	number	Indicates minimum number of upper case letters required.
Minimum Digits	number	Indicates the minimum number of digits (1, 2, 3 etc.)
Minimum Special	number	Indicates the number of special characters required. For example: ! @ # \$ % ^ , . ; etc.

## baja-AXDigestScheme

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.

## 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.

This component is located in the **baja** palette.

## baja-Category

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 CategorySheet view, which shows you that component's stored category memberships (CategoryMask).

Categories reside under the station's CategoryService, which has views CategoryBrowser and CategoryManager. Categories play an integral role in station security, where you can give users permissions for some (or all) categories.

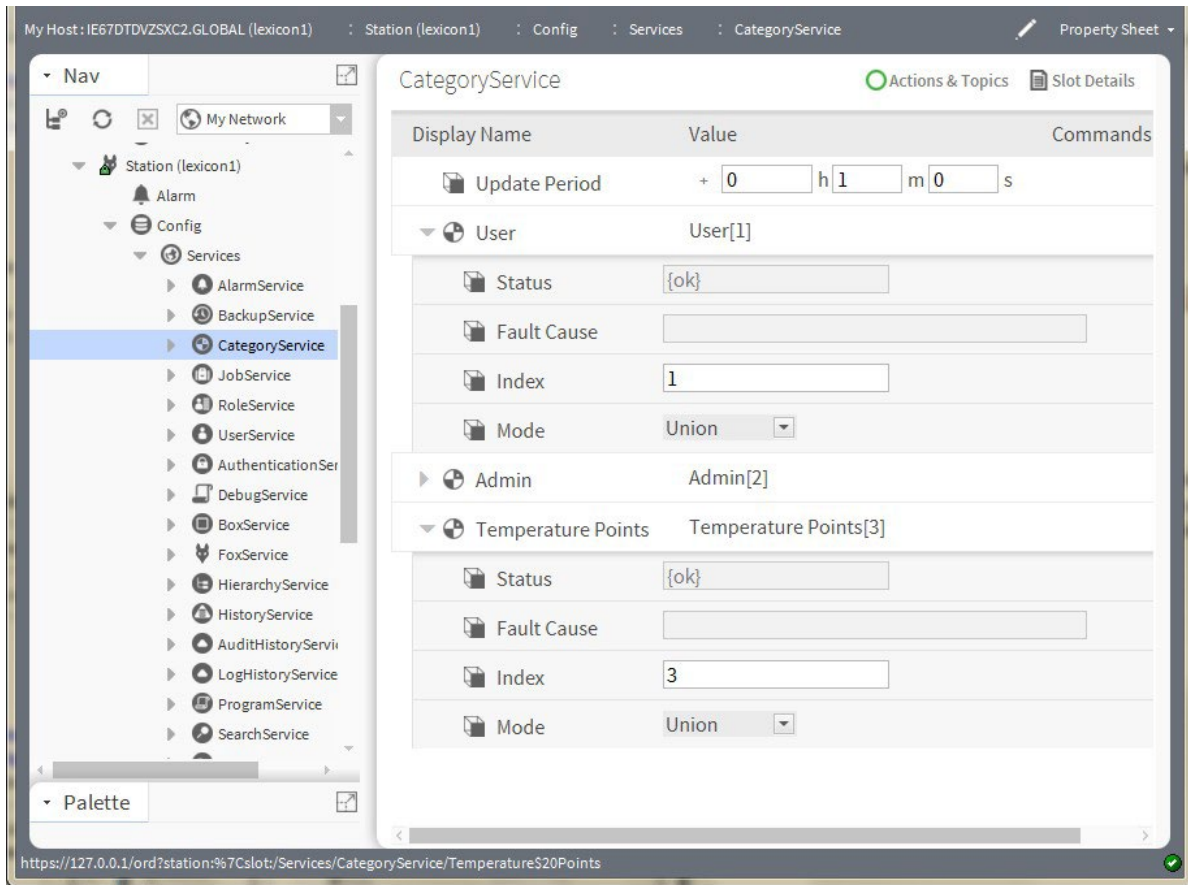
## baja-CategoryService

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.

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 176: CategoryService property sheet



In addition to being the container for child categories, the **CategoryService** has only one slot: **Update Period**.

## User, Admin and additional basic category properties

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.
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li>• {ok} indicates that the component is polling successfully.</li> <li>• {down} indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li>• {disabled} indicates that the <b>Enable</b> property is set to false.</li> <li>• fault indicates another problem.</li> </ul>
Mode	drop-down list	There are two modes: Union and Intersection.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Index	integer	Sequential number that identifies the property in the station.

## baja-Component

Component is the required base class for all Baja component classes. The Component is available in the baja module.

### Using Containers

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 create a map of name/Ref pairs as dynamic slots.

## **baja-DataFile**

DataFile represents a data file in the file system of a session.

## **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**

Folder is a special container designed to store components. The Folder is available in the baja palette.

## **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 callfoo, the order of reflection mappings 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(bajoui:dialog.error)%

## **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 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.

## baja-Job

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 StationDiscovery-Job 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.

## baja-JobService

The JobService contains Jobs that were executed by different processes in the station. Each job appears as a child component. By default, the JobService is included when you create a new station using the New Station wizard. The default view of the JobService is the Job Service Manager.

NOTE: All jobs in a station are cleared upon a station restart.

## 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.

## 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.

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*

## 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 bajai:Widgets.

## 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.

## 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.

## baja-Spy

Spy is an object wrapper for an instance of Spy, with an available SpyViewer interface to view diagnostic information about the running station.

## baja-Station

Station (Config) represents the component configuration of a station in the Baja 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.

## baja-StationSaveJob

This component appears as a child of the job service and displays the following properties relative to thesave job:

Property	Value	Description
Job State	read-only	States the backup job is in currently: unknown, running, canceling, canceled, success, failed.
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.

## baja-SyntheticModuleFile

`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.

## 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.

## baja-User

User is the component that represents a station connection, typically a specific person who needs to access the system. Users are children of the station's `UserService`. User components are also children of the `UserService`'s `UserPrototypes` container, to allow centralized user support.

## baja-UserPasswordConfiguration

`UserPasswordConfiguration` (Password Configuration) 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. For details see the *Niagara Station Security Guide*. 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.

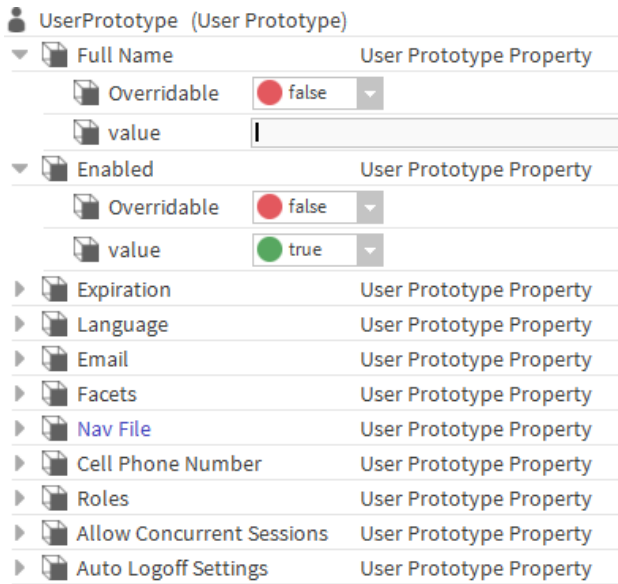
## baja-UserPrototype

The `baja-UserPrototype` (an alternative to the legacy Default Prototype) was created to provide better control over how users are created and where their properties come from. This component is used only for remote authentication schemes (e.g. LDAP, Kerberos, SAML, and etc.). It is implemented with the User Service.

NOTE: The SAML Authentication Scheme only supports the `baja-UserPrototype`. While LDAP and Kerberos support this user prototype as well the default user prototype.

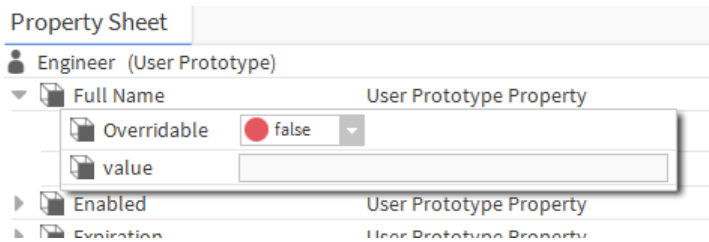
Although the properties are similar to those of the Default Prototype, `UserPrototype` has only the relevant properties on it. Also, there is an `Overridable` user property that, when selected to `true`, prevents a value from being overwritten with a default value at next login.

Figure 177: UserPrototype properties



Each of these properties has two sub-properties, as shown:

Figure 178: User properties



Name	Value	Description
Overridable	true, false (default)	Determines whether or not the property can be manually overridden on a user that was created from this prototype.
Value	string	Determines what the matching property on the user will be set to when creating or updating from a prototype.

## baja-UserPrototypes









UserPrototypes is a frozen container on a station's UserService. It contains a single frozen Default Prototypeuser, as well as any additional users needed for support of centralized users in the station's NiagaraNetwork.

## baja-UserService

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, the UserService is included when you create a new station using the New Station wizard. The UserService component is available in the baja module.

Figure 179: User Service property sheet view

UserService	
Display Name	Value
 Lock Out Enabled	<input checked="" type="checkbox"/> true
 Lock Out Period	+ <input type="text" value="0"/> h <input type="text" value="0"/> m <input type="text" value="10"/> s
 Max Bad Logins Before Lock Out	<input type="text" value="5"/> [1 - 10]
 Lock Out Window	<input type="text" value="0"/> h <input type="text" value="0"/> m <input type="text" value="30"/> s
 Default Auto Logoff Period	<input type="text" value="0"/> h <input type="text" value="15"/> m
 Guest	guest
 User Prototypes	User Prototypes
 admin	admin

### Effect of property changes on user session

Starting in Niagara 4.3, any active session associated with a user will be terminated if the following changes are made in UserService property sheet.

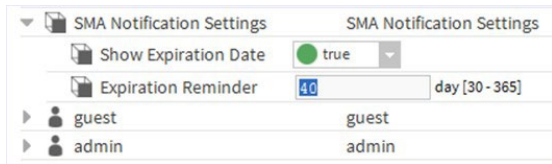
- If you remove the User from the UserService Property Sheet.
- If the **Enabled** property is set to `false`.
- If the **Expiration** property is changed to a date which has already expired.
- If the **Authentication Scheme Name** is changed.
- If the **Allow Concurrent Sessions** is set to `false`.

Property	Value	Description
Lock Out Enabled	true or false	If enabled ( <code>true</code> ), a number of consecutive authentication failures temporarily and disables login access to the user account for the duration of the lock out period (next property). Using lockout makes it difficult to automate the guessing of passwords.  NOTE: Each user has a Clear Lock Out action.
Lock Out Period	hours minutes seconds (default is 10seconds)	If lock out is enabled, this property defines the period of time a user account is locked out before being reset. While locked out, any login attempt (even a valid one) is unsuccessful.  NOTE: The default Lock Out value guards against an automated, brute-force password attack, where a computer application issues hundreds of login attempts a second. The 10 second latency thwarts such an attack, as the attacker must wait 10 seconds after each five unsuccessful login attempts. If deemed necessary, you can adjust this value to guard against human attack.
Max Bad Logins Before Lock Out	number from 1 —10 (default is 5)	If lock out is enabled in conjunction with the <b>Lock Out Window</b> , this property specifies the number of consecutive failed user login attempts that trigger a lock out after a window of time.
Lock Out Window	hours minutes seconds (default is 30seconds)	If lock out is enabled, and a user fails to log in successfully before the <b>Max Bad Logins Before Lock Out</b> window (period) expires, the user is locked out for the <b>Lock Out Period</b> duration.  The system enforces changes to lock out properties the next time the user logs in. For example, if <b>Max Bad Logins Before Lock Out</b> is set to 5, user ScottF fails to log in four times within the <b>Lock Out Window</b> , and an admin-level user changes <b>Max Bad Logins Before Lock Out</b> 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 <b>Max Bad Logins Before Lock Out</b> of 3.
Default Auto Log-off 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 <b>Use Default Auto Logoff Period</b> property is set to <code>true</code> .
SMA Notification Settings	multiple properties	See the next section.
User Prototypes	multiple properties	See the next section.

## SMA Notification Settings

These are the properties to configure the Software Maintenance Agreement (SMA) expiration reminder. For details, see the *Niagara Platform Guide*.

Figure 180: SMA expiration reminder properties



Type	Value	Description
Show ExpirationDate	true (default), false	When set to <code>true</code> , the initial SMA expiration reminder displays in the browser-connected station Login window at 45 days prior to expiry. When set to <code>false</code> , the initial SMA expiration reminder is hidden, it does not display.  NOTE: Once the SMA expires, the SMA expiration reminder cannot be hidden.
Expiration Reminder	45 day (default) [30–365 days]	By default, this is set to 45 days before expiration.

## User Prototypes

UserPrototypes is a frozen slot on a station's UserService. It contains a frozen **Default Prototype** (properties shown in the following figure) to support centralized users in the station's NiagaraNetwork, as well as a frozen Alternate Default Prototype for any additional user prototypes needed to support remote authentication schemes such as LDAP, Kerberos, and SAML.

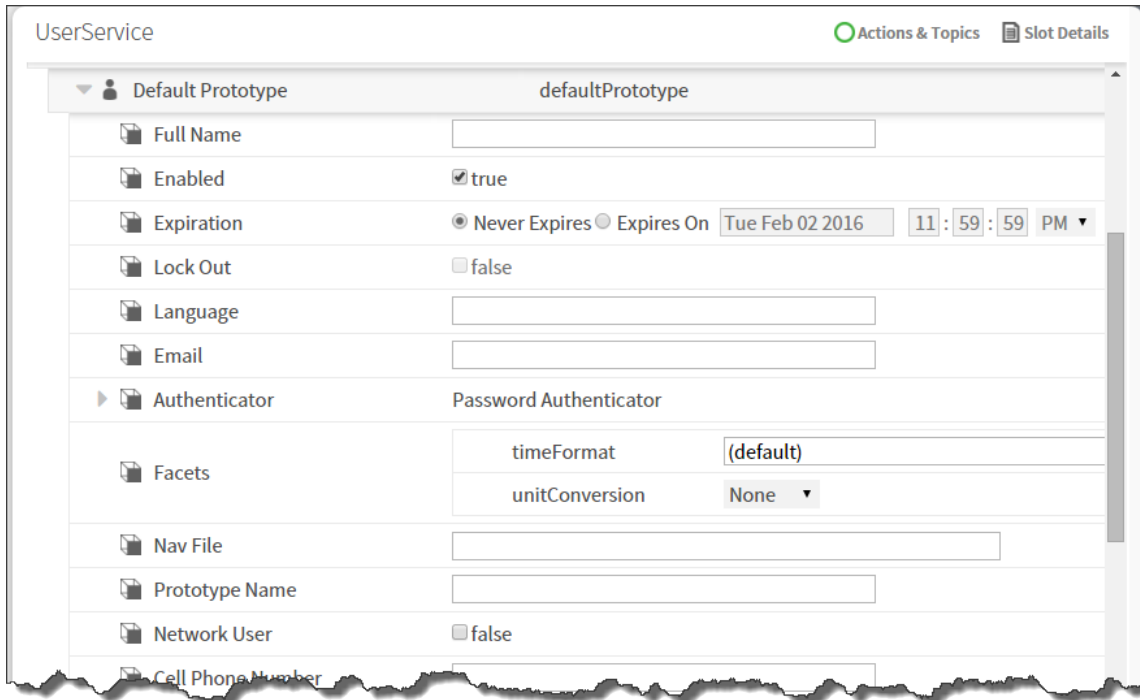
### Default Prototype

The User Service and Niagara Network still use the existing default user prototype functionality and there are no current plans to migrate them to the new UserPrototype. Also, LDAP and Kerberos will continue to support the default user prototypes.

### Alternate Default Prototype

This property allows you to specify an alternate user prototype to start with when creating a new user (instead of only using defaultPrototype). For example, you would use this to select a prototype (other than default prototype) specifically created to support a remote authentication scheme, such as SAML.

Figure 181: Default Prototype user properties



Property	Value	Description
Full Name	text	The user's name.
Enabled	true (default) or(false)	Unchecked (false) disables this user. Disabled users cannot access the system.
Expiration	radio buttons	<ul style="list-style-type: none"> <li>Never Expires permits this user to always log in.</li> <li>Expires On [date and time] allows this user to log in until the expiration date and time.</li> </ul>
Lock Out	true or false (default)	Checked enables a user to log in. Unchecked (true) prohibits this user from logging in.
Language	two lower-case letters	Defines the ISO 639 language code. For a list of codes, see the following: <a href="http://www.loc.gov/standards/iso639-2/langcodes.html">http://www.loc.gov/standards/iso639-2/langcodes.html</a> .
Email	email address	Defines the user's email address.
Authenticator	additional properties	Manages user password.
Facets	timeFormat and unitConversion	Configures the time format and units to use when this user logs in to the system.
Nav File	file:^nav/Nav-File.nav	Identifies the file to use for displaying a customized navigation tree.
Prototype Name	text	Identifies the name of the prototype used to create this user.

Property	Value	Description
Network User	true or false (default)	When true, this user account can be synchronized to other stations on the network.
Cell Phone Number	number	Defines the user's mobile phone number.
Authentication Scheme Name	drop-down list	Identifies the authentication scheme used to verify this user.
Roles	radio buttons	Identifies the user's role.
Allow Concurrent Sessions	true (default) or false	When checked, allows multiple sessions. When unchecked, a new session invalidates the old session.
Auto Logoff Settings	additional properties	Refer to the below section in this document.
Default Web Profile	additional properties	Refer to the below section in this document.
Mobile Web Profile	additional properties	Refer to the below section in this document.

## Auto Logoff Settings

Property	Value	Description
Auto Logoff Enabled	true (default) or false	<p>When <code>true</code>, a station connection (via Workbench or web browser) automatically disconnects if a period of inactivity exceeds the amount of time specified for the <b>Default Auto Logoff Period</b> (in the <code>UserService</code>).</p> <p>When <code>false</code>, the station does not automatically terminate a user's session due to inactivity.</p> <p>NOTE: Separate auto logoff options exist for Workbench which functions independently of these station settings. For details, see the <i>Getting Started with Niagara guide</i>.</p>
Use Default AutoLogoff Period	true (default) or false	<p>If the property is set to <code>true</code>, the <b>Default Auto Logoff Period</b> (configured in the <code>UserService</code>) displays the specified time.</p> <p>When <code>false</code>, the specified <b>Default Auto Logoff Period</b> is not observed. Instead, use the <b>Auto Logoff Period</b> property to set a different auto logoff time period.</p>
Auto Logoff Period	0h 15m (default) (2-4 hours)	<p>When <b>Use Default Auto Logoff Period</b> property is set to <code>false</code>, use this property to specify a different time period. The range is 2 minutes to 4 hours.</p> <p>Otherwise, this property is read only, showing the value specified in the <code>UserService</code>'s <b>Default Auto Logoff Period</b>.</p>



## Authenticator—Password Config

Property	Value	Description
Force Reset at Next Login	true or false (default)	Requests that the user create a new password the next time they log in.
Expiration	radio button	Configures a password change for a specific date and time.

## Default Web Profile

These properties configure the information available to a specific user who accesses the system through a browser. By default all information is visible.

Property	Value	Description
Type Spec (type of profile)	drop-down list	Identifies the type of profile: <ul style="list-style-type: none"> <li>hx (default)</li> <li>axvelocity</li> <li>mobile</li> <li>Workbench</li> </ul>
Type Spec (type of profile)	drop-down list	Identifies the type of profile: <ul style="list-style-type: none"> <li>BasicHxProfile</li> <li>DefaultHxProfile</li> <li>HTML5HxProfile (default)</li> <li>HandHeldHxProfile</li> </ul>
Hx Theme	drop-down list	Selects the look of the user interface: <ul style="list-style-type: none"> <li>Zebra</li> <li>Lucid</li> </ul>
Enable Hx Work- bench Views	yes (default)	Removing the check mark disables the views.
Enable Nav Tree Side Bar	yes (default)	Removing the check mark disables the Nav tree.
Enable Search Side Bar	yes (default)	Removing the check mark disables the use of the search side bar.
Enable Palette Side Bar	yes (default)	Removing the check mark disables the use of the palette side bar.
Enable Nav File Tree	yes (default)	Removing the check mark disables the Nav tree.
Enable Config Tree	yes (default)	Removing the check mark disables the Config tree.
Enable Files Tree	yes (default)	Removing the check mark disables the Files tree.
Enable Histories Tree	yes (default)	Removing the check mark disables the Histories tree.
Enable Hierarchies Tree	yes (default)	Removing the check mark disables the Hierarchies tree.

## Mobile Web Profile

Property	Value	Description
Type Spec (type of profile)	drop-down list	Identifies the type of profile: <ul style="list-style-type: none"> <li>Hx</li> <li>Mobile</li> </ul>
Type Spec (type of profile)	drop-down list	Identifies the type of profile: <ul style="list-style-type: none"> <li>BasicHxProfile</li> <li>DefaultHxProfile</li> <li>HTML5HxProfile (default)</li> <li>HandHeldHxProfile</li> </ul>
Mobile Nav File	ord	Identifies the location of the file that defines the mobile navtree for this user.
Hx Theme	drop-down list	Selects the look of the user interface: <ul style="list-style-type: none"> <li>Zebra</li> <li>Lucid</li> </ul>

### baja-UserServicePasswordConfiguration

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.

### 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.

Initial applications of virtual components are expected in various drivers for Niagara. For details, see the *Niagara Drivers Guide*.

### 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.

Other object spaces are Files and History. Initial applications of virtual gateways are expected in Niagara drivers. For details, see the *Niagara Drivers Guide*.

### 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.

TextBlock is available in the baja palette.

## **baja-ZipFile**

ZipFile represents a zip file in the file system of a session.

## **Components in chart module**

Components in chart module are:

- BarChart
- ChartCanvas
- ChartHeader
- ChartPane
- DefaultChartLegend
- LineChart

### **chart-BarChart**

One of two chart types available (the other is LineChart).

### **chart-ChartCanvas**

ChartCanvas is the canvas widget under a ChartPane.

### **chart-ChartHeader**

ChartHeader is the header widget under a ChartPane.

### **chart-ChartPane**

ChartPane is the container widget created when adding a Px chart.

### **chart-DefaultChartLegend**

DefaultChartLegend is the legend widget under a ChartPane.

### **chart-LineChart**

One of two chart types available (the other is BarChart).

## **Components in control 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**

BooleanPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt. The Boolean-Point is available in the `Points` folder of the control palette.

### **control-BooleanWritable**

BooleanWritable 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.

The BooleanWritable is available in the `Points` folder of the control palette

### **control-NumericPoint**

NumericPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt. The Numeric-Point is available in the `Points` folder of the control palette.

### **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.

The NumericWritable is available in the `Points` folder of the control palette.

### **control-EnumPoint**

EnumPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt. The

EnumPoint is available in the `Points` folder of the control palette.

### **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.

The EnumWritable is available in the `Points` folder of the control palette.

### **control-StringPoint**

StringPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt. The StringPoint is available in the `Points` folder of the control palette.

### **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.

The StringWritable is available in the `Points` folder of the control palette.

### **control-DiscreteTotalizerExt**

DiscreteTotalizerExt is a control point extension for accumulating runtime and change of state counts on binary or enum values. Two actions are available to clear (zero) accumulated totals, `ResetChangeOfStateCount` and `ResetElapsedActiveTime`.

The DiscreteTotalizerExt is available in the `Extensions` folder of the control palette.

### **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.

The NumericTotalizerExt is available in the `Extensions` folder of the control palette.

### **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**

TimeTrigger is a component that fires a topic at configured times. It is available configured as both `Interval` or `Daily` in the `Trigger` folder of the control palette.

## **Components in file 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 a 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.

## Components in help module

- BajadocOptions

### help-BajadocOptions

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`.

## Components in net 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.

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:

- Add it to your station (under the Services node, for example)
- 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.
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.



Property	Value	Description
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: <ul style="list-style-type: none"> <li>• None: no authentication is required at the proxy server.</li> <li>• Basic: basic authentication is required at the proxy server.</li> </ul>
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.

## Components in program 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 is available in the **program** palette.

#### Actions

Execute — this function takes some input, does whatever the action should (including possibly calling sub action) and returns the action's output.

### program-ProgramService

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

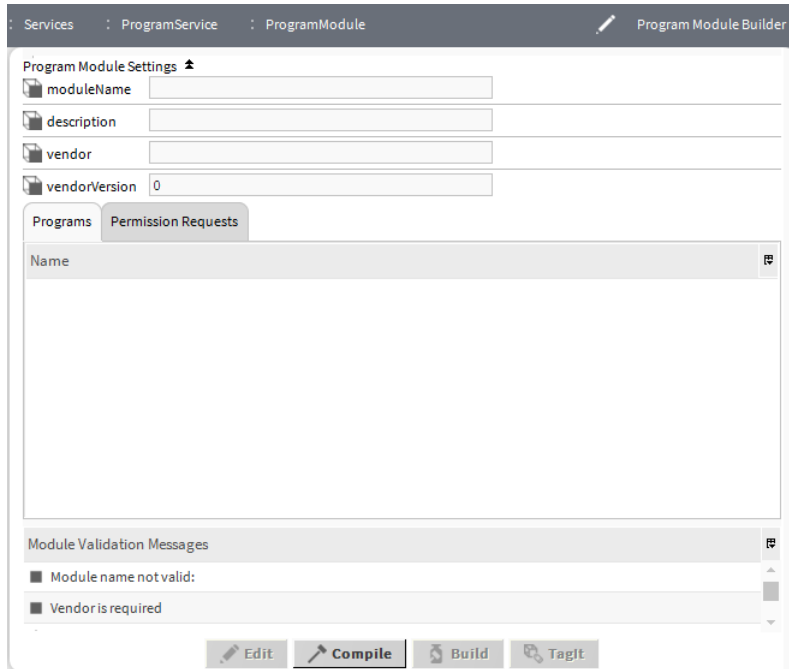
Run Robot — It is used to create robots.

## 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.

The ProgramModule is available in the **program** palette.

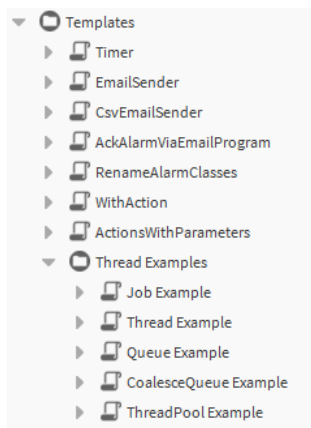
Figure 182: Program Module Builder view



## Templates

Templates folder is available in the **program** palette.

Figure 183: Templates folder



## Components in saml module

- SAMLAttributeMapper
- SAMLAuthenticationScheme

### saml-SAMLAuthenticationScheme

The SAML Authentication Scheme extends the SSO authentication scheme. SAML SSO is enabled by adding a SAML Authentication Scheme to the station. The scheme must be configured with a number of IdP configuration values, typically, these are obtained from the IdP SAML Server administrator.

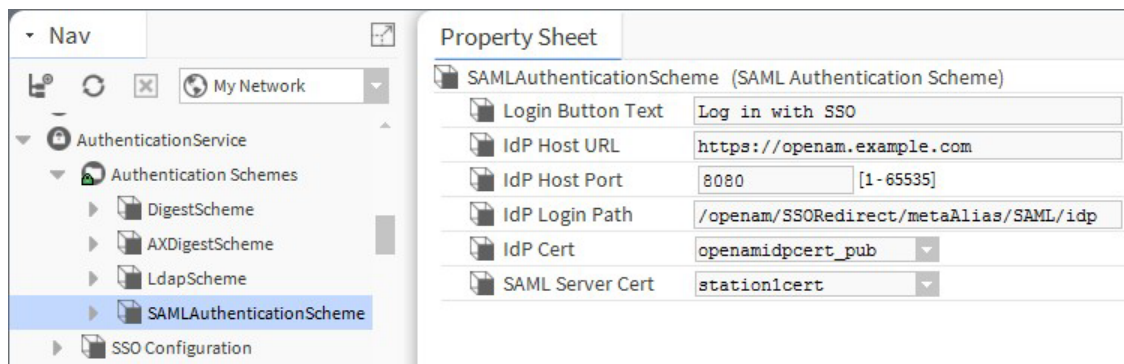
Additionally, most SAML IdPs require you to provide an XML file with metadata about the Service Provider to add it to the SAML network. In Niagara 4.8 and later, if a station is configured with a SAML Authentication Scheme, you can visit the following URL to automatically generate the station's SAML metadata XML:

<https://host.domain.com/saml/samlrp/metadata?scheme=<schemeName>> (where you replace

<schemeName> with the name of the station's SAMLAuthenticationScheme).

Since SAML is an open standard, a number of third-party SAML Servers are available (i.e. OpenAM, Salesforce, etc.). In the example shown here, the authentication scheme is configured for the OpenAm Identity Provider.

Figure 184: SAML Authentication Scheme properties



### SAML Authentication Scheme properties

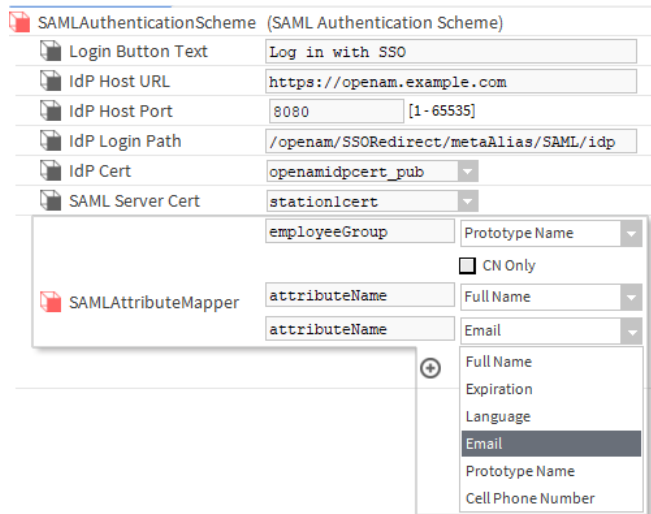
Type	Value	Description
Login Button Text	text string, "Log in with SSO" (default)	The preferred text label for the SSO login button that appears on the Login window. This button always displays if the corresponding scheme is in the authentication schemes folder.
IdP Host URL	text string, https://idp.domain.com (default)	IdP provided data, this is the URL for the host of your IdentityProvider.
IdP Host Port	443	IdP provided data, this is the port number of your IdentityProvider
IdP Host Login Path	/path/to/login	IdP provided data, this is the location on the Identity Provider that you must navigate to trigger SAML authentication.
IdP Cert	drop-down list	IdP provided data, this is the certificate required to encrypt messages sent to the IdP, and validate messages sent from the IdP.
SAML Server Cert	drop-down list	This is the certificate used by the station to sign messages that are sent back to the IdP. Note that this certificate is also provided to the IdP SAML Server admin so that the IdP can read and validate the messages. It is also used to decrypt messages sent from the IdP to the station.

## saml-SAMLAttributeMapper

During SAML Single Sign On, the SAML Identity Provider (IdP) may send the Service Provider (SP) various attributes. These may contain information about the user, and can be used by the station to build the User object. Many SAML IdPs can be configured to return the attributes with customized name. However, other IdPs may not be configurable, or IT restrictions may prevent configuring an IdP that supports this feature. In this case, you can configure the SAML Authentication Scheme to map specific SAML attributes to properties in the User Prototype.

This is done by dragging the SAMLAttributeMapper from the **saml** palette to the SAMLAuthenticationScheme.

Figure 185: Opening the SAML Attribute Mapper



NOTE: Refer to the IdP-provided documentation to determine which SAML attributes are coming in from the IdP. As an alternative, you can install a SAML add-on to your web browser which lets you view the attributes coming in from the IdP. For example, there is the SAML DevTools extension for Chrome which you can use.

NOTE: In some cases an IdP sends back multiple values for the prototypeName attribute. After the following patches if the IdP sends back multiple prototypeNames, the SAMLAuthenticationScheme considers all returned values and extracts the one that appears highest on the list of UserPrototypes (similar to how LDAP works).

- For Niagara 4.4U1:
  - **saml-rt-4.4.93.40.1**
  - **saml-wb-4.4.93.40.1**
- For Niagara 4.6:
  - **saml-rt-4.6.96.28.2**
  - **saml-wb-4.6.96.28.1**

## User properties that can be mapped from SAML attributes

The following User properties can be mapped:

- Full Name
- Expiration
- Language
- Email
- Prototype Name
- Cell PhoneNumber

All other properties will be acquired from the UserPrototype associated with the user.

## Default mappings

If no mappings are specified on the SAMLAuthenticationScheme, the following mappings will be used.

SAML Attribute Name	User Property	Extra Information
Full Name	fullName	Not applicable.
Expiration	Expiration	Format: "D-MMM-YY h:mm:ss zz"
Language	Language	Not applicable.
Email	Email	Not applicable.
Prototype Name	prototypeName	Select the CN Only checkbox if the IdP returns multiple values for user prototype.
Cell Phone Number	cellPhoneNumber	Not applicable.

## How attribute mappings are processed

Attribute mappings are processed as follows when a user logs in to the system.

1. Customized mappings are considered first. If there are multiple mappings to the same property, the first successful mapping is used. For example, if there were two mappings to the "expiration" property, and the first mapping failed to parse properly, the second mapping would be attempted. If the first mapping parsed correctly, the second would be ignored.
2. Once all customized mappings are processed, the default mappings will be attempted for any User property not yet mapped.
3. Any property not mapped from a SAML attribute will be pulled from the UserPrototype, if possible.

## saml-SAMLIdPService

In Niagara 4.9 and later, there is added support for the SAML IdP Service which allows you to take advantage of SAML functionality without having to setup an external IdP.

The `samlIDP` feature license is required to run this service.

For additional details, see “About the SAML IdP Service”.

Figure 186: SAML IdP Service properties

Status	{ok}
Fault Cause	
Enabled	<input checked="" type="checkbox"/> true
Idp Signing Cert	idpCertificate
Entity ID	https://192.0.0.3:443/saml/
Time Skew	+ 0 h 3 m 0 s
Apply Time Skew To Response	<input type="checkbox"/> false
Circle Of Trust Folder	Circle Of Trust Folder
xmlEncrypter	Saml Xml Encrypter

## Properties

In addition to the common properties (status, and fault cause), this component has the following configuration properties.

Name	Value	Description
Enabled	true (default), false	Enables/disables the service.
IdP Signing Cert	string	Server Certificate selected from the Supervisor station's UserKey Store.
EntityID	string	Supervisor station's IP address (or hostname) plus the portnumber the WebService is running on (80 for http/443 for https).  NOTE: It is necessary to append the characters "/saml/" to the EntityID value. For example, if you entered "https:// 192.68.19.20:443" as the EntityID, you then need to append it with "/saml/", so that it reads as shown here: "https://192.68.19.20:443/saml/".
Time Skew	0000h 03m 00s (default)	Sets the number of minutes to extend the validity period of the SAML request from the subordinate station. This is intended to allow SAML message to be accepted when the supervisor and subordinate stations cannot synchronize their time. Use positive values.
Apply Time Skew	true, false (default)	Select "true" to apply the specified Time Skew setting to the response. For cases where a time difference exists between the Supervisor and a remote station, this will apply the Time Skew to the response(s).

## saml-CircleOfTrust

Located in the SAMLIdPService, this component allows you to specify a collection of users that can login to a collection of stations. This is useful for managing a select set of stations and the users logging into those stations.

Configured via the Circle Of Trust Editor view, where the user is able to name the circle, provide a description, etc.

Figure 187: Circle of Trust Editor

Display Name	Value	Commands
Description	West Campus Buildings	
Http Redirect Endpoint	https://137.19.60.202:443/saml/idp/auth/httpre	
Enabled	<input checked="" type="checkbox"/> true	

Stations   Users   Auth Schemes   Prototypes

filter list

Name
<input checked="" type="checkbox"/> Building1
<input checked="" type="checkbox"/> Building2
<input type="checkbox"/> Building3

Add Station   Edit Station   Delete Station

Each circle must have a Supervisor that acts as the Identity Provider (IdP). The IdP manages only one circle. Only a Niagara station may serve as an IdP at this time.

Hardware support depends on the release version of this feature.

This component is found in the **saml** palette.

Name	Value	Description
Description	string	Description for this Circle of Trust
Http Redirect Endpoint	read-only	Shows the URL for this Circle of Trust.  NOTE: This value (as well as the IdP Host URL+ IdP Host Port)is used when configuring the <b>IdP Login Path</b> in the remotestation's <b>SAML Authentication Scheme</b> . Typically coveredby the Configure Niagara IdP and SAML Authentication Schemes provisioning job, but for stations not in the Niagara- Network the scheme must be added manually.
Enabled	true (default), false	Enables/disables the component.
Stations	additional properties	Selects the station(s) to include in this Circle of Trust. You are not limited to stations in the NiagaraNetwork. Refer to the details below on the Add Station button.
Users	additional properties	Selects user(s) from your UserService to include in this Circle of Trust. This allows specified users to log into the stations in the circle.
Auth Schemes	additional properties	Specifies which authentication schemes may be used when logging in.  NOTE: This is to accommodate users who may not yet exist in your station. For example, you might specify the LdapScheme so that LDAP users can login.
User Prototypes	additional properties	Selects one or more user prototypes in the Supervisor's User-Service that may be used when logging in to the remote station.



## Components in web module

- WebService
- ClientEnvironment

### web-WebService

This service encapsulates access to the HTTP server as well as the servlet infrastructure used to expose custom applications over HTTP. The WebService is available in the web palette. It is also one of the default services in a station created by using the New Station tool. Only one WebService is supported in a station.

Figure 188: Webservice properties

Property Sheet	
WebService (Web Service)	
Status	{ok}
Fault Cause	
Enabled	true
Http Port	82 tcp
Http Enabled	true
Https Port	443 tcp
Https Enabled	true
Https Only	false
Https Min Protocol	TLSv1.2
Cipher Suite Group	Recommended
Https Cert	tridium
Require Https For Passwords	true
Remember User Id Cookie	true
Allow Username Autocomplete	true
Login Template	<input checked="" type="checkbox"/> null
Log File Directory	file:^^webLogs
Client Environments	Client Environments
Web Launcher Config	Web Launcher Config
Cache Config	Cache Config
Warmup Config	Web Warmup Config
Hostname Redirect Settings	Hostname Redirect Settings
Http Header Providers	Http Header Providers
JettyWebServer	Jetty Web Server (started)
User Data Storage	User Data Config

Name	Value	Description
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li>• {ok} indicates that the component is polling successfully.</li> <li>• {down} indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li>• {disabled} indicates that the <b>Enabled</b> property is set to false.</li> <li>• fault indicates another problem.</li> </ul>
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.

Name	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Http Port	80 (default)	Specifies the TCP port the service listens on for HTTP client connections, where port 80 is the default.
Http Enabled	true (default) or false	Determines if HTTP client requests are processed. When set to true, turns on a standard Http connection (no communication security) using port 80. When enabled, <b>Fox Enabled</b> in the <b>FoxService</b> must also be set to true (for Web Launcher use).  When set to false, turns off the standard Http connection causing the system to ignore any attempts to connect using Http port 80. If <b>Https Only</b> is enabled, this setting (false for <b>Http Enabled</b> ) is irrelevant.
Https Port	443 (default)	Specifies the TCP port the service listens on for HTTPS (secure) client connections, where port 443 is the default.
Https Enabled	true (default) or false	Determines if HTTPS client requests are processed. When set to true, turns on secure Http communication using port 443. When enabled, <b>Foxs Enabled</b> in the <b>FoxService</b> must also be set to true (for Web Launcher use).  When set to 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	When set to true redirects any attempt to connect using a connection that is not secure (Http alone) to Https.  When set to false, does not redirect attempts to connect using Http alone.
Https Min Protocol	drop-down list TLSv1.0+ (default) TLSv1.1+ TLSv1.2	The minimum level of the TLS (Transport Layer Security) protocol to which the server will accept negotiation. The default includes versions 1.0, 1.1 and 1.2. It works with most clients, providing greater flexibility than an individual version.  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	drop-down list of server certificates; defaults to tridium	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 in to the server. If other certificates are in the host platform's key store, you can select them from the drop-down list.
Require Https For Passwords	true (default) or false	When set to true, the <b>HTTPSs Enabled</b> property also must be set to true, or the system disables the New button (for creating a new user in the UserService). This prevents the creation of a password for a new user across a connection that is not secure.  When set to false, the New button (for creating a new user in the UserService) remains enabled even if <b>HTTPSs Enabled</b> is false. This combination of settings creates a security vulnerability when creating passwords for new users and is not recommended.

Name	Value	Description
Login Template	null	When <code>Any</code> is selected, no custom login template is used. When <code>Any</code> is not selected, the option list shows available custom login templates that you can select for a station login page.
Log File directory	filepath	Default is <code>file:^^webLogs</code> . The folder in the station's file space in which log files are stored. Log file names use a <code>YYMMDD.log</code> (date) convention, such as <code>230501.log</code> for a file created May 1, 2023.
Client Environments	additional properties	This property is a container for Mobile Client Environment ( <code>mobile</code> ) entries, available if the station's host is licensed with the mobile feature. It is used in detection of a user's browsertype (e.g. desktop or mobile) and the selection of the appropriate <code>webProfile</code> for that user. See details below. Also, refer to the <i>Niagara Mobile Guide</i> .
Show Stack Trace	<code>true</code> or <code>false</code> (default)	Shows the stack trace, if set to <code>true</code> .
Load JxBrowser from Cloud	drop-down list	Loads the JxBrowser from the cloud.
Web Launcher Module Caching Type	<code>Host</code> (default) or <code>User</code>	The default option, <code>Host</code> , results in a folder and the downloading of installation modules to the <code>module</code> folder ( <code>n4applet</code> for N4, and <code>wbapplet</code> for AX). This results in the creation of multiple folders of downloaded modules, which negatively affects platform memory usage.  The <code>User</code> option results in the creation of a <code>.sharedModuleCache</code> folder. The system then downloads to a sub-folder at this location ( <code>n4applet</code> for N4, and <code>wbapplet</code> for AX). This option minimizes the memory required when running in JACE.
Web Launcher Config	additional properties	Container for several subproperties used to configure aspects of Niagara Web Launcher which provides an applet-like Workbench environment that runs completely outside of a web browser.
Cache Config	additional properties	In Niagara 4.4 and later, contains subproperties used to configure Cache Config, which caches all station home image files in the web browser. See more details below.
Hostname Redirect Settings	<code>true</code> or <code>false</code>	When set to <code>true</code> it allows host to redirect to mentioned host station in the host name field, when browsed.  When set to <code>false</code> it denies host to redirect to mentioned host station in the host name field, when browsed.  Open the browser and type local station name or IP address of the local host in the url, the host will be redirected to the station mentioned in the host name field. The station name will be displayed in the URL.
Http Header Providers	additional properties	In Niagara 4.10 and later contains HTTP Header provider component. It is used for best security purpose. See more details below.

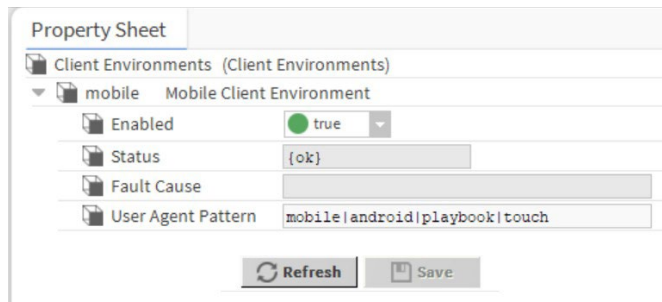
Name	Value	Description
JettyWebServer	read-only	Jetty Web Server Started. See more details below.
User Data Storage	true (default), or false	Enabled by default, this allows web apps to store user-specific data (i.e. user options for the HTML5 Alarm Console) in the userdata folder on the station file system. Typically left enabled, a user with admin privileges can set this to disabled to clear User Data Storage.

## Client Environments—Mobile

The Client Environments 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

Figure 189: Client Environments properties



Name	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li>• {ok} indicates that the component is polling successfully.</li> <li>• {down} indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li>• {disabled} indicates that the <b>Enable</b> property is set to false.</li> <li>• <i>fault</i> indicates another problem.</li> </ul>
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
User Agent Pattern	text separated by the pipe symbol ( )	A list of one or more user agents separated by the pipe symbol that identify the target display types.


## Cache Config

In Niagara 4.4 and later the Cache Config property, enabled by default, caches all station home image files in the web browser.

NOTE: When upgrading from Niagara 4.3 to Niagara 4.4, you need to clear the browser cache once manually to take advantage of this change. Additionally, when changes are made to a station home image file, clear the browser cache manually to update the cache. One exception to this is if the changed image file type is not one that is included in the Cached File Extensions property setting.

To revert back to the Niagara 4.3 behavior, set the **Enabled** property to `false`.

Figure 190: Cache Config properties

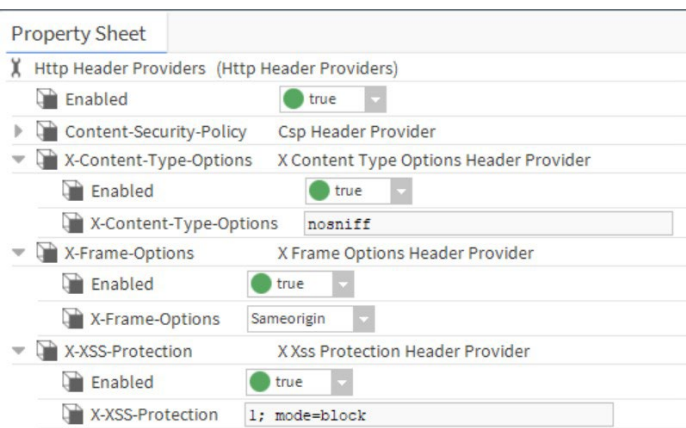


Type	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Cached File extensions	png, jpg, gif, svg (default) or *	Set the desired file type(s) to cache or set to "*" to cache all file types without re-validation.

### Http Header Providers

In Niagara 4.10 and later, there is added support for customizing HTTP headers in responses from the web server. You can access this component by double clicking the Services→WebServices→HTTP Header Providers. This component contains four different headers that you may customize as needed. Any or all of these headers may be turned off by setting their Enabled property to false, but for best security leave them all Enabled.

Figure 191: Http Header Providers properties



Name	Value	Description
Content-Security- Policy	additional properties	In response to a request resource such as Images, JavaScript and CSS the <b>Content Security Policy</b> component notifies the browser, what restrictions should be put on them. See more details below.
X-Content-Type- Options	nosniff	The <b>X-Content-Type-Options</b> component indicates to browsers that they should apply additional restrictions to auto-detection of content types in downloaded files.  nosniff blocks a request if the request destination is of type: <ul style="list-style-type: none"> <li>style and the MIME type is not text/css, or</li> <li>script and the MIME type is not a JavaScript MIME type.</li> </ul>

Name	Value	Description
X-Frame-Options	Sameorigin (default) Deny or Any (least secure)	<p>The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a &lt;frame&gt; or &lt;iframe&gt;. You can use this to avoid clickjacking attacks, by ensuring that content is not embedded into other sites.</p> <p>If you specify Sameorigin, the page will load in a frame as long as the site including it in a frame is the same as the one serving the page (same server). If a page specifies Sameorigin, browsers will forbid framing only if the top-level origin FQDN (fully-qualified-domain-name) does not exactly match FQDN of the subframe page that demanded the Sameorigin restriction. This is considered a safe practice.</p> <p>If you specify the Deny, attempts to load the page in a framewill always fail.</p> <p>NOTE: The Deny option inhibits display of some typical Shell Hx Profile views.</p> <p>NOTE: If you specify the Any option, then Cross-Frame Scripting (SFS) and Cross-Site Scriptin (XSS) are allowed.</p>
X-XSS-Protection	1; mode=block	If an attack detects, the browser prevents framing of page andEnables X-XSS-Protection filtering.

## Content-Security-Policy

The default values for this header have been customized for typical usage using the Niagara HTML5 Hx Profile.

In addition to the standard properties (Enabled and Status), these properties are unique to this component.

Figure 192: Content Security Policy properties

Property Sheet

Content-Security-Policy (Csp Header Provider)

Enabled  true

Status {ok}

Violation Text

child-src

connect-src 'self' ws://%hostname%:%port% wss://%hos

default-src 'self'

frame-src

font-src

img-src 'self' data:

manifest-src

media-src

object-src

report-uri /csp-reports

script-src 'self' 'unsafe-inline' 'unsafe-eval'

style-src 'self' 'unsafe-inline'

Additional Directives

Refresh Save

Type	Value	Description
Violation Text	text	<p>In the case of a CSP violation, the browser chooses to report a Niagara station, where it will be logged under the web.reporting.csp log.</p> <ul style="list-style-type: none"> <li>The first violation will be logged with SEVERE priority;</li> <li>Subsequent violations will be logged FINE.</li> </ul> <p>NOTE: A CSP violation should not occur during normal usage of Niagara system; if you receive one, it would indicate that your Content-Security-Policy configuration should be changed to match browser behavior.</p>
child-src	text	Defines the valid sources for web workers and nested browsing contexts loaded using elements such as <frame> or <iframe>.
connect-src	'self' ws://%host-name%:%port% wss://%hostname%:%port%	Restricts the URLs which can be loaded using script interfaces.
default-src	'self'	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	'self' data:	Specifies valid sources of images and favicons.
manifest-src	text	Specifies valid sources of application manifest files.

Type	Value	Description
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	/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	'self' 'unsafe-inline' 'unsafe-eval'	Specifies valid sources for JavaScript.
style-src	'self' 'unsafe-inline'	Specifies valid sources for stylesheets.
Additional Directives	text	

NOTE: The Security Dashboard provides information about the HTTP Headers configuration and whether there is any performance degradation. It provides notification for any non-secure headers and explains why the settings are non-secure. To secure the header's settings, set the values as described in the web-WebService HTTP Headers properties table.

## JettyWebServer

Figure 193: Jetty Web Server properties

Property Sheet	
JettyWebServer (Jetty Web Server)	
Server State	started
Min Threads	8 [8 - 30]
Max Threads	30 [14 - max]
Thread Idle Timeout	000000h 05m 00s [1 second - +inf]
N C S A Log	N C S A Request Log

Name	Value	Description
Server State	read-only	Displays the state of the server.
Min Threads	4 - 30 (defaults to 4)	Specifies the minimum number, concurrent connections(threads) that the station makes with the server.
Max Threads	10– max (defaults to 30)	Specifies the maximum number of multiple, concurrent connections (threads) that the station makes with the server.
N C S A Log	NCSA Request Log	Refer to the below section.

## NCSA Log

This is a common format of a standardized text file that web servers use to keep track of processed requests.



Figure 194: NCSA Log properties

Name	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Retain Days	7 (default)	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	list of time zones	Identifies the time zone to use for time stamps.

Name	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Retain Days	7 (default)	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	list of time zones	Identifies the time zone to use for time stamps.

## 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).

For details, see the *Niagara Mobile Guide*.

## Components in workbench 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.

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.

### 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.

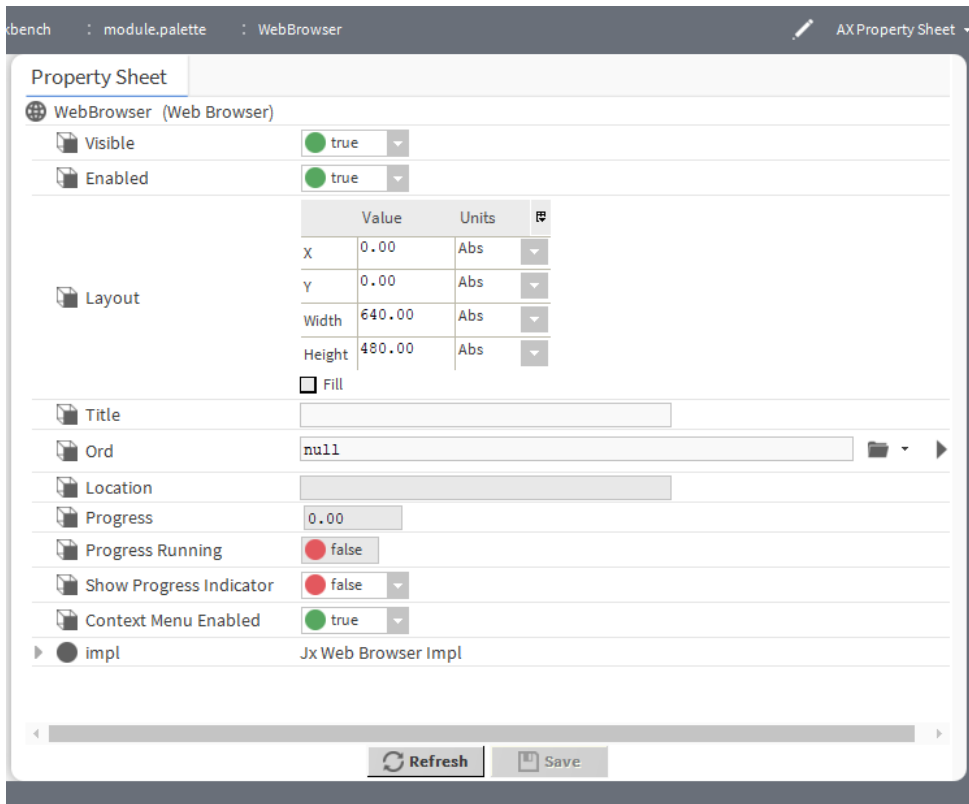
The WireSheet options allow you to view and change the following:

- Show Thumbnail
- Show Grid
- Show Status Colors

## 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.

Figure 195: Web Browser properties



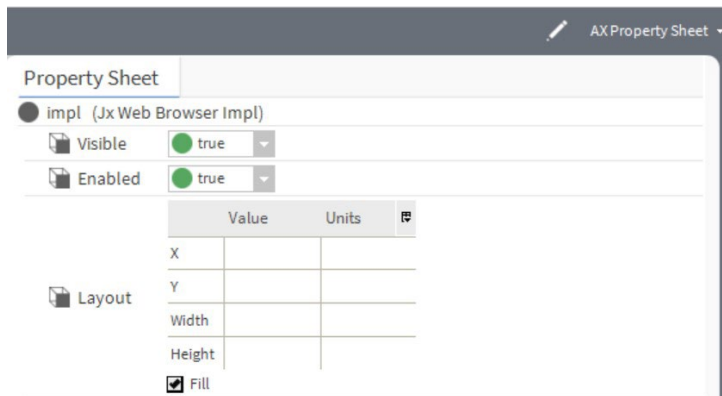
This component is located in the **workbench** palette. Double-click the WebBrowser component to open the Property Sheet view.

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 ProgressIndicator	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-JxWebBrosrerImpl.

### jxBrowser-JxWebBrosrerImpl

Figure 196: Property sheet Jx Web Browser Impl



The Impl (jxBrowser-JxWebBrosrerImpl) component is found in the **Workbench** palette, under the **Web Browser** 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 to 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 check- box turns fill on or off.

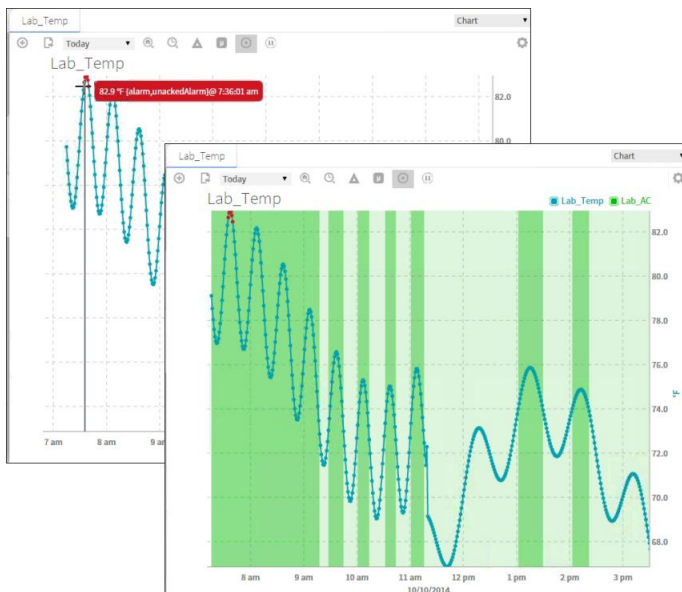
## workbench-WebWidget

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.

Examples of the baux 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 197: 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.

## Getting Started with Niagara

Figure 198: CircularGauge WebWidget



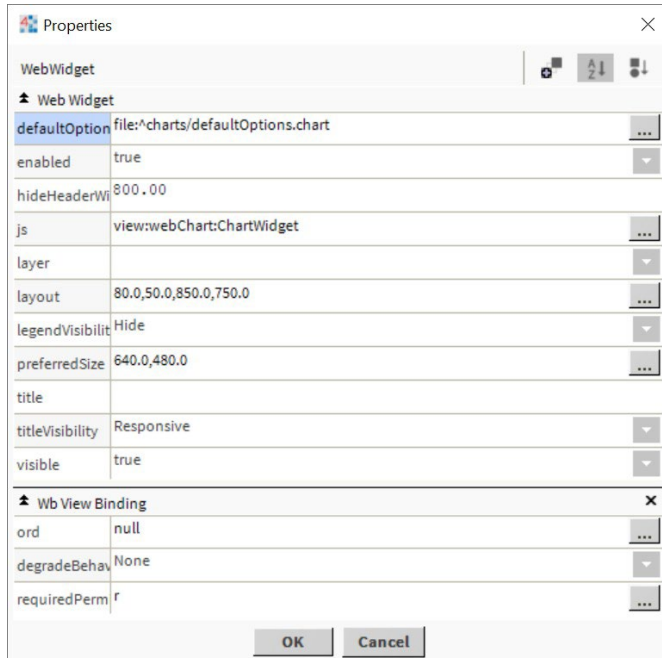
- A Dashboard may be added to any PxPage and displayed in the PxViewer. Additional WebWidgets may be added to the Dashboard pane to customize the presentation of data. The dashboard is used to write dashboard-specific data to and from a station for a specific user.

Figure 199: Dashboard WebWidget



## Configurable properties for the Chart widget

Figure 200: Chart widget properties



Property	Value	Description
defaultOption	file:^charts/defaultOptions.chart	Provides ord for the defaultOption widget. You can browse to select another widget.
hideHeaderWidth	number	All headers are visible when you set the value 800 pixel and above.
Visible	true (default), false	Enables/disables display of the widget.
Enabled	true (default), false	Enables and disables use of the component.

Property	Value	Description
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 Absolute, Percent, or Preferred units. The Fill checkbox turns fill on or off.
Js	view:webChart: ChartWidget (default)	Provides ord for the Javascript widget. You can browse to select another widget.
layer	drop-down	null
legendVisibility	drop-down	Provides legendVisibility as per set value. (for example 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)	Provides ord for bound label. You can browse to select the Ord. Also provides selectable options for Degrade Behavior (None, Disable, and Hide).

# **CHAPTER 6 PLUGIN GUIDES**

## **Topics covered in this chapter**

- ◆ Types of plugin modules
- ◆ backup-BackupManager
- ◆ chart-ResourceManager
- ◆ help-BajadocViewer
- ◆ Plugins in html module
- ◆ Plugins in ldap module
- ◆ Plugins in program module
- ◆ raster-RasterViewer
- ◆ wiresheet-WireSheet
- ◆ Plugins in wbutil module
- ◆ 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:

- 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

## **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.

The default backup destination depends on your station connection, as either:



## Getting Started with Niagara

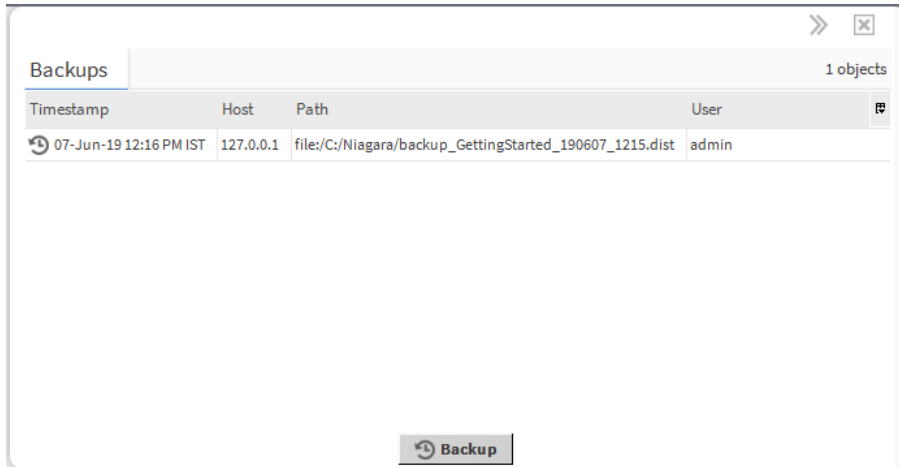
- **Workbench (Fox) — !backups:** A subdirectory `backups` under your Niagara release directory. If you have not previously made station backups, this directory is automatically created.
- **Browser access (Wb Web Profile) — !backups:** A subdirectory `backups` under the Niagara subfolder of your installed Java 2 runtime environment (Java plugin).

For example: `C:\Program Files\Java\j2re11.4.2_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 201: 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.

## chart-ResourceManager

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.

## 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.

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.

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 provide 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.

## 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 an 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 column 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	o	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.

Flag	Char	Applies	Description
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 an event source.

## Plugins in html module

This module has two plugins.

### html-WbHtmlView

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: <a href="#fragment">Title<a> and <a name="fragment">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:  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://bajauri/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 `&`
- `&apos;` - apostrophe `'`

- &copy; - copyright ©
- &gt; - greater than >
- &ldquo; - double quotation, left “
- &lsquo; - single quotation, left ‘
- &lt; - less than <
- &mdash; - em dash —
- &nbsp; - non breaking space " "
- &ndash; - en dash –
- &rdquo; - double quotation, right ”
- &rsquo; - single quotation, right ’
- &quot; - quotation mark "
- &reg; - registered trademark ®
- &trade; - 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

## Getting Started with Niagara

- 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

## html-SpyViewer

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.

## Plugins in ldap module

Plugins in ldap module are as follows:

- ldap-BasicKrb5ConfEditor
- ldap-AdvancedKrb5ConfEditor

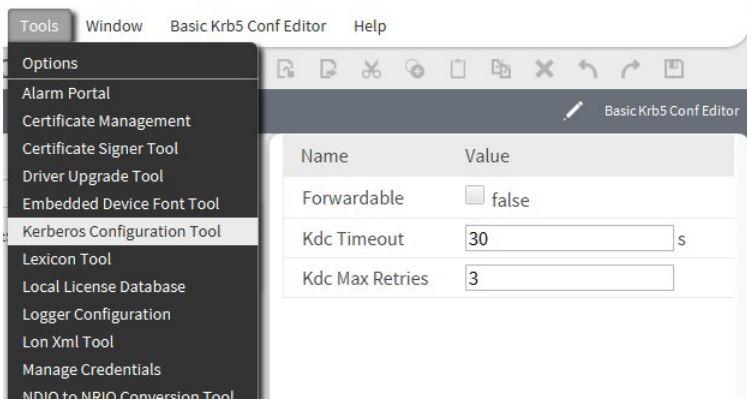
### ldap-BasicKrb5ConfEditor

In Niagara 4.3 and later, an added editor view available under the Tools menu, Basic Krb5 Conf Editor, allows you 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 202: Basic Krb5 Conf Editor view



For more details see the setup and configuration procedures for Kerberos in the *Niagara LDAP Guide*.

### Properties

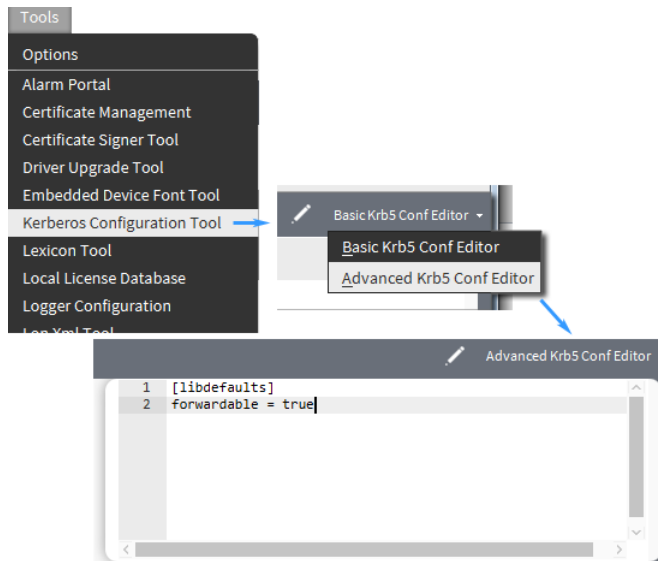
Property	Value	Description
Forwardable	true (default), false	Enables and disables forwarding of Kerberos tickets.
Kdc Timeouts	30 (default)	Required for redundant server support, specifies the length of time the station attempts to connect to the key distribution center before failing the connection 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 the cut off time should be to consider the connection 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.

### Idap-AdvancedKrb5ConfEditor

In Niagara 4.3 and later, an added editor view available in the Workbench. Located under Tools→Kerberos Configuration Tool and the views dropdown list, the Advanced Krb5 Conf Editor 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.

Figure 203: Views dropdown list



The file requires only the two lines contained in this view.

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.

## Plugins in program 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.

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.

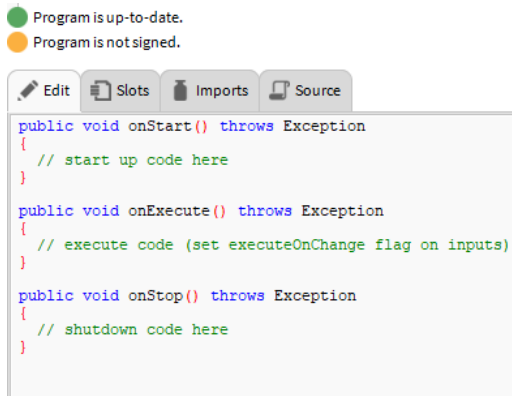
As needed, use the Clear Selected Items control to remove any items before running the operation.

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*.

### Program Editor

The Program Editor view provides the ability to view and edit Program components.

Figure 204: 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.

Tab	Description
Imports	<p>Shows the modules that have been imported. It also allows the following:</p> <ul style="list-style-type: none"> <li>• Import Type—imports a new type.</li> <li>• Import Package—imports a new package.</li> <li>• Remove—Removes added Import Type and Import Package.</li> </ul>
Source	<p>Edits the source of the program component. The editor supports special Color coding for Java Files. An example from the demo Database follows:</p> <pre data-bbox="402 436 1328 1409"> /* 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*/ import com.tridium.program.*; /* program Predefined*/  public class ProgramImpl     extends com.tridium.program.ProgramBase {     // Getters     // Setters      public BStatusNumeric getOut() { return (BStatusNumeric)get("out"); }     public BStatusNumeric getInA() { return (BStatusNumeric)get("inA"); }     public BStatusNumeric getInB() { return (BStatusNumeric)get("inB"); }      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

## program-ProgramModuleBuilder

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.

## program-RobotEditor

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

## raster-RasterViewer

The RasterViewer displays bitmapped image files: GIF, JPEG, PNG in the main window with Format and image Size at the bottom.

## 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.

### 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 thumb nail. You can also hold down the Ctrl key and move the thumb- nail 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.

## Plugins in wbutil module

### 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.

NOTE: When an admin user (user with Admin write privilege on the CategoryService and on a particular station component being adjusted) is making a category mask adjustment, the user must also have at least 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 205: 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		●							

## Columns

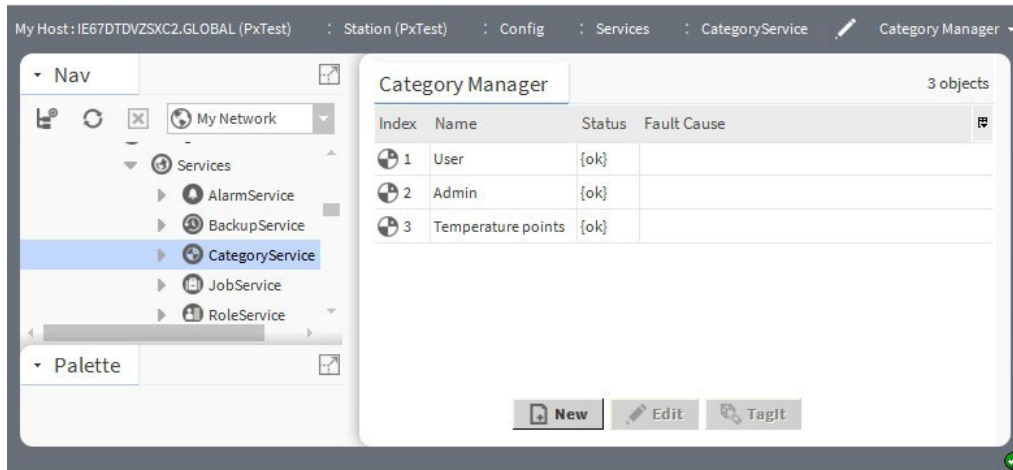
Column	Value	Description
Inherit	check mark or blank	A check mark indicates that the object inherits the category from its parent in the table.
User	Category 1	All system objects except for those listed as assigned to <code>Admin</code> are assigned to this category.
Admin	Category 2	These objects default to the <code>Admin</code> category: <ul style="list-style-type: none"> <li>The configuration services: <code>UserService</code>, <code>CategoryService</code>, and <code>ProgramService</code></li> <li>All files (the entire file space)</li> </ul>
Categories 3–8	bold bullet, grayed out bullet, or blank	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.

## 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.

You can assign an object to many categories at the same time. Each object stores its own categories.

Figure 206: Category Manager, Temperature Points as Category



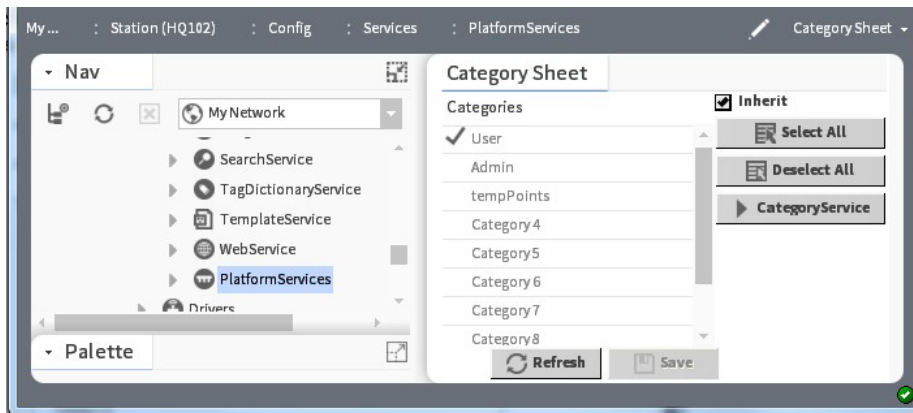
Column	Value	Description
Index	number	A unique number for the category, as it is known to the station.
Name	text string	Descriptive text that reflects the purpose of the entity or logical grouping.
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li>{ok} indicates that the component is polling successfully.</li> <li>{down} indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li>{disabled} indicates that the <b>Enable</b> property is set to false.</li> <li>fault indicates another problem.</li> </ul>
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.

## 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.

NOTE: When an admin user (user with Admin write privilege on the CategoryService and on a particular station component being adjusted) is making a category mask adjustment, the user must also have at least 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 207: Category Sheet



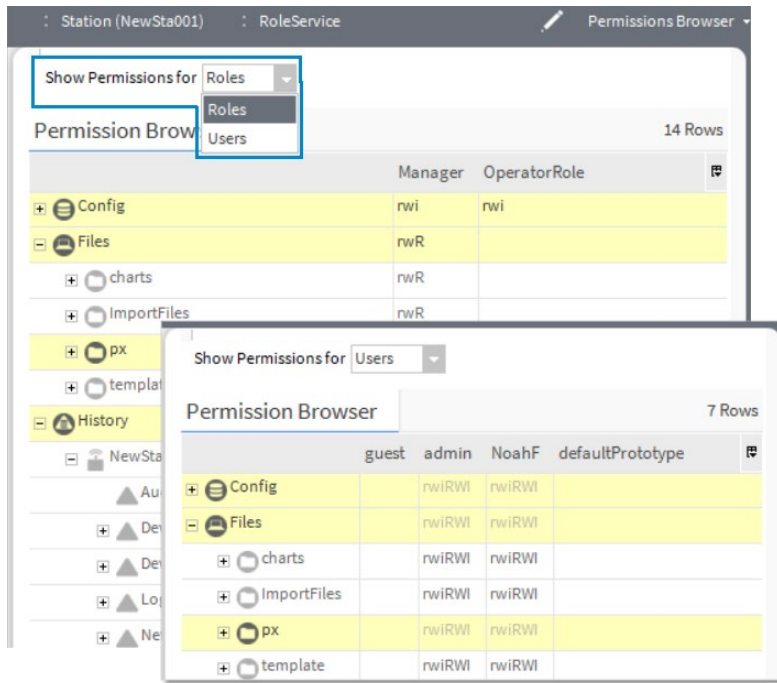
Option/button	Value	Description
Categories	text	Provides one table row for each category name.
Inherit	toggle	A check mark indicates that the component belongs to the same categories as its parent component. No check mark allows you to make explicit category assignments for this component.
Select All	button	Effective if <code>Inherit</code> is cleared, clicking this button assigns this component to all categories in this station.
Deselect All	button	Effective if <code>Inherit</code> is cleared, clicking this button removes this component from all categories.
CategoryService	button	Opens the Category Browser.
Refresh	button	Re-displays the Category Sheet.
Save	button	Records the changes made.

## 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.

NOTE: In Niagara 4.8 and later, there is added support for the `UserService` in the Permissions Browser view. Use the Show Permissions for dropdown 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 208: 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.

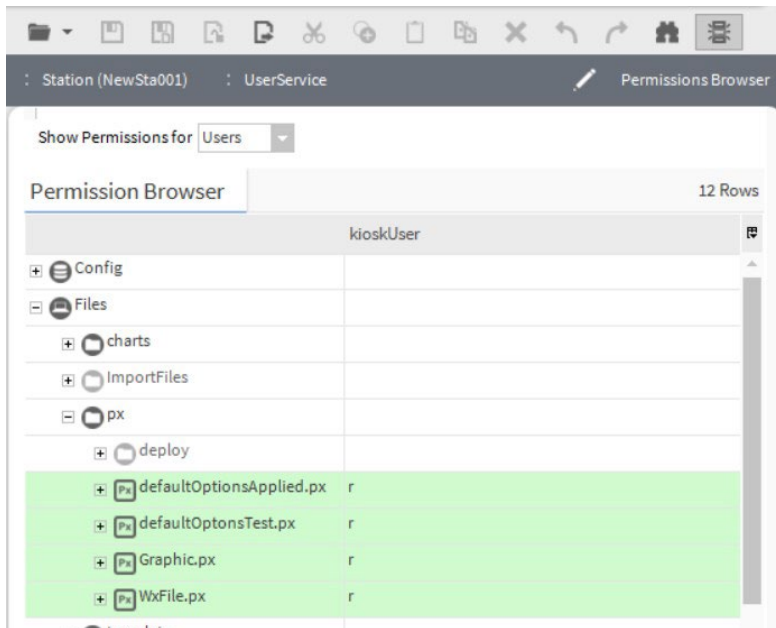
- Yellow rows are objects explicitly assigned with permissions.
- Dimmed rows represent objects that inherit their permissions from their parent object.

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 dropdown list. This allows you to globally change permission levels for any category in the station.

Additional enhancements to the Permissions Browser view include the following:

- Highlight Accessible – applies green shading to entries that are accessible by at least one displayed column (user or role). Do this by clicking the Highlight Accessible icon in the Workbench toolbar.

Figure 209: Highlight Accessible tool highlights entries accessible by this user




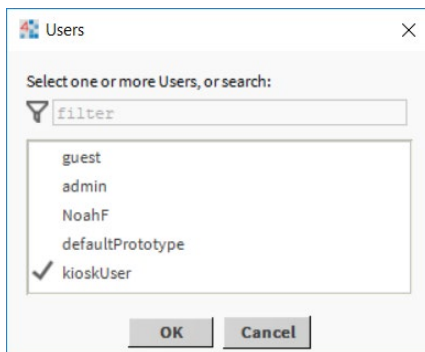
- Filtering – filters results in the table for selected users/roles. Do this by clicking the table Options icon (  ) at the far right side of the column heading row, to display the options menu, and click on Show Users... (or Show Roles...) to open a window that lists all users (or roles) which you can filter by search or selection and click OK.

Figure 210: Filtering shows permissions for selected users/roles



Column	Value	Description
First column	Nav tree for station Config, Files and History	Each Nav tree node occupies a row in the table. This expandable tree lets you navigate to objects of interest to review current permissions.
Admin	permissionsR = readW = writeI = invoke action <sub>admin</sub> level permissions appear in upper case.	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 an action for all objects.
user	permissionsr = readw = writei = invoke action <sub>operator</sub> permissions appear in lower case.	Reports the rights assigned to the <code>user</code> role. The default is no rights assigned.

### wbutil-ResourceEstimator

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.

### wbutil-ToDoList

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.

### wbutil-UserManager

The User Manager is the primary view of the UserService. You use it to add, edit, and delete users for accessing the station.

## Plugins in workbench module

This module has several plugins which provide many of the standard views.

### workbench-CollectionTable

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.

### workbench-DirectoryList

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.

## workbench-HexFileEditor

The HexFileEditor allows you to view hexadecimal files. It provides a binary view of a file's contents.

## workbench-JobServiceManager

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.

## workbench-ModuleSpaceView

The Module Space View allows you to view Modules.

## Options Button

The Options button enables and disables columns (turns them on and off).

## 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.

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.

## 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.

There are two types of Property Sheet views:

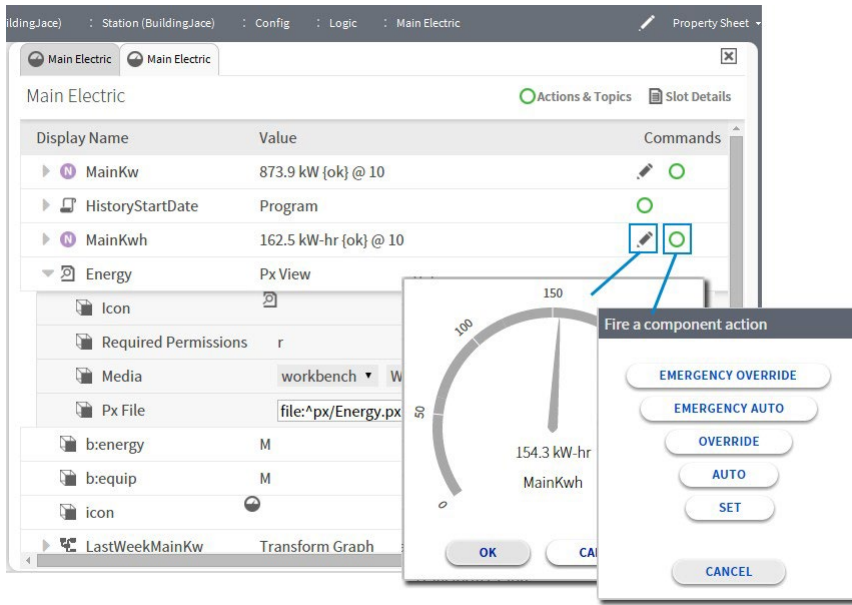
- Property Sheet view

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.



## Getting Started with Niagara

Figure 211: HTML5 property sheet view



- 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	Goes to any of the views of the selected component.
Actions	Perform any available actions on the component.
New	Create a new item of standard types.
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 clipboard and after pasting the copy, deletes the cut item.
Copy	Copies to clipboard, copied item remains
Paste	Pastes the copied item from the clipboard
Duplicate	Makes a copy in the same location as the original item
Delete	Removes the item
Find	This menu item displays the Component Finder window.
Link Mark	Sets a selected component to your popup menu, making it temporarily available for "Linking From" or "Linking To" other points.

Item	Description
Link From	Allows you to link to a selected component from another component that has been marked, using Link Mark from the popup menu.
Link To	Allows you to link 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" or "Relating To" other points.
Relate From	Allows you to add a relation from selected component to another that has been marked, using Link Mark from the popup menu.
Relate To	Allows you to add a relation to a selected component from another that has been marked, using Link Mark from the popup menu.
Rename	Allows you to rename 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	Allows you to set a display name for the selected component (as it appears in a Nav tree, and in the Wire Sheet, and Property Sheet views).
Reorder	<p>Opens the Reorder Points window, which provides the following commands for reordering points within the selected parent component.</p> <ul style="list-style-type: none"> <li>• Move Up</li> <li>• Move Down</li> <li>• Sort by Name</li> <li>• Sort by Type</li> <li>• Reset</li> </ul>
Composite	This menu item opens the Composite Editor window.
Export	Exports a selected component to the oBix .xml format.
Config Flags	Allows you to assign or remove permissions flags on a component. This is available on properties in the AX Property Sheet view. Right-click a property to see the Config Flags window.

## PropertySheet Toolbar

The Workbench toolbar contains navigation and editing buttons.

Button	Description
Back	Returns to the previous view.
Uplevel	Moves up one level in tree.
Sidebar	Shows the Sidebar options.
Recent Ords	Shows recent Ords.
Home	Moves up one level in tree.
New Folder	Creates a new directory.

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 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

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.

## workbench-ServiceManager

Service Manager allows you to view services. It is available on the ServiceContainer.

## workbench-SlotSheet

The Slot Sheet shows all the user visible slots of the Component. This includes Properties, Actions, and, Topics.

The Slot Sheet is a table that shows the following for each slot:


- Slot
- #
- Name
- Display Name
- Definition
- flags
- Type

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:

- Add Slot (Ctrl + A)
- Rename Slot (Ctrl + R)
- Config Flags
-  Config Facets
- Reorder
- Display Name

### SlotSheet Component Menu

If you right-click any Component in the Slot Sheet or the background, you can choose from the following:

- Add Slot (Ctrl + A)

- Copy (Ctrl + C)
- Delete (Delete)
- Rename Slot (Ctrl + R)
- Config Flags
- Config Facets
- Add Name Map

### SlotSheet Toolbar

The Workbench toolbar contains navigation and editing buttons. When the Slot Sheet is visible, additional toolbar buttons include:

-  (Add Slot).
-  (Rename Slot).
-  (Config Flags)
-  (Reorder)

### Add Slot

Add Slot allows you to add a slot to the Component.

### Rename Slot

Rename Slot allows you to rename a slot in the Component.

### Add Name Map

Add Name Map allows you to add a Name Map to the Component. You right-click on the property **displayNames** to delete or rename Name Map, and **displayNames\_xx** to delete or rename Name Map (xx) where xx is the Language Code.

### Config Facets

Configure Facets on the Component. This is available on Properties. Right-click a property to view the Config Facets window.

## workbench-StationSummary

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

## workbench-Synthetic Module File View

The Synthetic Module File View is the default view on a synthetic module. For details, see the *Niagara Engineering Notes*.

## workbench-TextFileEditor

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'`

## workbench-WbPxView

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.

## 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.

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.

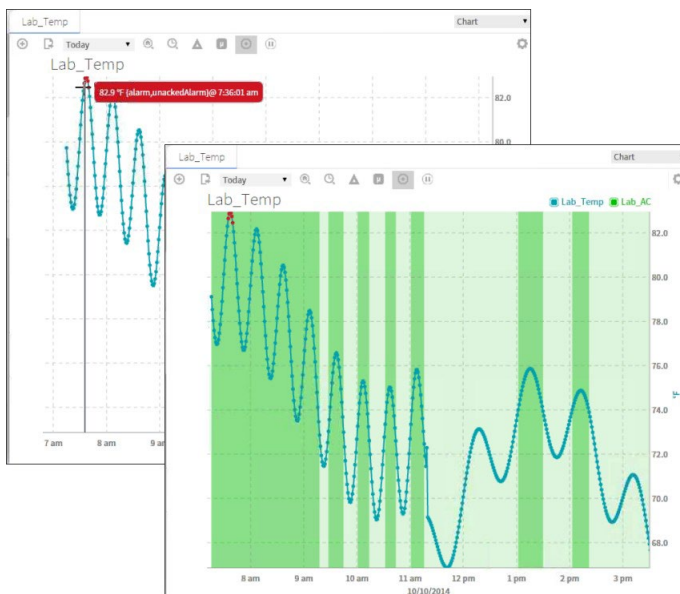
## workbench-WebWidget

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.

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 212: 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 213: CircularGauge WebWidget



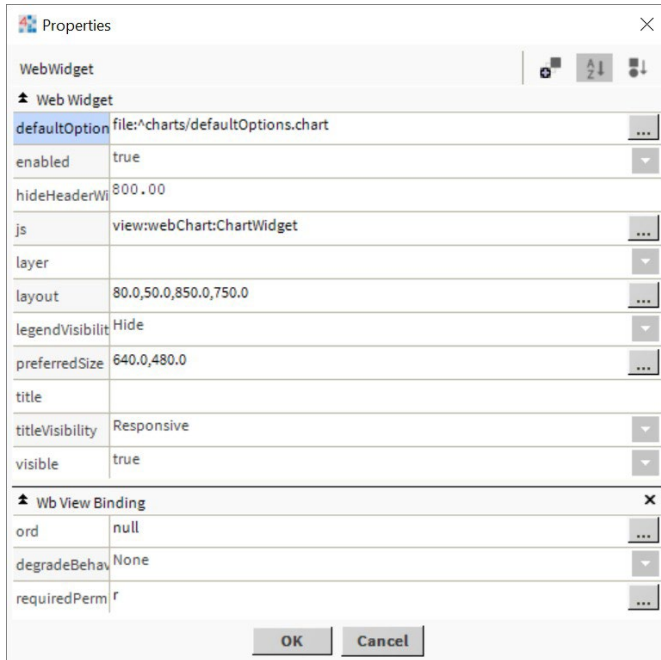
- A Dashboard may be added to any PxPage and displayed in the PxViewer. Additional WebWidgets may be added to the Dashboard pane to customize the presentation of data. The dashboard is used to write dashboard-specific data to and from a station for a specific user.

Figure 214: Dashboard WebWidget



## Configurable properties for the Chart widget

Figure 215: Chart widget properties



Property	Value	Description
defaultOption	file: ^charts/defaultOptions. chart	Provides ord for the defaultOption widget. You can browse to select another widget.
hideHeaderWidth	number	All headers are visible when you set the value 800 pixel and above.
Visible	true (default), false	Enables/disables display of the widget.
Enabled	true (default), false	Enables and disables use of the component.
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 Absolute, Percent, or Preferred units. The Fill checkbox turns fill on or off.
Js	view:webChart: ChartWidget (default)	Provides ord for the Javascript widget. You can browse to select another widget.
layer	drop-down	null
legendVisibility	drop-down	Provides legendVisibility as per set value. (for example <i>Responsive, Show, Hide</i> )
preferredSize	Default values: Width=000.00, Height=000.00	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 <i>Responsive, Show, Hide</i> )
wbViewBinding	Binding null — >WebWidget (default)	Provides ord for bound label. You can browse to select the Ord. Also provides selectable options for Degrade Behavior (None, Disable, and Hide).

# **CHAPTER 7 INTERFACE REFERENCE**

## **Topics covered in this chapter**

- ◆ About keyboard shortcuts
- ◆ Types of menu bar items
- ◆ Types of popup menu items
- ◆ Types of side bars
- ◆ Types of edit commands
- ◆ Types of toolbar icons
- ◆ Types of console commands

This section contains reference topics about keyboard shortcuts, menu bar items, menus, and toolbar icons.

- About keyboard shortcuts  
Explains how to use the keyboard to enter menu bar and popup menu commands – without using the mouse.
- Types of menu bar items  
Describes the different menu items available from the menu bar.
- Types of menu items  
Describes the menus that appear in different views and contexts.
- Types of toolbar icons  
Describes the types of icons that are in the toolbar.

## **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:

- 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)

## Types of menu bar items

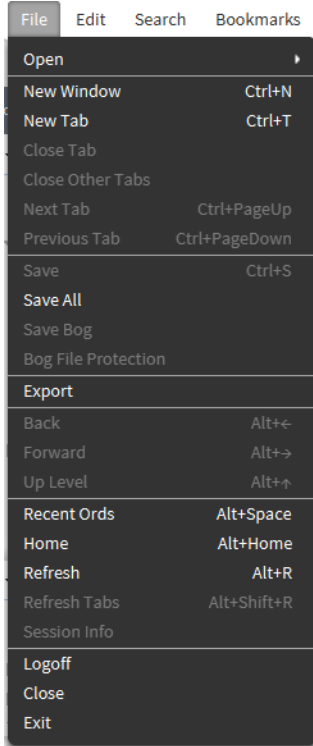
This section describes the menu items that appear in the Workbench menu bar:

- 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

The File menu in the menu bar provides the following options:

Figure 216: File menu



Menu	Description
Open	<p>This menu has submenus. :</p> <ul style="list-style-type: none"> <li>• <b>Ord:</b> You open option by selecting File→ Open→Ord. The Ord Chooser also allows you to directly type the Ord of a location to go there. If no view is specified, the default view for the item will be presented. You may also paste an Ord instead of typing it by pressing the paste shortcut Ctrl + V after you have copied a node or Ord into the clipboard.</li> <li>• <b>File:</b> You open a file by selecting File→Open→Open File from the main menu.</li> <li>• <b>Directory:</b> You open a Directory by selecting File→Open→ Open Directory from the main menu.</li> <li>• <b>Query:</b> You open a Directory by selecting File→Open→Query from the main menu.</li> <li>• <b>Platform</b> <ul style="list-style-type: none"> <li>– You open a Platform by selecting File→Open→ Open Platform from the main menu. This will cause the following window to be presented so that you can complete each field.</li> <li>– <b>Host</b> – This is the address of the computer running the Platform that you wish to access.</li> <li>– <b>Port</b> – This is the port used.</li> <li>– <b>Password</b> – This is the password given to you by your system administrator.</li> </ul> <p>Remember these credentials - This will save this connection in your connection list.</p> <p>Once you successfully connect to the Platform, it will appear in the tree and the available tools will be displayed.</p> </li> <li>• <b>Station:</b> 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. <ul style="list-style-type: none"> <li>– <b>Address</b> – This is the address of the computer running the Station that you wish to access.</li> <li>– <b>User name</b> – This is the user name given to you by your system administrator.</li> <li>– <b>Password</b> – This is the password given to you by your system administrator.</li> <li>– <b>Remember these credentials</b> – This will save this connection in your connection list.</li> </ul> <p>Once you successfully connect to the Station, it will appear in the tree and your home page will be displayed.</p> <p>If you will be away from the system, you should close the Station to prevent unauthorized access.</p> </li> <li>• <b>Find Stations:</b> 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.</li> </ul>
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 and 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.

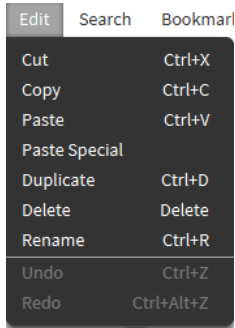





Menu	Description
Close Tab	Closes the existing tabs.
Next Tab	Selects the next tab. The shortcut is Ctrl + PageUp.
Previous Tab	Selects the Previous tab. The shortcut is Ctrl + PageUp.
Save	Saves the value of the Component.
Save All	Save all open views. This saves all Tabs when browsing with mutiple 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.
Refresh Tabs	Refreshes the 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.
Close	<p>You may close the current window at any time, if it is not the primary window, by any of the following methods:</p> <ul style="list-style-type: none"> <li>• Selecting File→Close from the main menu.</li> <li>• Clicking the window in the top left corner and selecting Close from the menu.</li> <li>• Clicking the close icon in the top right corner of the window.</li> </ul>
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.



## About the Edit menu

The Edit menu in the menu bar provides the following options:

Figure 217: Edit menu



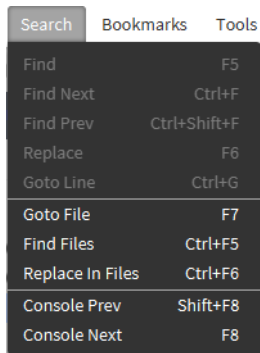
Menu	Description
Cut	<p>Copies the current selection to the clipboard, changes its color to gray and deletes it when you Paste it somewhere else. If the selection is a string, Cut copies the selection to the clipboard and removes it from the string. To use it, select an item and:</p> <ul style="list-style-type: none"> <li>In general, right-click the component(s) and click Cut in the menu. (To Cut more than one component, hold the Ctrl and click to select the items.)</li> <li>In the Nav tree, right-click and click Cut in the menu.</li> <li>In the Wire Sheet, select the component and click the Cut button ()</li> <li>In the Wire Sheet, select the component and click Edit→Cut from the main menu.</li> <li>In a window other than the main browser window, select the item to Cut and click the shortcut key Ctrl + X (hold down Ctrl and enter X).</li> </ul>
Copy	<p>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:</p> <ul style="list-style-type: none"> <li>In general, right-click the component and click Copy in the menu.</li> <li>In the Wire Sheet, right-click, select the component and click the Copy in the menu.</li> <li>In the Wire Sheet, select the component and click the Copy button ()</li> <li>In the Wire Sheet, select the component and click Edit→Copy from the main menu.</li> <li>In a window other than the main browser window, select the item to Copy and click the shortcut key Ctrl + C (hold down Ctrl and enter C).</li> <li>To copy a component, you must Copy it from the Palette or an existing Database. You can follow one of the following steps: <ul style="list-style-type: none"> <li>In the Palette sidebar, expand a module (like control) and sub-directory (like Points), Right-click on a Component (like BooleanWritable) and select Copy.</li> <li>In the namespaces sidebar (tree), expand System, Modules, a module (like control), Files, module.palette, and sub-directory (like Points), Right-click on a Component (like BooleanWritable) and select Copy.</li> <li>OR right-click on a component that you want to Copy from a Wire Sheet, Property Sheet or the tree and select Copy.</li> </ul> </li> </ul>
Paste	<p>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 into clipboard. It may be accessed by cutting or copying component(s):</p> <ul style="list-style-type: none"> <li>In general, right-click the component(s) and click Paste in the menu. (To Cut more than one component, hold the Ctrl and click to select the items.)</li> <li>In the Nav tree, right-click and click Paste in the menu.</li> <li>In the Wire Sheet, select the component and click the Paste button ()</li> <li>In the Wire Sheet, select the component and click Edit→Paste from the main menu.</li> </ul>

Menu	Description
	<ul style="list-style-type: none"> <li>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).</li> </ul> <p>You can use Drag to Cut and Paste in one operation.</p> <p>You can add a Component to a running Station or an offline Bog 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 at <code>module://control/doc/javax/baja/control/BBooleanWritable.bajadoc.do</code> as follows:</p> <ul style="list-style-type: none"> <li>In the Wire Sheet, right-click on the background and select Paste to add a new component. The new Component is created and selected.</li> <li>In the Nav tree, right-click on a container and select Paste to add the new component inside the container.</li> </ul>
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 access 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: <ul style="list-style-type: none"> <li>In general, right-click the component(s) and click Delete in the menu. (To Delete more than one component, hold the Ctrl and click to select the items.)</li> <li>In the Nav tree, right-click and click Delete in the menu.</li> <li>In the Wire Sheet, select the component and click the Delete button (  ).</li> <li>In the Wire Sheet, select the component and click Edit→Delete from the main menu.</li> </ul>
Undo	This reverses the last Action as if it had not been performed. It is only available for certain actions as follows: <ul style="list-style-type: none"> <li>Paste component</li> <li>Cut component</li> <li>Link</li> <li>Delete Links</li> </ul>
Redo	Restores an Action after Undo has removed it. It is only available after a successful Undo.

## About the Search menu

The Search menu in the menu bar has the following options:

Figure 218: Search menu



Menu	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 short-cut 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.

## About the Bookmarks menu

The Bookmarks menu in the menu bar has the following options:

Menu	Description
Add To Bookmarks	You may add a Bookmark by selecting Bookmarks → Add to Bookmarks from the main menu.
Manage Bookmarks	You may manage Bookmarks by selecting Bookmarks → Manage Bookmarks from the main menu.
Bookmarks Go Into	You may select Bookmarks by selecting Bookmarks → Go Into from the main menu.
Bookmarks File	You may select Bookmarks by selecting Bookmarks → File from the main menu.

## 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: <ul style="list-style-type: none"> <li>• General</li> <li>• Lexicon</li> <li>• Text Editor</li> <li>• Wire sheet</li> </ul>
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.

Menu	Description
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 Pass- word. 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 <code>local: file:!stations/station-name/config.bog bog: slot:/.</code>
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.

## About the Window menu

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 theWorkbench 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.

## 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:

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.

Item	Description
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	<p>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:</p> <ul style="list-style-type: none"> <li>• HxPx Media</li> <li>• Mobile PxMedia (obsolete)</li> <li>• Report PxMedia</li> <li>• Workbench PxMedia</li> </ul>

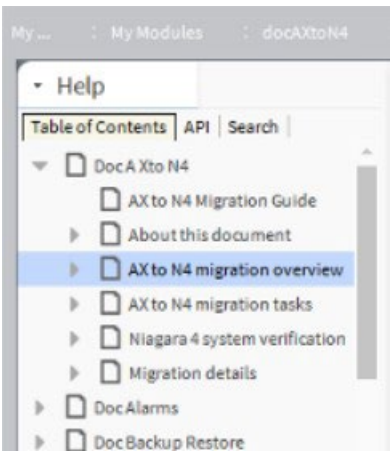


## About the History Ext Manager menu

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:

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.

## About the Help menu

The Help menu in the menu bar has the following options:

Menu	Description
 <p>Help Contents Index</p>	<p>The help contents provide 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.</p>
<p>Help On View</p>	<p>This provides help for the current Plugin. It is accessed by selecting Help→ On View from the menu with the Plugin in use.</p>
<p>Help Guide On Target</p>	<p>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.</p>
<p>ocm,sfx)=\graphics:graphic49AFCF2A688A307AA61E374A8EBFD39</p>	<p>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.</p>
<p> Help Find Bajadoc</p>	<p>This is accessed by selecting Help→ Find Bajadoc from the menu. It searches for the requested bajadoc.</p>
<p> Help About</p>	<p>This is accessed by selecting Help→ About from the menu. It provides the software release and license information.</p>

## Types of popup menu items

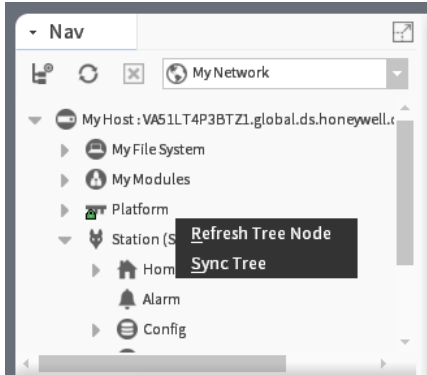
Workbench provides view–specific, or context–specific commands for editing components in many of the views. Following, is a list of the standard Workbench popup (right–click) menus.

- Nav side bar
- Wire sheet
- Property sheet
- Px Editor
- History extension manager
- Todo list
- Point Manager

## About the Nav side bar menu items

The Nav sidebar provides a tree-type hierarchical view of the system. The Nav side bar menu is the menu that displays command options when you right-click on a component item in the Nav tree.

Figure 219: Nav side bar popup menu with nothing selected

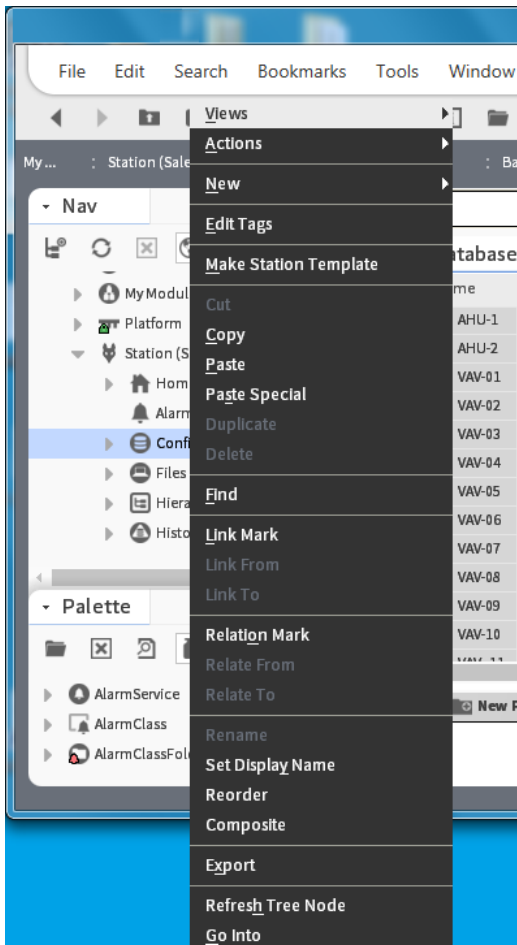


This popup menu has these options:

Item	Description
Refresh Tree Node	Refreshes the Nav tree.
Sync Tree	Synchronises the Nav tree.



Figure 220: Nav side bar popup menu with component selected



The popup menu has the following options:

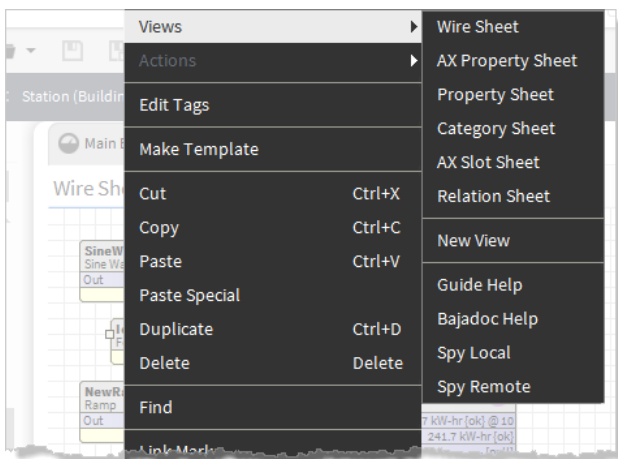
Item	Description
Views	Goes to any of the views of the selected component.
Actions	Perform any available actions on the component.
New	Provides additional options for creating new objects.
Edit Tags	Opens the Edit Tags window, permitting you to add or remove tags on a component.
Make Template	Creates 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 clipboard and after pasting the copy, deletes the cut item.
Copy	Copies to clipboard, copied item remains
Paste	Pastes the copied item from the clipboard
Paste Special	Gives you more control of how the content is displayed or functions when pasted.
Duplicate	Makes a copy in the same location as the original item
Delete	Removes the item
Find	Findes the item.
Link Mark	Sets a selected component to your popup menu, making it temporarily available for "Linking From" or "Linking To" other points.
Link From	Allows you to link to a selected component from another component that has been marked, using Link Mark from the popup menu.
Link To	Allows you to link 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" or "Relating To" other points.
Relate From	Allows you to add a relation from selected component to another that has been marked, using Link Mark from the popup menu.
Relate To	Allows you to add a relation to a selected component from another that has been marked, using Link Mark from the popup menu.
Rename	Allows you to rename 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	Allows you to set a display name for the selected component (as it appears in a Nav tree, and in the Wire Sheet, and Property Sheet views).
Reorder	<p>Opens the Reorder Points window, which provides the following commands for reordering points within the selected parent component.</p> <ul style="list-style-type: none"> <li>• Move Up</li> <li>• Move Down</li> <li>• Sort by Name</li> <li>• Sort by Type</li> <li>• Reset</li> </ul>

Item	Description
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	Go Into allows you to re-root the nav tree at any arbitrary node. Right-click the node and select the Go Into command. This will make that node the new root of the tree. All the nodes you have "gone into" are persistently saved as a special type of Bookmark. Use the pulldown to switch between 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 "Unpin" a pinned slot has the opposite effect.
More...	Indicates the presence of additional menu items. Click More... to display those items.

### About the Wire Sheet menu items

Most of the Wire Sheet menu commands are described in another topic in this document.

Figure 221: Wire sheet popup menu



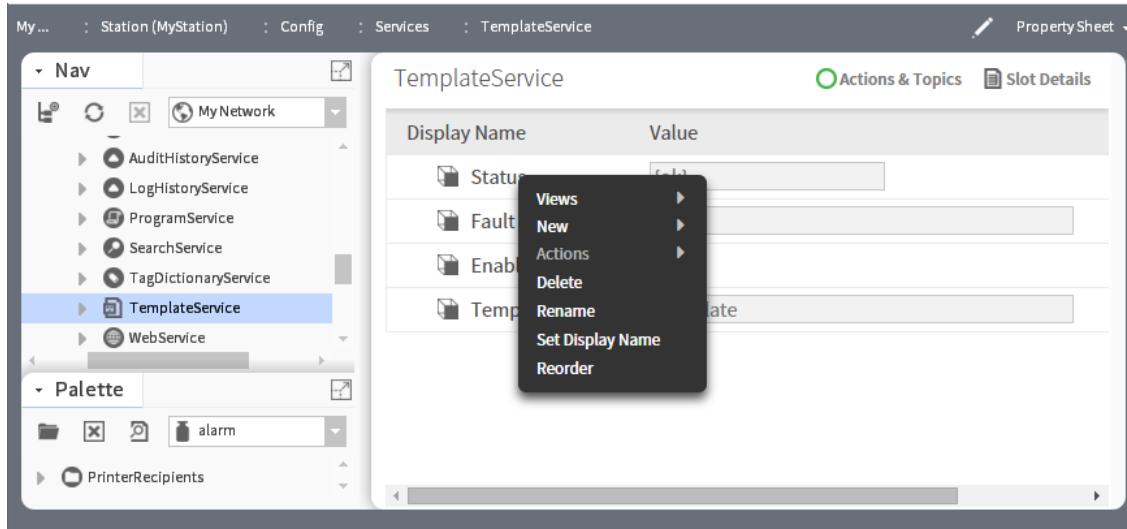
Following are wire sheet specific menu options:

Item	Description
Arrange	Provides options for aligning components on the wire sheet to make them easier to view. <ul style="list-style-type: none"> <li>• Arrange All — Redistributes the layout of all components on the wire sheet.</li> <li>• Arrange Selection — Redistributes the layout of all selected components on the wire sheet.</li> </ul>
Select all	Selects all components on the active wire sheet.

## About the property sheet popup menu items

Most of the property sheet menu commands are described below.

Figure 222: Property sheet popup menu

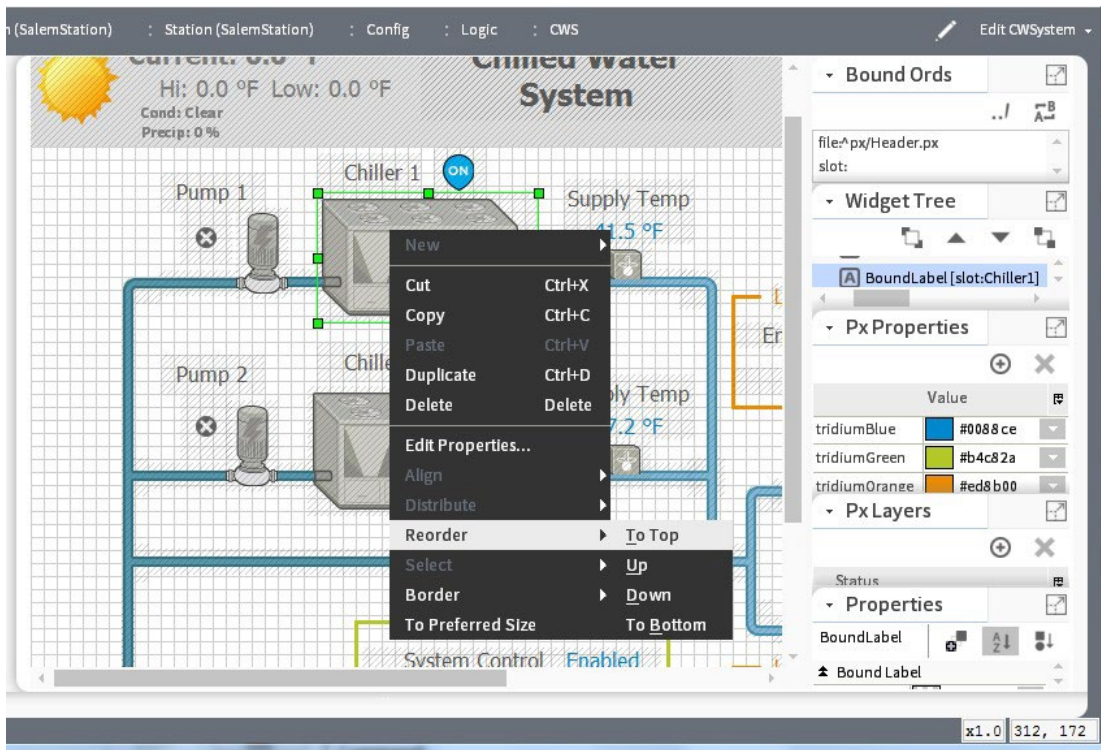


Item	Description
Views	Shows the different views of the component.
New	Uses to create new component, folder points, and text block.
Actions	There are different action options depending on coponent.
Delete	Deletes the component.
Rename	Renames the component.
Set Display Name	Sets the display name of the component.
Reorder	Records the component.
Config flags	Opens the Config window which you can use to add configuration flags on individual slots.

## 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.

Figure 223: Px Editor popup menu



The following menu commands are on the Px Editor popup menu:

Item	Description
New	Create a new item of standard types.
Cut	Copies to clipboard and after pasting the copy, deletes the cut item.
Copy	Copies to clipboard, copied item remains.
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...	Refer to the section “About the Properties side bar”.
Align	<ul style="list-style-type: none"> <li> <b>Left</b>                      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.                 </li> <li> <b>Right</b>                      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.                 </li> <li> <b>Top</b>                      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.                 </li> </ul>

Item	Description
	<ul style="list-style-type: none"> <li>• Bottom</li> </ul> <p>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.</p>
Reorder	<ul style="list-style-type: none"> <li>• To Top</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• Up</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• Down</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• To Bottom</li> </ul> <p>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.</p>
Border	<ul style="list-style-type: none"> <li>• Add Border</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• Remove Border</li> </ul> <p>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).</p>

## About the history extension manager popup menu items

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	This menu item provides a way to refresh the History Id after a rename. It applies the formatting property (as designated by the 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 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.
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	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. You can only rename one history at a time.
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.

## About the Todo list menu items

The Todo list menu has the following items:

Item	Description
Add	This menu item 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	This menu item, when clicked, dims and lines-through the selected item(s) in the Todo list so that the item(s) appears to be “crossed off” the list. If an item is already marked complete and this menu item is selected, the item will be restored to its “unmarked” state. With no items selected, this menu item is dimmed (unavailable).
Edit	Displays the Edit window, when clicked, 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, when clicked; deletes selected items when the prompt is affirmed.



## About the point manager menu items

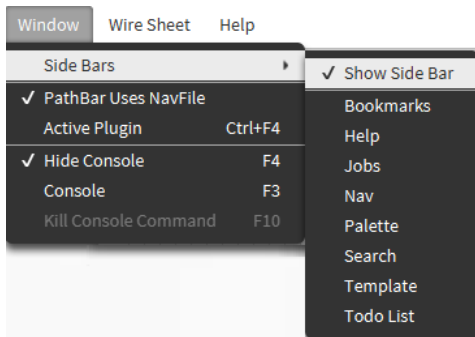
The point manager popup menu has the following items:

Item	Description
Views	This menu item provides 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	This menu item allows you to perform any available actions on the component. For example, write to a "writable" point.
New	This menu item allows you to add a new component to the pointsmanager. Valid options are presented in the submenu.
Link Mark	This menu item sets a selected point to your menu, making it temporarily available for "Linking From" or "Linking To" other points.
Link From	This menu item allows you to link a selected point to another point that has been marked, using Link Mark from the popup menu.
Link To	This menu item allows you to link a selected point to another point that has been marked, using Link Mark from the popup menu.
Rename	Select this menu item to rename the selected point. This menu item displays the Set Point Name window. You can only rename one point at a time.
Reorder	This menu item displays the Reorder Points window, which provides the following commands for reordering points within the selected parent component. <ul style="list-style-type: none"> <li>• Move Up</li> <li>• Move Down</li> <li>• Sort by Name</li> <li>• Sort by Type</li> <li>• Reset</li> </ul>
Composite	This menu item displays the Composite Editor window.
New Folder	This menu item allows you to add and name a new points folder.
New	This menu item allows you to add and name a new point.
Edit	This menu item displays the Point Editor window.

## Types of side bars

The Workbench interface may be customized by adding unique side bars that are designed to fit particular applications.

Figure 224: Default Side Bars menu



The following side bars may be displayed in the side bar pane by default:

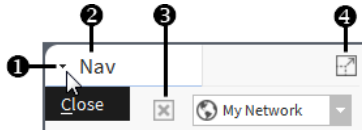
Side bar	Description
Bookmarks	Displays a list of your bookmarks.
Help	Provides a tree view of available help documentation.
Jobs	The Jobs side bar lists all current jobs in all of the stations with which you have a connection. The current status of each job is shown.
Nav	Provides a tree view of the 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 side bar title bar

All side bars have a title bar across the top. You can click and drag on the title bar to vertically resize the side bar or you can click on a minimized side bar to restore it to the previous size.

In addition, all side bars have the following controls:

Figure 225: Side bar title bar

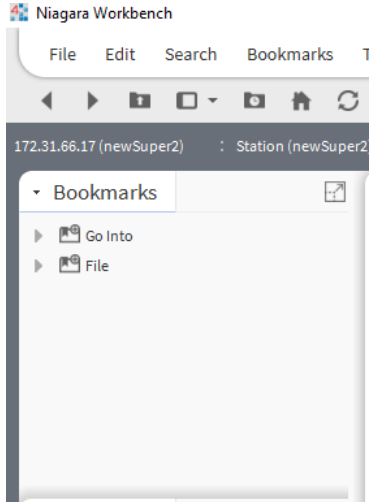


- ❶ Close dropdown — drop down control used to close the side bar
- ❷ Pane title— text that identifies the side bar type
- ❸ Close — alternative to the Close dropdown, closes the side bar
- ❹ Expand/Restore — clicking on the expand/restore icon causes the side bar to expand, filling the side bar pane and collapsing other open side bars. Click again to restore the normal side bar display.

## About the Bookmarks side bar

When you open the Bookmarks side bar, it appears in the side bar pane, as shown below.

Figure 226: Bookmarks side bar

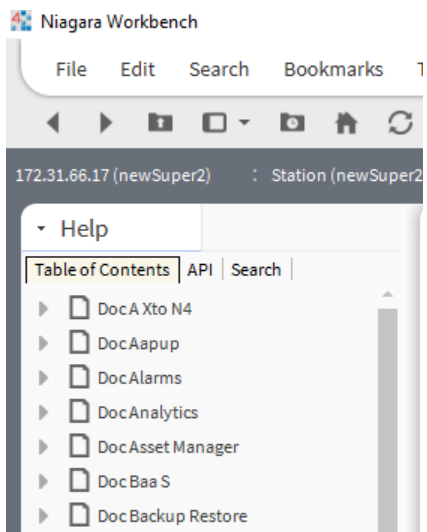


From the bookmark side bar, you can double click on bookmark nodes or use popup menus to perform all operations that are available from the side bar (for example, go directly to a bookmarked location, manage bookmarks, edit bookmarks, and more). The quick access provided here is very helpful for changing screens without having to go through multiple selections using other menus or submenus.

## About the Help side bar

When you open the Help side bar, it appears in the side bar pane, as shown below.

Figure 227: Help side bar



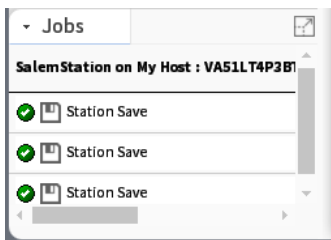
The help side bar has three tabs that you may select by clicking on the tab. The three help tabs are listed and briefly described, as follows:

Menu	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.

## About Jobs side bar

When you open the Jobs side bar, it appears in the side bar pane. The Jobs side bar contains a list of jobs that have been performed or that are currently being performed.

Figure 228: Jobs side bar

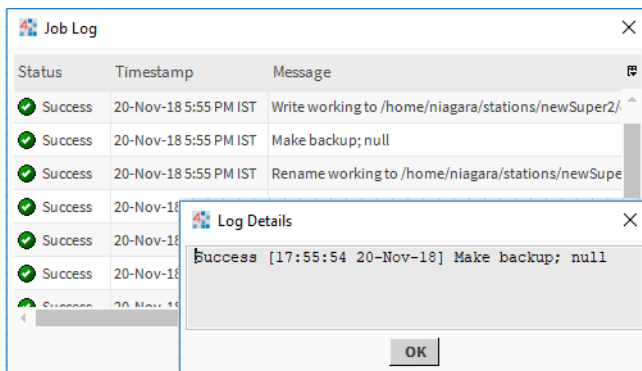


The following icons on the Jobs side bar indicate job status:

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 side bar, 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 229: Log Details window

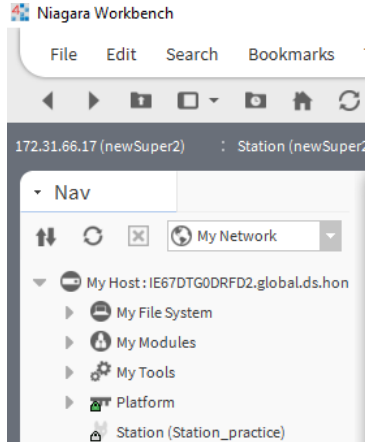


## About the Nav side bar

The Nav side bar contains the tree view that provides a hierarchical view of the whole system.

When you open the Nav side bar, it appears in the side bar pane, as shown below.

Figure 230: Nav side bar



Items are displayed in the tree with a symbol based on type. If the item is a file, the symbol reflects the file type.

At the highest level, the Nav side bar tree may include the following (when working from a localhost, as shown):

- My Host (local system)
- My File System
- My Modules
- Platform
- Stations (connected or disconnected)

From the Nav side bar, you can double click on nodes in the Nav tree or use popup menus to perform all operations that are available from the Nav side bar (for example, 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 through various screens and views in Workbench. Items are displayed in the tree with an icon that represents the associated function or file type.

### Types of nodes in the Nav tree side bar

The Nav tree side bar may include several different types of nodes and child nodes.

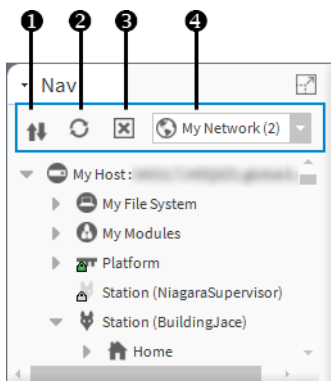
Menu	Description
My Host node	Represents a physical computer (hardware) that the rest of the nodes (subnodes) reside on.
My File System node	Represents the top level of a tree view of the host file system. File system subnodes represent drives and locations on the host system. It is important to understand 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.

Menu	Description
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 Niagara Platform Guide for more details.
Station node	<p>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).</p> <ul style="list-style-type: none"> <li>• Home</li> <li>• Alarm</li> <li>• Config node</li> </ul> <p>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:</p> <ul style="list-style-type: none"> <li>- Services Component for storing services, such as alarm service, his-tory service, tagdictionary service, and more.</li> <li>- Drivers Provides a place to store driver modules (such as the NiagaraNetwork, BACnet drivers, Modbus, and more).</li> </ul> <p>When expanded, displays a tree view of loaded driver modules. For more details, see the Niagara Graphics Guide.</p> <ul style="list-style-type: none"> <li>- Apps</li> <li>- Schedules</li> <li>- Control or Logic</li> </ul> <p>Control points may be displayed directly in the root of the Config node.</p>

### About the Nav tree side bar toolbar

In addition to the standard side bar title bar the Nav tree side bar has a toolbar, located just below the title bar.

Figure 231: Nav side bar tool bar



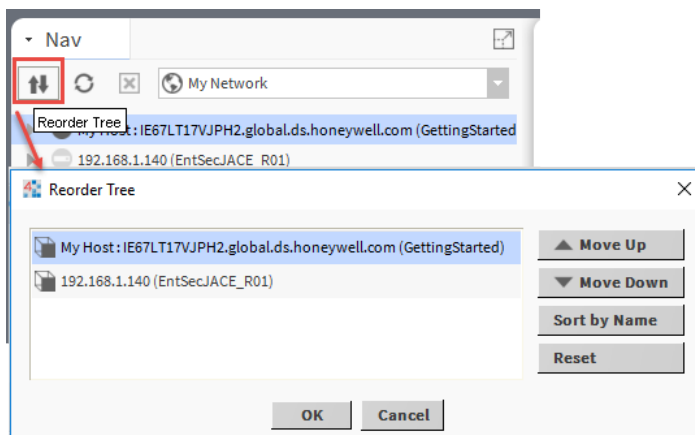
- ❶ Recorder Tree button — opens a new Recorder Tree window to make changes in the tree nodes.
- ❷ Sync tree button — synchronizes the tree node display with the currently selected view.
- ❸ Close tree button — closes the currently displayed side bar.
- ❹ Drop-down tree selector — when multiple Nav side bars 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. 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 232: Recorder Tree

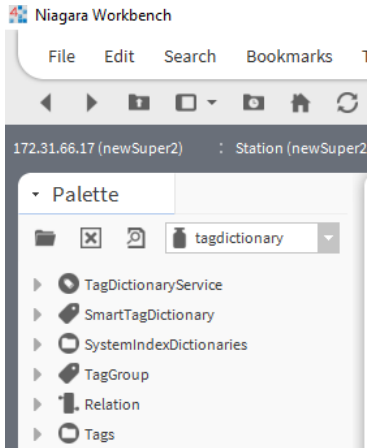


## About the Palette side bar

When you open the Palette side bar, it appears on the left side of the Workbench in the side bar pane. The Palette side bar provides a place to open and view sets of modules or custom palettes that you build for yourself.



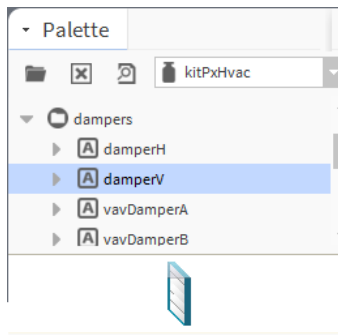
Figure 233: Palette side bar with preview pane



From the Palette side bar, 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 side bar (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 side bar also has a component preview pane (shown below) that displays an image (when available) of the selected component.

Figure 234: 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 `compPreviewWidget` property to a widget to display an image in the preview pane of the palette side bar.

### About the Palette side bar toolbar

In addition to the standard side bar title bar the Palette side bar has a toolbar, located just below the title bar.

Figure 235: Palette side bar tool bar



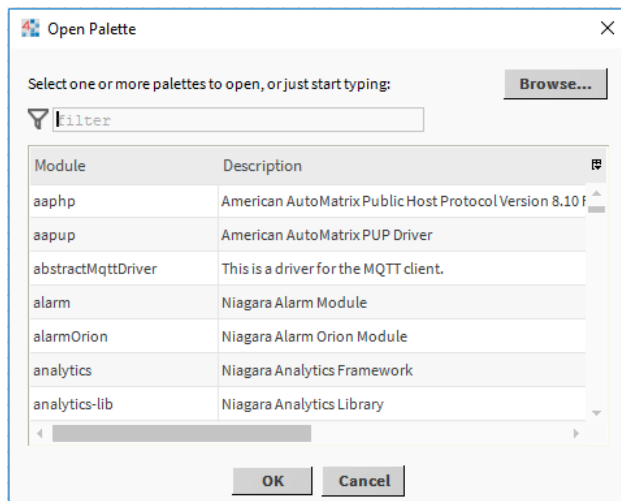
The palette toolbar includes the following:

<b>1</b>	Open palette button — opens the Open Palette window.
<b>2</b>	Close palette button — closes the currently displayed palette.
<b>3</b>	Preview button — toggles the preview pane on and off. Previews are available on some components.
<b>4</b>	Dropdown palette selector — when palettes are open in the palette side bar, this selector allows you to choose which palette to display.

### About the Open Palette window

The Open Palette 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 236: Open Palette window



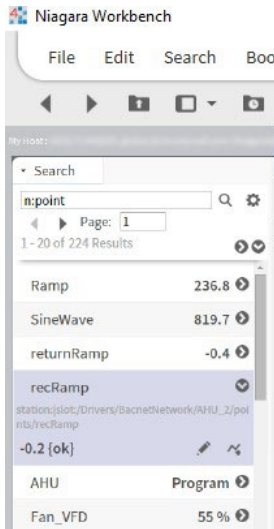
The Open Palette window is shown above and has the following features:

Menu	Description
Filter field	This is a text field that allows you to type the beginning letters of the desired palette name to filter out palettes from the view. 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" entry in this field. All palettes are listed in the table if no text is entered in this field.
Browse button	This button opens the File Chooser window to allow you to select palettes that are located in alternate locations.
Table of palettes	The table of palettes has the following columns.
Module	This is the name of the palette's parent module.
Description	This is a short title or name of the palette's parent module.

## About the Search side bar

The Search Service uses Niagara Entity Query Language (NEQL) syntax to query the system for component tags.

Figure 237: Search side bar



The Search side bar 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.

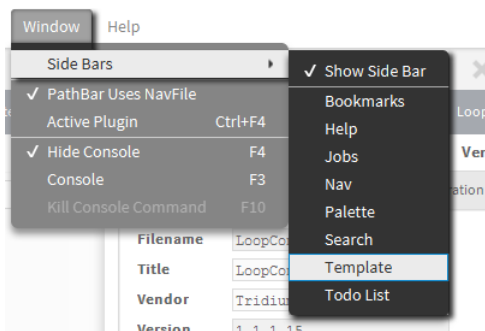
Click the gear icon to configure the number of search results to display per page. The “>” or “<” page through the results. The search results area provides access to additional information too. Click the “>” icon to the right of a result to expand it, showing the Ord and current status. In an expanded result, click the icons (at right) to view the live data either in a gauge or chart.

## Template side bar

The Template side bar 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.

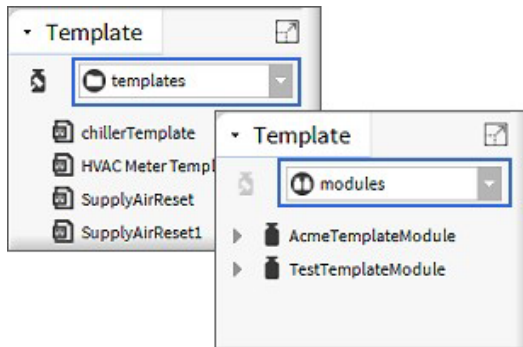
If not already visible in Workbench, display the side bar by clicking Window→Side Bars→Template, as shown here.

Figure 238: Template side bar



In the side bar, 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 239: Template side bar



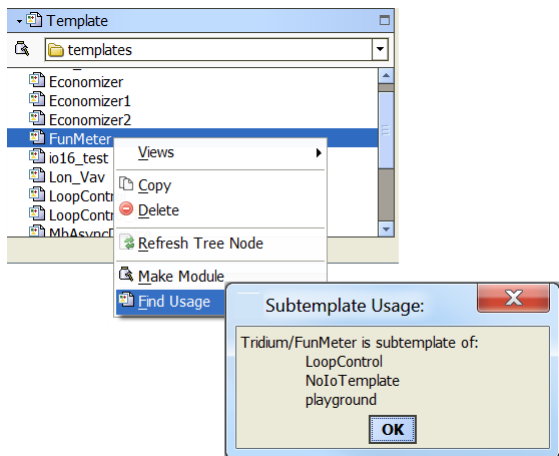
Double-click on a template file to open it in the Template view. When you open a file in the templates folder, you can proceed to make changes and save the file. Optionally, you can create a new variation of an existing template by clicking Save As in the view to save it with a new name.

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 4.3 and later, 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.

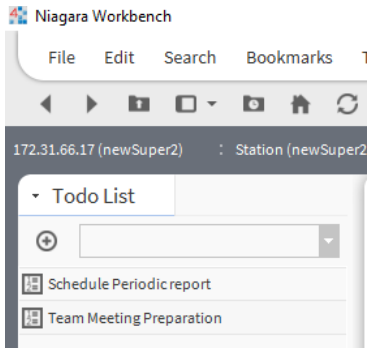
Figure 240: Find Usage menu option invokes Subtemplate Usage window



### About the Todo list sidebar

The Todo List sidebar is a convenient way to create and access Todo List items from a palette.

Figure 241: Todo bar

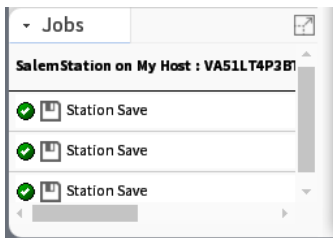


Click the + button on the toolbar to open the Add window. This window provides a text fields for adding and categorizing Todo list items.

### About the Jobs side bar

The Jobs side bar shows all the current jobs in all the stations with which you have a connection.

Figure 242: Jobs side bar



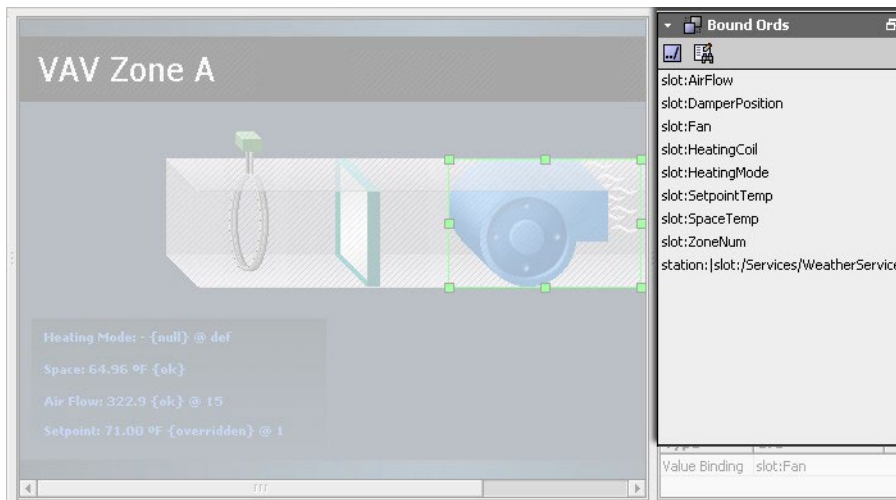
The current status of each job is shown as: running, canceling, canceled, success, or failed. If the job is running, a progress bar displays estimated progress.

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.

### About the bound ords side bar

The bound ords side bar is available when the Px Editor view is active. It displays a listing of all the bound ords in the current Px view.

Figure 243: Bound ords side bar

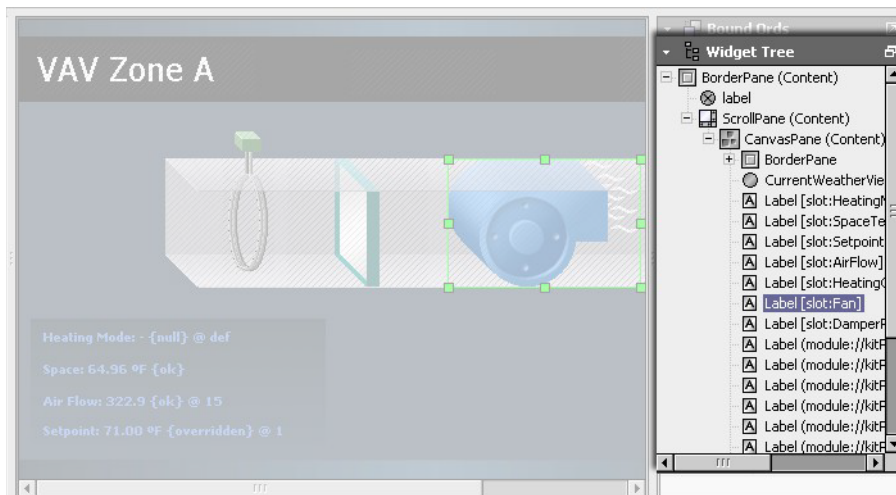


Double click on any ORD in the list to display the ORD in the ORD editor window.

### About the widget tree side bar

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 244: Widget tree side bar

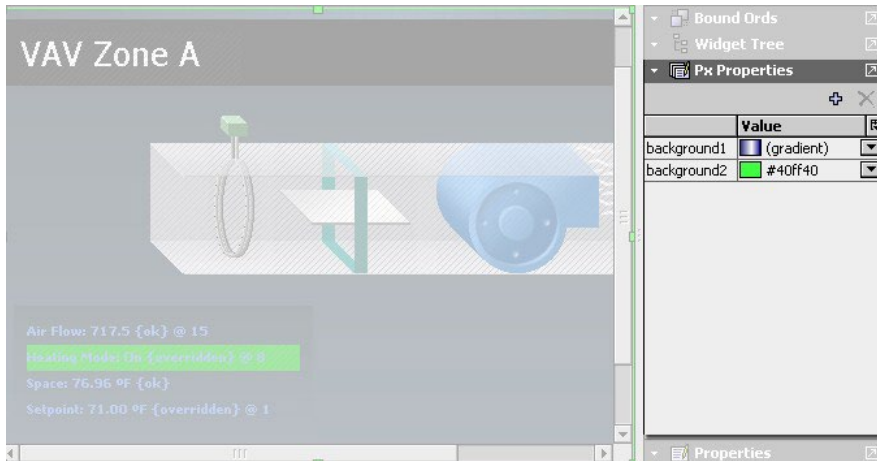


It is often easier to use the Widget Tree to select objects when you have a lot of objects on a view—especially when there are several layers of objects. When you select an object in the tree view it is selected in the Px view as well and displays the selection borders and handles.

## About the Px properties side bar

This side bar 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 245: Px properties side bar

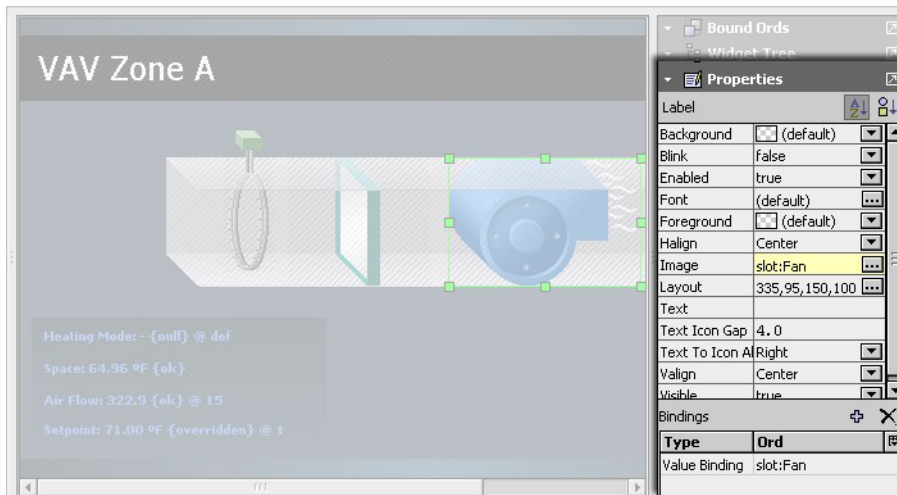


Use the menu bar icons to add, define, assign, and delete Px properties. For more information about px Properties, see the Niagara Graphics Guide.

## About the Properties side bar

This side bar 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 246: Properties side bar



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 side bar).

## Types of edit commands

In addition to view-specific editing tools, Workbench provides commands for editing components in many of the views. Following, is a list of descriptions of the standard commands that are available in Workbench views for editing components.

Command/Action	Description
Drag	Dragging files or components only works within a single Workbench application. However, it is possible to drag files from Windows Explorer into the Workbench to see the default 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	Use the Cut command to delete the selected object and send it to the clipboard.
Copy	Use the Copy to send the selected object to the clipboard without deleting it
Paste	Use the Paste command to copy the current contents of the clipboard to the destination as a set of new dynamic properties.
Duplicate	Use Duplicate to create a copy of the current selection in the same container as the selection.
Delete	Use the Delete command to remove a selected item from its parent container.
Undo	Use the Undo command to reverse the previous command. Undo is only available for certain commands, such as, Paste, Cut, Delete, and the Link action
Redo	Use the Redo command to restore a command-action after the Undo command has removed it.
Rename	Use Rename to change the name of a component.

## Types of toolbar icons

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.



Icon	Description
 Back	Hyperlink back to last displayed URL
 Forward	Hyperlink forward to the URL selected by the Back command
 Up Level	Jump to the parent of current page

Icon	Description
 Side Bars	Display the Side Bars list.
 Recent Ords	Display recent Ords list.
 Home	Jump to the home URL (Workbench splash screen).
 Refresh	Refresh the current page.
 Refresh Tabs	Refresh all open tabs.
 Info	Information on the current connection.
 Open	Display the Open list.
 Save( Ctrl + S)	Save the changes to the current component.
 Save Bog	Save changes to the bog file.
 Bog File Protection	Enter, change, or add bog file passphrase.
 Export	Export the current view or object.
 Cut (Ctrl + X)	Cut selected item and place it on the clipboard.
 Copy (Ctrl + C)	Copy selected item and place it on the clipboard.
 Paste (Ctrl + V)	Paste the items on the clipboard.
 Duplicate (Ctrl + D)	Copy and paste to duplicate the item.
 Delete	Delete the current selection.
 Undo (Ctrl + Z)	Reverse the previous command/action.
 Redo (Ctrl + Alt + Z)	Restore a command/action after the Undo command has removed it.

## Slot Sheet 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.

Icon	Description
 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.





## Px Editor toolbar icons

When the Px Editor view is the active the following additional icons are available.

Icon	Description
 Toggle View/Edit Mode	Displays, alternately, the Px Editor or 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. For more details, see the Niagara Graphics Guide.
 Toggle Browser Preview Mode	Visible in Px Viewer mode. Displays, alternately, the Px Viewer or 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. For more details, see the Niagara Graphics Guide.
 Right (Px Editor) side bar menu	Displays a dropdown list of side bar options for the Px Editor. The following options are available: <ul style="list-style-type: none"> <li>• Bound Ords Shows or hides the Bound Ords side bar.</li> <li>• Widget Tree Shows or hides the Widget Tree side bar.</li> <li>• Properties Shows or hides the Properties side bar.</li> </ul>
 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.





## About the history extension manager toolbar icons

Following, are the icons that appear when the History Extension Manager view is active.

Icon	Description
 HistoryExtManager Enable Collection	Click this icon to enable (start the collection process) for the selected entries.
 HistoryExtManager Disable Collection	Click this icon to disable (stop the collection process) for the selected entries.
 HistoryExtManager Rename History	Click this icon to rename the selected history. When clicked, the Set History Name dialog box appears.
 HistoryExtManager Edit System Tags	Click this icon to open 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.







## About the history editor toolbar icons



Following are additional icons that appear when the History Editor view is active.

Menu	Description
 Hide	With one or more records selected, this icon is available to set the trend flag of all selected records to "hidden". With no records selected, the icon is dimmed (unavailable).
 Unhide	Selecting this will restore visibility of 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.

## About the Todo list toolbar icons

Following are the icons that appear when the Todo List view is active.

Icon	Description
 Add	This icon opens the Add dialog box, when clicked. 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	This icon, when clicked, dims and lines-through the selected item(s) in the Todo list so that the item(s) appears to be “crossed off” the list. If the item is already marked as completed and this icon is selected, the item will be restored to its “unmarked” state. With no items selected, this toolbar icon is dimmed (unavailable).
 Edit	Displays the Edit dialog box, when clicked, 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.

Icon	Description
 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.

## Types of console commands

The Workbench console commands provide additional functionality.

The following types of commands may be typed in from the Workbench console.

Command	Description
Niagara shell commands	These commands are used at the command line and may be typed in directly
nre commands	These commands are used at the command line and may be typed in using the “nre” prefix.
wb commands	These commands are used at the command line and may be typed in using the “wb” prefix.
plat commands	These commands are used at the command line and may be typed in using the “plat” prefix.

## Shell commands

Menu	Description
Cd	This command displays and changes the current directory. Type <code>cd &lt;directory name&gt;</code> to change to a specific directory. Type <code>cd</code> to display the current directory.
Debug	This command 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	This command prints a message to the output stream. Type <code>print &lt;message&gt;</code> to print the literal message to the console.
Reset	This command resets the environment to its default state. Type <code>reset</code> to set the environment to its default state.
Set	This command displays and modifies environment variables. <ul style="list-style-type: none"> <li>Type <code>set</code> to display all the environment variables.</li> <li>Type <code>set &lt;prefix&gt;</code> to display all the environment variables that start with the specified prefix.</li> <li>Type <code>set &lt;name&gt;=</code> to remove the “named” variable.</li> <li>Type <code>set &lt;name&gt;=&lt;value&gt;</code> to set the “named” variable to the “value” specified.</li> </ul>
Which	This command resolves a filename in a path. Type <code>which &lt;filename&gt;</code> to find the first occurrence of the specified “filename”.

## nre (station) commands

Use the following syntax with the nre command.

```
nre [options] <class> [args]*
```

The following parameters may be used with the nre command.

Command	Description
Class	This is a class name or a module: classname to execute.
Args	This is the name of one or more arguments to pass through tomain.

The following options may be used with the `nre` command.

Option	Description
<code>-version</code>	This option displays the nre version.
<code>-modules:&lt;x&gt;</code>	This option displays the modules that match the pattern defined by "x".
<code>-hosted</code>	This option displays the id for the system host.
<code>-licenses</code>	This option displays a summary of the license information.
<code>-props</code>	This option displays a list of the system properties.
<code>-locale:&lt;x&gt;</code>	This option allows you to set the default locale. For example, to set the default locale to US English, type: <code>-locale:en_US</code>
<code>-@option</code>	This option allows you to pass the specified option to the JavaVM.
<code>-testheap</code>	This option tests and displays the max heap size.
<code>-buildreg</code>	This option causes a rebuild of the registry.

## wb (Workbench) 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	This option specifies the ORD of the initial view that you want to display when Workbench starts up.

The following options may be used with the `wb` command.

Option	Description
<code>-profile</code>	This option specifies the Workbench profile to assign when Workbench starts up.
<code>-file:ord</code>	This option specifies the initial file to display when Workbench starts up.
<code>-locale&lt;x&gt;</code>	This option sets the locale on startup.
<code>-@&lt;option&gt;</code>	This option allows you to pass the specified option to the JavaVM.

## plat (platform) 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>	This command displays a configuration summary for a remote host.
<code>fget</code>	This command gets one or more files from a remote host.
<code>flist</code>	This command provides file details for a single file, or for all files in a directory.
<code>ipconfig</code>	This command displays the TCP/IP configuration for a remote host.
<code>jacejar</code>	This command creates Niagara module files that can be run on embedded hosts.
<code>liststations</code>	This command lists stations that are managed by the Niagara platform daemon.
<code>moduleinstall</code>	This command installs Niagara modules to a remote host.
<code>reboothost</code>	This command requests that a remote Niagara platform daemon reboot its host.
<code>script</code>	This command runs one or more platform commands in a script.
<code>startstation</code>	This command requests that the Niagara platform daemon start a station.
<code>stopstation</code>	This command requests that the Niagara platform daemon stop a station.
<code>tellstation</code>	This command sends text to the console of a running Niagara station.
<code>watchstation</code>	This command monitors the output of the Niagara station.
<code>installdaemon</code>	This command installs the Niagara platform service (Win32 only).
<code>uninstalldaemon</code>	This command removes the Niagara platform service (Win32 only).
<code>installdialup</code>	This command installs the Niagara dialup service (Win32 only).
<code>uninstalldialup</code>	This command removes the Niagara dialup service (Win32 only).

The following options may be used with the `plat` command.

Option	Description
<code>-usage</code>	<code>plat -usage</code> prints the help listing in the console <code>plat &lt;command&gt; -usage</code> prints the command specific usage in the console
<code>-?</code>	<code>plat -?</code> prints the help listing in the console
<code>-help</code>	<code>plat -help</code> prints the help listing in the console <code>plat &lt;command&gt; -help</code> prints the command specific usage in the console
<code>-locale:&lt;x&gt;</code>	This option sets the default locale (en_US).
<code>-@&lt;option&gt;</code>	This option passes the option to the Java VM.
<code>-buildreg</code>	This option forces a rebuild of the registry.



# GLOSSARY

alarm	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.
alarm console	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).
alarm portal	A table view that lists alarms collected from multiple stations. To access this view from the main menu, click Tools→AlarmPortal.
category	<p>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).</p> <p>You manage categories using the <b>CategoryService</b>.</p>
certificate	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.
component	<p>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.</p> <p>Components differ from modules in that components comprise an implementation of the framework, whereas modules comprise the framework software itself.</p>
config flag	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.
container	For example: WebWidget
control point	<p>In the narrowest terms, control points refer to the eight point types found in the <b>Baja</b> control palette under the Points folder. In broader terms, control points include components from the <b>control</b> and <b>kitControl</b> palettes.</p> <p>Most of these components are based on the eight basic point types. They inherit from BooleanPoint, EnumPoint, NumericPoint, and StringPoint.</p> <p>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. AllControlPoints have a StatusValue property called Out.</p> <p>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.</p>

framework	Software that provides generic functionality. The framework can be customized by adding user-written code.
history	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.
niagarad	The Niagara daemon (niagarad) is a server process used to communicate between Workbench (as a client) and the platform that it is connected to.
node	<ol style="list-style-type: none"> <li>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.</li> <li>2. A position in a Nav tree hierarchy.</li> <li>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 Niagara Analytics Framework data model provide the structural framework for the model.</li> </ol>
object	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.
ORD	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.
palette	The palette provides a hierarchical view of available components. You copy or drag a component from a palette and paste it or drop it where you need it — on a wire sheet, property sheet, Px View, or in the palette Nav side bar pane.
point extension	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.
PX Editor	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.
template	<p>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.</p> <p>Templates are indexed and searchable.</p>

# INDEX

<b>A</b>	
actions	
default.....	111
Add Series command .....	60
Adding a log category .....	148
Advanced Krb5 Conf Editor view .....	226
API.....	21
authentication	
AXDigestScheme .....	175
DigestScheme .....	174
HTTPBasicScheme .....	176
authentication scheme .....	198
AuthenticationService.....	173
Auto Sampling setting .....	66
AX to N4 Migration Tool.....	93
AXDigestScheme .....	173, 175
Axis Orientation setting.....	65
<b>B</b>	
Background Color setting .....	65
baja-AuthenticationSchemeFolder .....	173
baja-AuthenticationSchemes .....	173
baja-Category .....	176
baja-FoxBackupJob .....	171
baja-Job.....	179
baja-Module .....	180
baja-ModuleSpace .....	180
baja-PermissionsMap.....	180
baja-SSOConfiguration .....	173
baja-Station.....	181
baja-User.....	182
baja-UserPrototype.....	182
Basic Krb5 Conf Editor view .....	225
batch editing .....	56
Batch Editor .....	91
BFormat.....	38
alarm extension example .....	39
BFormat errors .....	46
BFormats	
call resolution sequence .....	38
naming histories .....	41
Bookmarks side bar .....	273
boolean points .....	191
boolean writeable.....	191
bound ords side bar .....	282
BoundLabel.....	44
BFormat scripts for points .....	44
<b>C</b>	
Category	
Browser .....	233

	creating for code-signing certificate.....	85
	customize Workbench.....	71–72
Category Sheet.....		235
certificate		
importing into the User Key Store .....		87
installing in a remote platform/station.....		91
signing.....		86
Certificate Authority.....		84
Changing a log level.....		148
Chart Cursor.....		65
Chart type setting.....		64
Chart view .....		57
charts		
controls and options .....		57
code		
signing.....		88
code signing.....		83
Code Signing Options .....		83
code-signing		
troubleshooting .....		94
warning.....		83
code-signing certificate .....		84
Color (data color) setting .....		64
Command Bar.....		59
Commands.....		59
component guides.....		169
components.....		25
composites .....		130, 132
config flags .....		219
configure URL whitelist.....		70
Configuring logging in Niagara.....		148
control points.....		107
controls		
for charts.....		57
creating		
tabs.....		72
Creating a new station .....		158
Credentials Manager .....		149
CSR		
	<b>D</b>	
	Data Points setting.....	67
	Data Value popup setting .....	66
	Data Zoom Scope setting .....	65
	default scripts .....	46
	Delta command .....	61
	Desired Period setting.....	67
	DigestScheme .....	173–174
	driver upgrade .....	152–153
	driver upgrade tool.....	139

## E

enum point.....	191
enum writable .....	191
EnumWritable .....	116
escaped names .....	27
examples .....	39
alarm extension using BFormat scripts .....	39
exception	
approving for a code-signing certificate .....	90

## F

facets.....	116
Facets Limit Mode .....	65
fallback action .....	110
file types.....	36
Find Usage .....	281
FIPS 140-2	
options .....	77
Fixed Data popup setting .....	66
flags status .....	126
folder-level independent BFormat method.....	41
framework.....	15

## G

global password configuration.....	174
------------------------------------	-----

## H

Help side bar .....	273
help- BajadocOptions .....	195
help-BajadocViewer.....	218
hierarchy	
setting up on a Wire Sheet.....	104
histories	
naming using a BFormat script.....	41
History Chart controls and options .....	67
history extension .....	41
history extensions .....	121
Home Zoom command.....	61
html-WbHtmlView.....	220
HTTPBasicScheme.....	173, 176

## I

isValid status check.....	127
---------------------------	-----

## J

Java.....	18
Jobs side bar.....	274, 282
JVM .....	18
jxBrowser.....	213
JxWebBrowserImpl.....	213

**L**

Lexicon Module Migrator .....	145
Lexicon tool .....	143
license .....	163
link	
knobs .....	106
selection .....	106
link objects on the wire sheet .....	101–102
linking	
multiple links .....	102
linking rules .....	128
links	
editing .....	104
viewing on the Wire Sheet .....	103
Loading splash screen .....	97
Loading            Splash            Screen	
Replacing .....	97
local license database .....	146
Logger Configuration tool .....	146

**M**

maximum override duration .....	116
menu bar .....	49
minimum on and off times .....	129
Mobile Client Environment .....	211

**N**

Nav	
popup menu items .....	263
side bar toolbar .....	276
Nav popup menu items .....	263
Nav side bar .....	275
Nav tree side bar .....	275
NDIO to NRIO conversion .....	152–153
new features .....	15
New Station tool	
About .....	156

action on Finish .....	156
configurable parameters .....	156
New Station Wizard .....	158, 161
Niagara Tools .....	146
NumericWritable .....	116

**O**

object	
linking on the wire sheet .....	101–102
object signing .....	83
Object-to-String scripting .....	44
objects	
maintaining in view on the wire sheet .....	105
offline bogs	
recompiling and signing .....	93
Console .....	49

Open Palette window .....	279	Properties side bar .....	284
options		property sheet .....	240
for charts .....	57	popup menu .....	267
ORDs .....	32	provisioning	
override actions .....	109	installing a certificate .....	91
<b>P</b>		proxy extension .....	120
Palette		Px .....	27
side bar toolbar .....	278	Px Editor	
Palette side bar .....	277	popup menu items .....	267
password		Px Editor menu .....	261
global configuration .....	174	Px properties side bar .....	283
password configuration .....	182, 189	Px widgets	
path bar .....	49	and BFormats .....	43
Pause command .....	62		
Platform Connections options .....	82		
point actions .....	109		
point extensions .....	119		
point properties .....	108		
point status .....	124		
points			
naming using BFormat scripts .....	39		
popup			
editing .....	71		
menu			
popup menus .....	54		
property sheet .....	267		
presentation .....	27		
priority input scan .....	128		
priority level conventions .....	128		
priority linking rules .....	128		
Program Editor .....	91		
program			
object			
signing .....	88		
program object signing .....	83, 89		
program objects			
recompiling and signing .....	93		
Program Service .....	196		
propagate flags status .....	126		
properties .....	219		
Chart widget .....	215, 250		

<b>R</b>		
	recompiling and signing program objects .....	93
	recordtree .....	277
	related documentation .....	13
	related objects .....	124
	Removing a log category .....	148
	Robot Editor .....	91
	root CA certificate	
	installing in remote platform/stations .....	91
<b>S</b>		
	saml-CircleOfTrust .....	201
	saml-SAMLIdPService .....	200
	SAMLAttributeMapper .....	199
	SAMLAuthenticationScheme .....	198
	Sample Size setting .....	67
	Sampling command .....	62
	Sampling Period setting .....	67
	Sampling setting .....	67
	Sampling Type setting .....	66
	Save command .....	60
	schemes	
	about .....	33
	types of .....	34
	Search	
	side bar .....	279
	self-signed certificate	
	approving an exception .....	90
	set action .....	110
	Settings	
	Chart view .....	62
	Show Data Gaps .....	66
	Show Grid setting .....	65
	Show Start Trend Gaps .....	66
	side bar pane .....	49
	side bars .....	263
	about .....	53
	Bookmarks .....	273
	bound ords .....	282
	controls .....	272
	default .....	271
	Help .....	273
	Jobs .....	274, 282
	Nav .....	275
	Nav tree .....	275
	Palette .....	277
	Properties .....	284
	Px properties .....	283
	Search .....	279
	Todo List .....	281
	widget tree .....	283
	signing a module .....	142
	Single Sign On	
	configuration properties .....	173
	SSO .....	173, 198–199
	station template	



- creating ..... 161
- Linux Supervisor..... 160
- stations ..... 31
- status..... 127
- Status Coloring command..... 62
- Status Coloring setting ..... 66
- status flags ..... 124
- Subtemplate Usage ..... 281
- systems capabilities
  - distributed systems ..... 19
  - embedded systems ..... 19

**T**

- Table controls and options ..... 55
- tables
  - batch editing ..... 56
- tabs
  - closing ..... 73
  - opening a new tab ..... 72
- Template side bar..... 280
- templateoptions ..... 96
- templates ..... 197
- text file editor ..... 246
- text scripting ..... 44
- Time Range command ..... 61
- Time Zoom command..... 61
- title bar ..... 272
- Todo List side bar ..... 281
- tool bar ..... 49
- Tools ..... 138
  - Embedded Device Font Tool ..... 139
  - Jar Signer Tool..... 141
  - Kerberos Configuration Tool..... 142
  - Module Info..... 151
- Tools Menu ..... 158
- troubleshooting
  - BFormat..... 46
  - code-signing..... 94

**U**

- user prototypes ..... 183
- using the station template..... 160

**V**

- view pane .....49
- view selector .....49
- Viewing logged data ..... 149
- virtual components..... 189
- virtual gateway..... 189
- virtual machine ..... 18

<b>W</b>	
WeatherService	
and BFormats .....	45
Web Browser .....	97
Web Browser View .....	248
web Chart plugin .....	57
Web Service .....	203
web-browser whitelist	
configure .....	70
widget tree side bar .....	283
Window controls .....	52
wire sheet .....	96
basic linking .....	101
continuous linking .....	102
maintaining objects in view .....	105
object management .....	101
zoom controls .....	105
Wire Sheet	
configuring link colors .....	103
popup menu items .....	266
setting up a hierarchy of object relationships .....	104
viewing links .....	103
Workbench .....	49
creating additional windows .....	71
customizing .....	69
options .....	73
properties .....	74
tools .....	138–139, 141–142, 151
Workbench Fox Analyzer .....	165
Workbench GUI .....	49
Workbench Library Service .....	167
Workbench Tools .....	165
workbench-JobServiceManager .....	240
workbench-PropertySheet .....	240
workbench-TextFileEditor .....	246
workbench-WebBrowser .....	212
workbench-WebBrowserView .....	248
workbench-WebWidget .....	214, 248
Workbenchtools .....	137
writable points .....	127
<b>Z</b>	
zoom controls	
wire sheet .....	105



LG Electronics, U.S.A., Inc.  
Air Conditioning Technologies  
4300 North Point Parkway  
Alpharetta, Georgia 30022  
[www.lghvac.com](http://www.lghvac.com)

LG Customer Information Center, Commercial Products

1-888-865-3026 USA

Follow the prompts for commercial A/C products and parts.