

Technical Document

Getting Started with Niagara

December 8, 2015

niagara⁴

Getting Started with Niagara

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, NiagaraAX Framework, and Sedona Framework are registered trademarks, and Workbench, WorkPlaceAX, and AXSupervisor, are trademarks of Tridium Inc. All other product names and services mentioned in this publication that is 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 © 2015 Tridium, Inc. All rights reserved.

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

Contents

- About this guide13**
 - Document change log 13
 - Related documentation 13

- Chapter 1 About the Niagara framework15**
 - About Niagara 4..... 15
 - About control systems integration 17
 - About Java 18
 - About common networking and Internet protocols..... 19
 - About programming for non-programmers 19
 - About systems capabilities 19
 - About component software design 19
 - About the software architecture 20
 - About Baja..... 21
 - About APIs 21
 - About Niagara building blocks..... 22
 - About modules 22
 - About components..... 25
 - About presentation 27
 - About stations 31
 - About ORDs 32
 - About views..... 36
 - About lexicons 37

- Chapter 2 About Workbench39**
 - Tour of the Workbench GUI 39
 - Workbench window controls..... 43
 - About the side bar panes..... 44
 - About popup menus..... 45
 - Table controls and options..... 46
 - Editing multiple rows in a table 47
 - Controls and options for charts 48
 - Chart view 48
 - History Chart view..... 55
 - Customizing the Workbench environment 57
 - Creating Additional Windows 57
 - Editing options in the "new" popup menu..... 57
 - Creating multiple tabs in the view pane..... 57
 - Types of Workbench options..... 59
 - About Workbench themes 66
 - About branding the Workbench splash screen 68

- Chapter 3 Data and Control Model71**
 - Wire Sheet object management 71
 - Basic object linking..... 71
 - Continuous object linking 72

- Creating multiple links at the same time72
- Viewing links73
- Editing links73
- Organizing objects in a hierarchy74
- Zoom controls.....74
- Keeping all objects within view75
- Link navigation75
- About control points76
 - About point properties.....77
 - About point actions.....78
 - About point facets82
- About point extensions88
 - About the proxy extension88
 - About control extensions.....89
 - About history extensions90
- About control triggers.....92
 - How triggers are used92
 - About related objects.....92
- About point status92
 - Types of status flags93
 - Priority of status indication94
 - How status flags are set.....94
 - About "isValid" status check.....96
- About writable points.....96
 - About the priority scheme97
 - About minimum On and Off times.....98
- About composites99
 - Some composite examples99
 - Composite issues102
- Chapter 4 About Workbench tools103**
 - Types of Workbench Tools103
 - Alarm portal.....105
 - Bacnet Service107
 - Lexicon Tool107
 - Lexicon Report view107
 - Lexicon Editor view108
 - Lexicon Module Builder108
 - Lexicon Module Migrator108
 - Local License Database.....109
 - Logger Configuration tool110
 - Configuring settings for local Workbench logs.....111
 - Viewing logged data112
 - Lon Xml Tool.....112
 - Lonworks Service113
 - Credentials manager114
 - NDIO to NRIO Conversion Tool.....115

Converting NDIO modules to NRIO	115
New Driver wizard.....	118
New Module wizard	118
New Station wizard	119
Creating a new station.....	121
Creating a station template	123
Request License	125
Resource Estimator	126
Time Zone Database Tool	127
Manifest Info.....	127
Time Zone Database.....	127
Daylight Savings Time Calculator	128
Time Zone Calculator	128
Todo List.....	128
Workbench Job Service	129
Workbench Service Manager	129
Chapter 5 Component Guides	131
Components in alarm module	132
alarm-AlarmClass (DefaultAlarmClass).....	132
alarm-AlarmClassFolder.....	133
alarm-AlarmConsoleOptions.....	134
alarm-AlarmPortalOptions	135
alarm-AlarmService	135
Types of alarm extensions.....	141
alarm-AlarmSourceInfo	147
Types of alarm recipients	148
alarm-FaultAlgorithm	153
alarm-EnumChangeOfStateAlarmExt	153
alarm-EnumCommandFailureAlarmExt.....	153
alarm-MemoryAlarmService.....	153
alarm-OffnormalAlgorithm.....	154
alarm-OutOfRangeAlarmExt	154
alarm-StatusAlarmExt	154
alarm-StatusAlarmExt.....	155
alarm-StringChangeOfValueAlarmExt	155
Example	156
Components in alarmRdb module	156
alarmRdb-RdbAlarmService	157
Components in backup module.....	157
backup-BackupService.....	157
baja-FoxBackupJob.....	159
Components in baja module	159
baja-Category	160
baja-CategoryService	160
baja-Component	162
baja-DataFile.....	162

- baja-Directory 162
- baja-FileSystem 162
- baja-Folder 162
- baja-Format 162
- baja-IpHost 163
- baja-Job..... 163
- baja-JobService..... 164
- baja-LocalHost 164
- baja-Module..... 164
- baja-ModuleSpace..... 164
- baja-Permissions 164
- baja-PermissionsMap..... 164
- baja-PxView 164
- baja-ServiceContainer 165
- baja-Spy..... 165
- baja-Station 165
- baja-StationSaveJob..... 165
- baja-SyntheticModuleFile 165
- baja-UnrestrictedFolder..... 166
- baja-User 166
- baja-UserPasswordConfiguration 166
- baja-UserPrototypes..... 166
- baja-UserService 166
- baja-UserServicePasswordConfiguration 169
- baja-Vector 169
- baja-VirtualComponent 169
- baja-VirtualGateway 169
- baja-WsTextBlock..... 169
- baja-ZipFile 169
- Components in chart module 169
 - chart-BarChart 170
 - chart-ChartCanvas..... 170
 - chart-ChartHeader 170
 - chart-ChartPane 170
 - chart-DefaultChartLegend 170
 - chart-LineChart 170
- Components in control module 170
 - control-BooleanPoint..... 170
 - control-BooleanWritable..... 171
 - control-DiscreteTotalizerExt 171
 - control-EnumPoint 171
 - control-EnumWritable 171
 - control-NullProxyExt 171
 - control-NumericPoint 171
 - control-NumericTotalizerExt 171
 - control-NumericWritable 171

control-StringPoint	172
control-StringWritable	172
control-TimeTrigger	172
Components in converters module.....	172
converters-EnumToSimpleMap	172
Components in crypto module.....	172
crypto-CryptoService	172
crypto-SslProvider	172
Components in email module.....	172
email-Email	173
email-EmailAlarmAcknowledger	173
email-EmailRecipient	173
email-EmailService	173
email-IncomingAccount	173
email-OutgoingAccount	175
Components in file module	177
file-ApplicationFile	177
file-AudioFile.....	177
file-BajadocFile.....	177
file-BogFile	178
file-BogScheme	178
file-BogSpace.....	178
file-CFile	178
file-CssFile	178
file-ExcelFile.....	178
file-GifFile	178
file-HtmlFile	178
file-ImageFile	178
file-JavaFile.....	178
file-JpegFile	178
file-NavFile	178
file-PaletteFile.....	178
file-PdfFile	179
file-PngFile.....	179
file-PowerPointFile	179
file-PrintFile	179
file-PxFile.....	179
file-TextFile.....	179
file-VideoFile.....	179
file-VisioFile	179
file-WordFile	179
file-XmlFile.....	179
Components in help module	179
help-BajadocOptions.....	179
Components in history module	179
history-AuditRecord	180

- history-AuditHistoryService 180
- history-BooleanCovHistoryExt 181
- history-BooleanIntervalHistoryExt 181
- history-ConfigRule 181
- history-ConfigRules 181
- history-CovAlgorithm 181
- history-EnumCovHistoryExt 181
- history-EnumIntervalHistoryExt 181
- history-FoxHistory 181
- history-FoxHistorySpace 181
- history-HistoryConfig 182
- history-HistoryDevice 182
- history-HistoryEditorOptions 182
- history-HistoryGroup 182
- history-HistoryGroupings 182
- history-HistoryId 182
- history-HistoryPointList 182
- history-HistoryPointListItem 183
- history-HistoryService 184
- history-HistoryShortcut 185
- history-IntervalAlgorithm 185
- history-LocalDatabaseConfig 185
- history-LogHistoryService 185
- history-NumericCovAlgorithm 185
- history-NumericCovHistoryExt 185
- history-NumericIntervalHistoryExt 185
- history-StringCovHistoryExt 185
- history-StringIntervalHistoryExt 185
- Components in net module 185
 - net-InternetAddress 186
 - net-HttpProxyServer 186
- Components in onCall module 186
 - onCall-OnCallService 187
 - onCall-OnCallRecipient 187
 - onCall-OnCallSchedule 188
 - onCall-OnCallList 188
- Components in program module 189
 - program-Program 189
 - program-ProgramModule 189
 - program-ProgramService 189
- Components in the Sms module 189
 - sms-SmsService 190
 - sms-SmsRecipient 190
 - sms-Sms 190
 - sms-SmsAlarmAcknowledger 190
- Components in schedule module 190

schedule-BooleanSchedule	191
schedule-CalendarSchedule	191
schedule-EnumSchedule	191
schedule-NumericSchedule	191
schedule-StringSchedule	191
schedule-TriggerSchedule	191
schedule-BooleanScheduleSelector	192
schedule-NumericScheduleSelector	192
schedule-StringScheduleSelector	192
schedule-EnumScheduleSelector	192
Components in timesync module	192
timesync-TimeSyncClient	193
timesync-TimeSyncService	193
Components in web module	193
web-MobileClientEnvironment	193
web-WebService	193
Components in workbench module	197
workbench-ComponentFinder	198
workbench-KioskService	198
workbench-WbFieldEditorBinding	198
workbench-WbViewBinding	198
workbench-WsOptions	198
workbench-WebWidget	199
Chapter 6 Plugin Guides	201
Types of plugin modules	201
Plugins in alarm module	202
wbutil-RoleManager	202
alarm-AlarmDbMaintenance	203
alarm-AlarmDbView	203
alarm-AlarmClassSummary	203
alarm-AlarmExtManager	204
alarm-AlarmInstructionsManager	204
alarm-AlarmPortal	204
Plugins in backup module	204
backup-BackupManager	204
Plugins in chart module	205
chart-ResourceManager	205
Plugins in email module	205
email-EmailAccountManager	205
Plugins in help module	205
help-BajadocViewer	206
Plugins in history module	208
history-DatabaseMaintenance	208
history-DeviceHistoriesView	208
history-GroupManager	208
history-HistoryChart	208

- history-HistoryChartBuilder 209
- history-HistoryEditor 209
- history-HistoryExtManager 210
- history-HistorySummaryView 210
- history-HistoryTable 210
- history-LiveHistoryChart 211
- Plugins in html module 211
 - html-WbHtmlView 211
 - html-SpyViewer 216
- Plugins in onCall module 216
 - onCall-OnCallListManager 216
 - onCall-OnCallContactManager 216
 - onCall-OnCallUserReportView 217
- Plugins in program module 217
 - program-BatchEditor 217
 - program-ProgramEditor 217
 - program-ProgramModuleBuilder 220
 - program-RobotEditor 220
- Plugins in raster module 221
 - raster-RasterViewer 221
- Plugins in schedule module 221
 - schedule-CalendarScheduler 221
 - schedule-CurrentDaySummary 221
 - schedule-Scheduler 221
 - schedule-TriggerScheduler 222
- Plugins in timesync module 223
 - timesync-TimeSyncManager 223
- Plugins in wiresheet module 223
 - wiresheet-WireSheet 223
- Plugins in wbutil module 226
 - wbutil-CategoryBrowser 226
 - wbutil-CategoryManager 227
 - wbutil-CategorySheet 228
 - wbutil-PermissionsBrowser 229
 - wbutil-ResourceEstimator 230
 - wbutil-ToDoList 230
 - wbutil-UserManager 230
- Plugins in workbench module 230
 - workbench-CollectionTable 231
 - workbench-DirectoryList 231
 - workbench-HexFileEditor 232
 - workbench-JobServiceManager 232
 - workbench-LinkSheet 232
 - workbench-ModuleSpaceView 232
 - workbench-NavContainerView 233
 - workbench-NavFileEditor 233

workbench-PropertySheet	233
workbench-ServiceManager	238
workbench-SlotSheet	238
workbench-StationSummary	240
workbench-Synthetic Module File View	240
workbench-TextFileEditor	240
workbench-WbPxView.....	242
workbench-WbServiceManagerView	243
A References	245
About keyboard shortcuts	245
Types of menu bar items.....	246
About the File menu.....	247
About the Edit menu	249
About the Search menu	251
About the Bookmarks menu	252
About the Tools menu	252
About the Window menu.....	253
About the Px Editor menu	254
About the History Ext Manager menu	255
About the Help menu	255
Types of popup menu items.....	256
About the Nav side bar popup menu items.....	256
About the Wire Sheet popup menu items.....	259
About the property sheet popup menu items.....	260
About the Px Editor popup menu items.....	260
About the history extension manager popup menu items.....	262
About the Todo list popup menu items	263
About the point manager popup menu items	264
Types of side bars.....	265
About the side bar title bar.....	266
About the Bookmarks side bar.....	266
About the Help side bar	267
About Jobs side bar	268
About the Nav side bar.....	268
About the Palette side bar	271
About the Search side bar.....	273
Template side bar	274
About the Todo list sidebar	275
About the Jobs side bar.....	275
About the bound ords side bar	275
About the widget tree side bar	276
About the Px properties side bar	276
About the Properties side bar	277
Types of edit commands	277
Types of toolbar icons.....	278
Standard toolbar icons	279

- Slot Sheet toolbar icons..... 279
- Px Editor toolbar icons 280
- About the history extension manager toolbar icons 281
- About the history editor toolbar icons..... 281
- About the Todo list toolbar icons 281
- Types of console commands 282
 - Niagara shell commands..... 282
 - nre (station) commands 283
 - wb (Workbench) commands..... 284
 - plat (platform) commands..... 284

About this guide

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

Document change log

Updates to this document are listed below

- December 8, 2015: Corrections confined to the section on the NDIO to NRIO Conversion Tool.
- November 9, 2015: Niagara 4.1 release version.
Updated the chapter on Workbench Tools to include information on the NDIO to NRIO Conversion Tool.
Added Applet Module Caching Type property description in Components chapter, web-WebServices component.
- Initial release document: August 31, 2015.

Related documentation

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

Related documents

Document Title	Description
<i>AX to N4 Migration Guide</i>	Describes processes and considerations for users that want to migrate NiagaraAX stations to Niagara 4.
<i>Hierarchies Guide</i>	This document provides user information for working with the Hierarchies feature.
<i>Histories Guide</i>	This document covers the concepts of histories, history services, history components and plugins, and describes common histories-related tasks.
<i>Lexicon Guide</i>	This document explains the concepts of lexicons, and describes how to use Workbench lexicon tools.
<i>Relations Guide</i>	This document explains the concepts of entity-relationships, creating, editing, and tagging relations.
<i>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.
<i>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>Tagging Guide</i>	This document describes tags, tag dictionaries, tag groups, direct and implied tags, and other concepts along with common tagging tasks.

Document Title	Description
<i>Templates Guide</i>	This document explains template concepts and includes common template tasks.
<i>Web Charts Guide</i>	This document describes and illustrates use of web charting tools.
<i>Niagara 4.0 Installation Guide</i>	This document describes how to install Niagara 4 on various platforms.
<i>Niagara 4 Platform Guide</i>	Describes Workbench views that are available when you have a platform connection to a Niagara 4 host. It also describes PlatformServices that are available in a Niagara 4 station.
<i>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 a 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 Niagara 4 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

Software frameworks provide a platform to allow businesses to more easily build their end-product offerings. Tridium's 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 the fourth generation of Tridium's Niagara Framework. This UX framework provides an infrastructure which 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 "queryable" tags, the functionality on which many of the new features (search, tagging, relations, templates, hierarchies) are based. 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

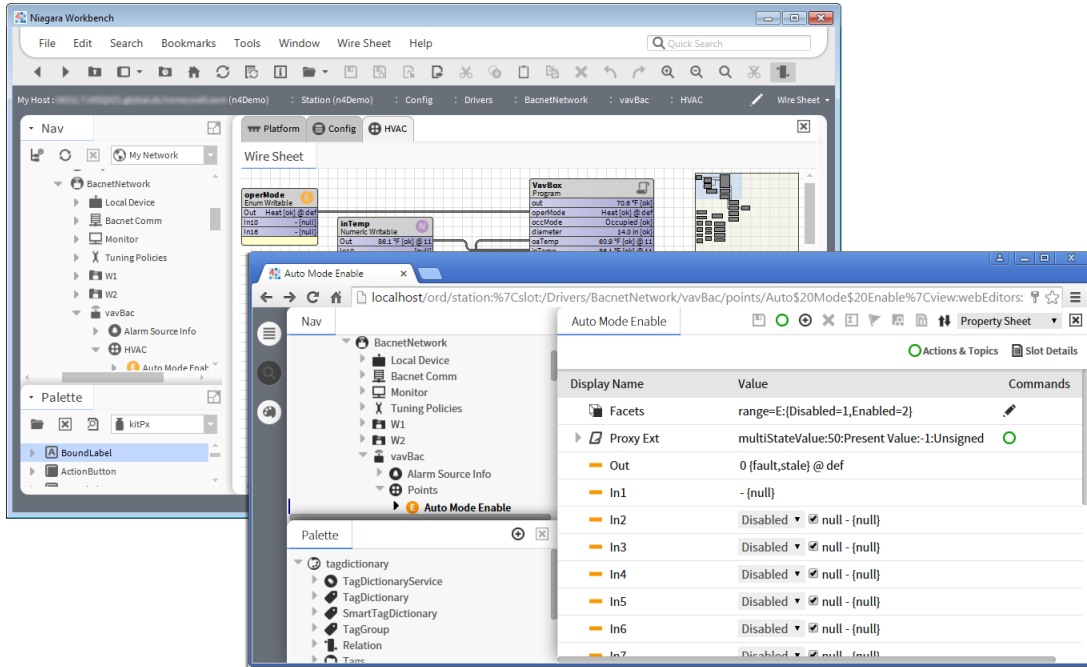
The user experience (UX) is at the heart of Niagara 4 . The Niagara 4 "bajaux" framework is based on open web technologies, such as HTML5, CSS3, JSON, and Bajascript v2.0. The bajaux framework effectively provides an improved user experience in utility, ease of use, and efficiency, as well as the capability 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 color to emphasize important information.

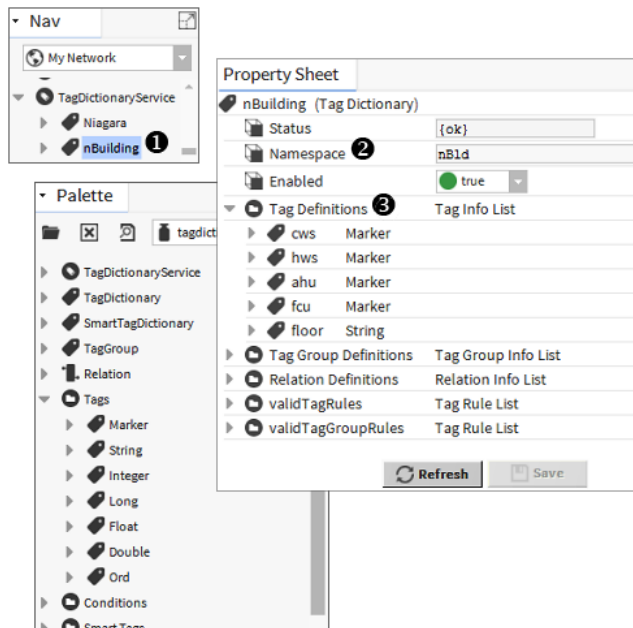
Figure 1 Workbench interface (left) compared with web browser view(right)



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.

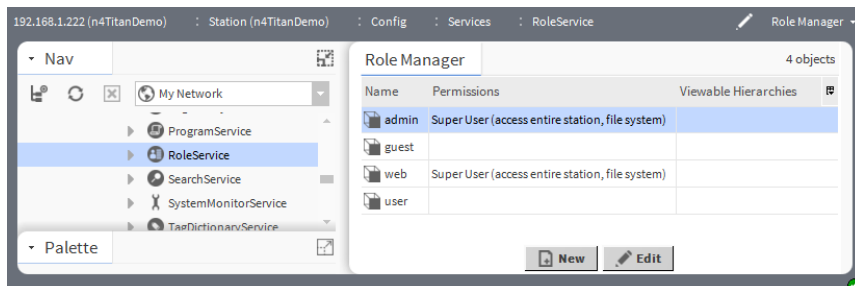
Figure 2 Custom tag dictionary (1), tag definitions in Property Sheet (3), and tagdictionary palette (lower left)



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:

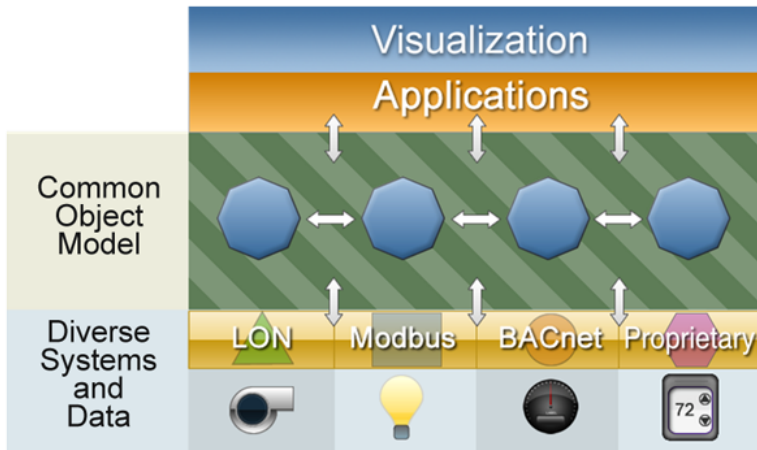
1. connecting devices on a common communications media
2. modeling those devices in software
3. programming applications to use the information in those devices

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, Tridium has developed a reference design processor core. Known as the Niagara Processor Module (NPM), this platform can be licensed from Tridium 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 Framework 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 Niagara 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 Niagarasoftware subsystems and the software processes and protocols, respectively.

Figure 5 Niagara software subsystems

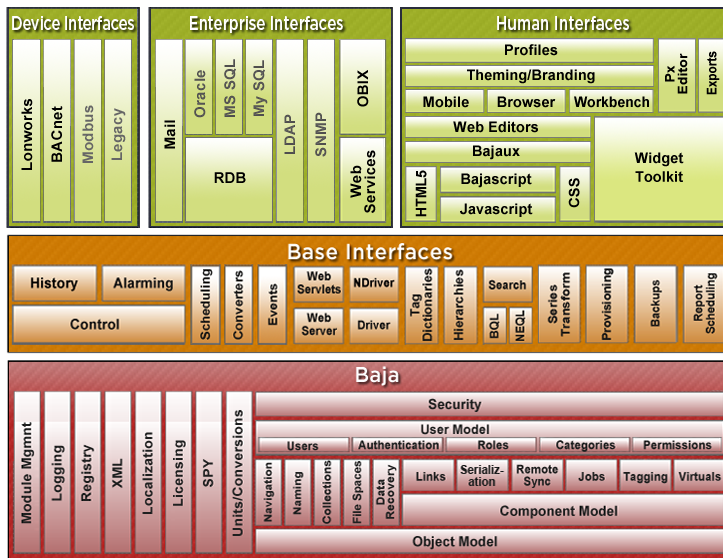
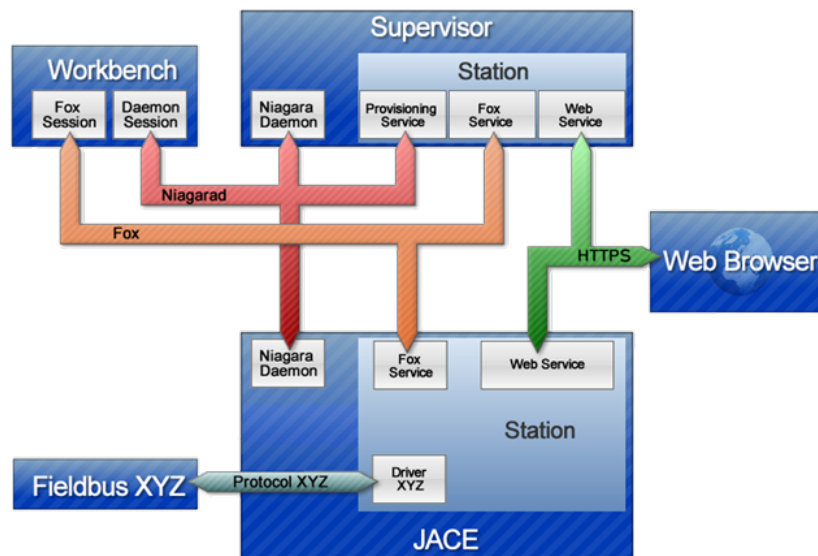


Figure 6 Niagara software processes and protocols



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 Niagara.

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. Niagara is an implementation of that specification. Furthermore you will find a vast number of features in Niagara, that are not included under the Baja umbrella. In this respect Niagara 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 — and using the Workbench API you can 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” for more about information) are all grouped under `javax.baja`. These are APIs that will be 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 developed by Tridium 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: Tridium has developed some APIs under `javax.baja` even though they are not currently part of the Baja specification. These are APIs that 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 Niagara 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.

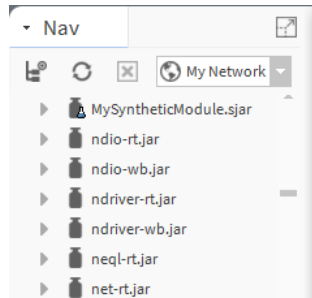
Major releases of the Niagara software are distributed along with a set of release modules, but as new modules are made available for that release, they may be distributed as independent revisions within that release.

Two types of modules are possible in the framework which make up the Niagara software, standard and synthetic modules. Within the standard (`.jar`) module type there are a number of subtypes based on applicable runtime profile. For example `moduleName-rt.jar`, `moduleName-wb.jar`, etc. The second type is the “synthetic Java archive” (`.sjar`) module which allows for the creation of memory-resident modules and types programmatically at run-time. Although they share many characteristics, the two types of modules are

distinctly different. This section describes standard modules simply called “modules.” For details on synthetic modules, refer to the “*NiagaraAX Synthetic Modules*” engineering notes document.

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. Refer to “About components” for more information about components.

Module characteristics

All (.jar) modules are composed of a single Java ARchive (JAR) file compliant with PKZIP compression. Modules contain an XML manifest, and are independently versioned and deployable. Modules state their dependencies on other modules and their versions.

About jar files

JAR files are the mechanism for distributing Java modules. A JAR file (.jar) 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 - it will not work. JAR files are the add-ins that are plugged into the software to give additional functionality. Inside the JAR file are the compressed software; its documentation; in some cases, examples or pre-canned applications; and libraries.

Within the standard (.jar) module type there are a number of subtypes which are based on the applicable runtime profile (bajaux, doc, runtime):

- `moduleName-rt.jar` indicates a runtime module
- `moduleName-wb.jar` indicates a 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 or module is versioned independently.

About module versions

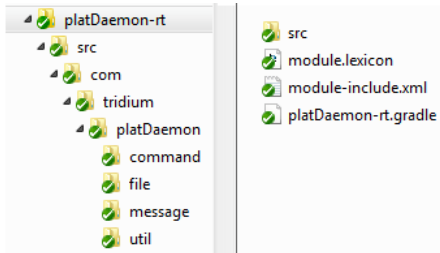
Niagara’s versioning system allows you to check to make sure that you have the latest available module for your system. Versions are specified as a series of whole numbers separated by a period, for example “4.0.16”. To understand which version of a module is more recent, simply observe the module version number. For example, 4.0.16 is later 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 will be zipped up for the module's jar file.

Figure 8 Directory structure used to build the module's jar



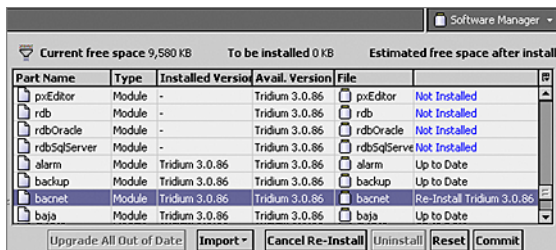
Benefits of modular software development

Modular software development provides several benefits, including the following:

- Improves tracking deployment and versioning

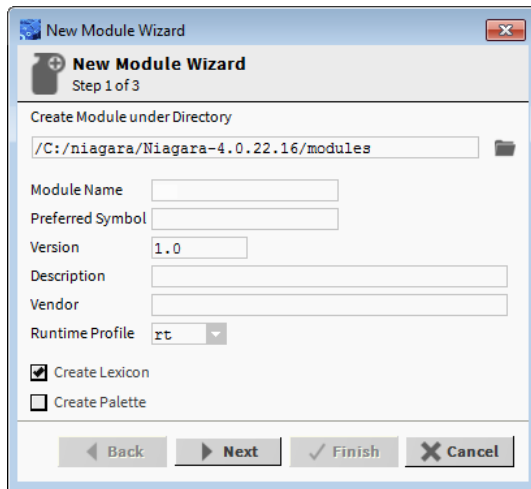
Modular development assists in tracking the deployment and versioning of Niagara features and releases. For example, if a new driver is available to be added to Niagara Framework, it can be packaged, delivered, and added in as a single module or with multiple modules, using the platform **Software Manager** view. Refer to the *Niagara 4 Platform Guide* section "Software Manager" for more information about managing modules.

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, if want to add additional modules you can do so 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.
- Create 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



NOTE: You can also create “synthetic modules”. Refer to the engineering notes document, *NiagaraAX Synthetic Modules* for details.

About components

A component is the primary building block that you use to engineer an application using Niagara 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 compose a Niagara implementation whereas modules compose the Niagara 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	Name	Flags	Type	Facets
Facets	facets	(none)	baja:Facets	(none)
Proxy Ext	proxyExt	(none)	control:NullProxyExt	(none)
Out	out	rtso	baja:StatusNumeric	(none)
In1	in1	r	baja:StatusNumeric	(none)
In2	in2	t	baja:StatusNumeric	(none)
In3	in3	t	baja:StatusNumeric	(none)
In4	in4	t	baja:StatusNumeric	(none)
In5	in5	t	baja:StatusNumeric	(none)
In6	in6	t	baja:StatusNumeric	(none)
In7	in7	t	baja:StatusNumeric	(none)

- 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 is 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 in Niagara, 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.

For detailed information about points and point extensions, refer to "Data and Control Model".

About component naming

In a Niagara 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 on "Escaped names," which covers using ASCII code in component names.
- The first character in the name must be a letter (not a numeral).
- Name must be unique for every component within the same parent component. Workbench automatically enforces this rule, via a popup error dialog.
- 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. For more information about using the palette side bar, refer to “About the palette side bar”.

Escaped names

Workbench allows you to name components “improperly,” such as with spaces or other non-alphanumeric characters, without any warning. Further, various Niagara 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%” that 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, will have 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 using rules—see the section, “About component naming”). 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:

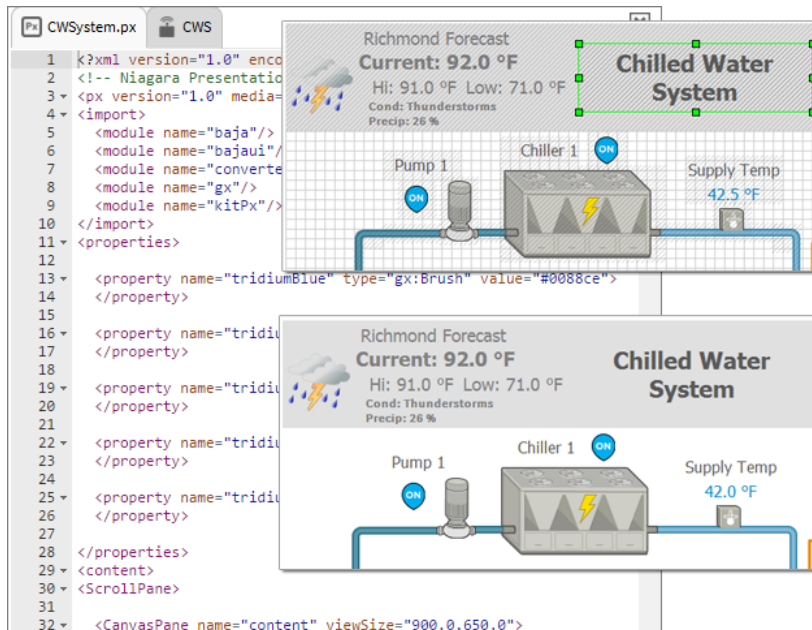
- information
 - real-time data
 - historical data
 - configuration data
 - alarm data
 - scheduling data

- 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 a 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

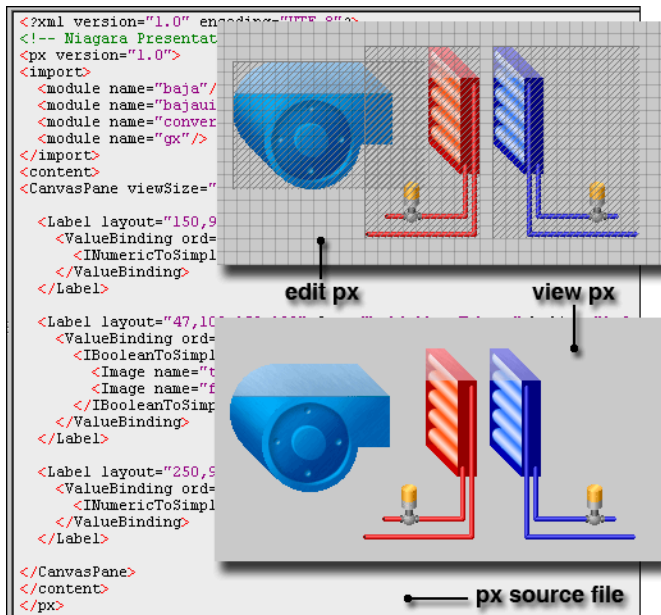
Niagara's presentation architecture is based on many modules and their associated public APIs:

- **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.
- **bajai** — the bajai module provides the widget toolkit. Widgets are the basis for graphical composition, layout, user input, and data binding. The bajai 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 allows 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 a 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 a 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

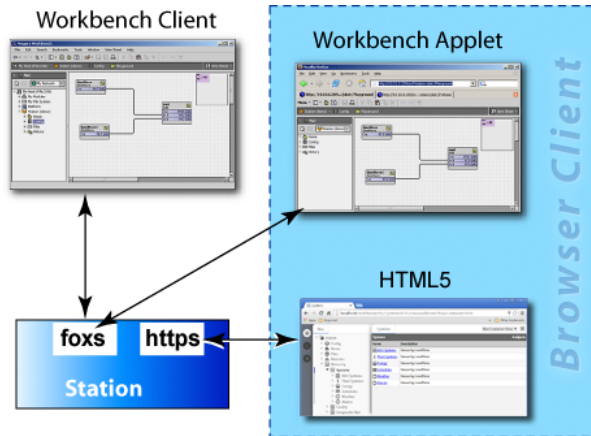
Niagara supports four primary target media.

- **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.
- **Workbench applet** — the applet allows the full Niagara stack to be downloaded to a client machine using a web browser. Once loaded to the client, you can run Workbench as an applet within an HTML page. In this scenario, the presentation engine will be available with the full feature set. The WbProfile feature may be used to customize how Workbench is decorated inside the HTML page.
- **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 tool bar or in a modern HTML5-capable web browser.

Refer to the *Graphics Guide* section “Types of Px target media options” for information about presentation media technologies.

Figure 14 Presentation media options



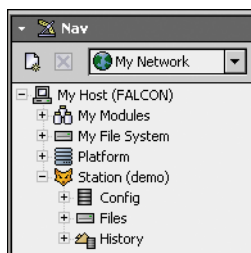
About stations

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 a JACE controller.

Figure 15 Station in nav tree

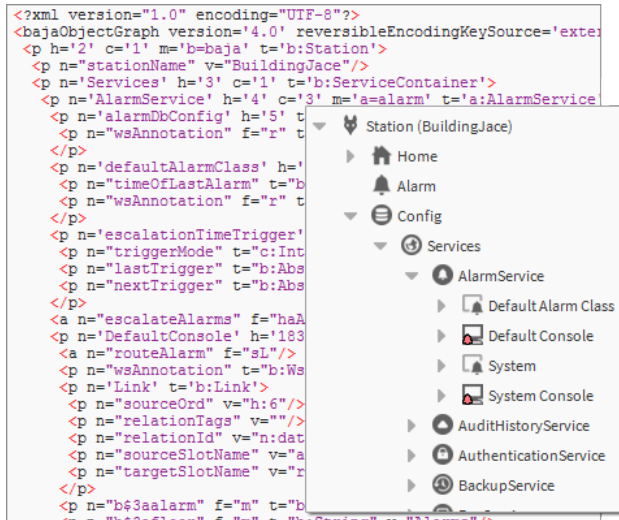


A system can be a single station or multiple stations depending on the size of the project and it is defined by a bog file. See “About BOG files”.

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

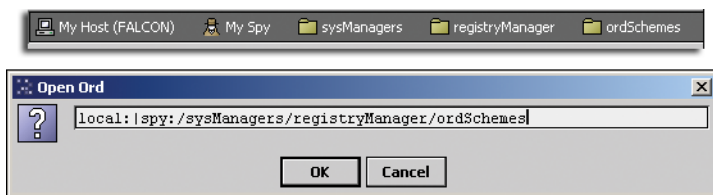


About ORDs

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** dialog box.

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

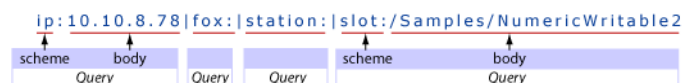
- `ip:somehost|fox:|station:|slot:/MyService`
- `ip:somehost|fox:|station:|h:/42`
- `ip:somehost|fox:|file:/C:/dir/file.txt`
- `local:|file:!jre/lib/logging.properties`
- `local:|module://icons-ux/x16/cloud.png`
- `local:|spy:/`

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.

Types of space

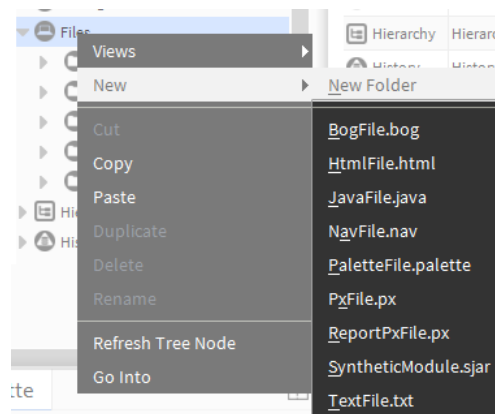
Space defines a group of objects that share common strategies for loading, caching, lifecycle, naming, and navigation. Following, is a list of some of the different types of space:

- 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. Refer to “About BOG files” for more information about this type of file.
- **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* section “About the Nav file” for more information about nav files.
NOTE: Also see “About file naming”.
- **PaletteFile.palette**
These files are custom collections of components that you create and save for viewing in the palette side bar. Refer to “To create a palette” for information about creating custom palettes.
- **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. Refer to “About presentation xml (Px)” for more information about Px files.
- **ReportPxFile.px**
The ReportPxFile is a regular Px file that has been “pre-loaded” with a ReportPane widget. The reporting functionality in helps you design, display, and deliver data to online views, web browser, and various other formats.
- **SyntheticModule.sjar**
The “synthetic Java archive” (.sjar) module allows you to create memory-resident modules and types programmatically at run-time. This type of module are distinctly different from standard modules . For details on synthetic modules, refer to the “*NiagaraAX Synthetic Modules*” engineering notes document.
- **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).

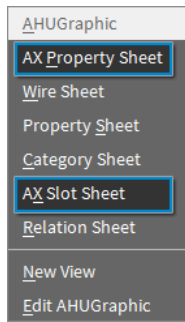
About views

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.

See “About the view pane” for more information on the view pane. Refer to the “Plugin Guides” for a comprehensive list of views.

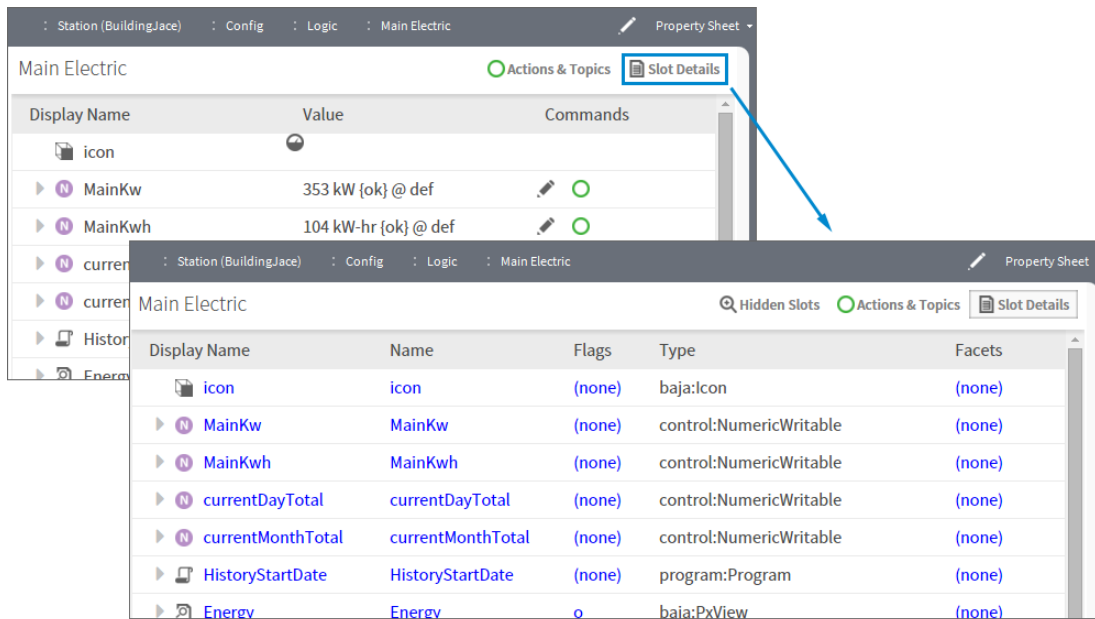
Two AX views are available under the **Views** selector menu and the right-click pop-up menu (as shown here): the **AX Property Sheet** view, and **AX Slot Sheet** view.

Figure 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



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 4 Lexicon Guide*.

Chapter 2 About Workbench

Topics covered in this chapter

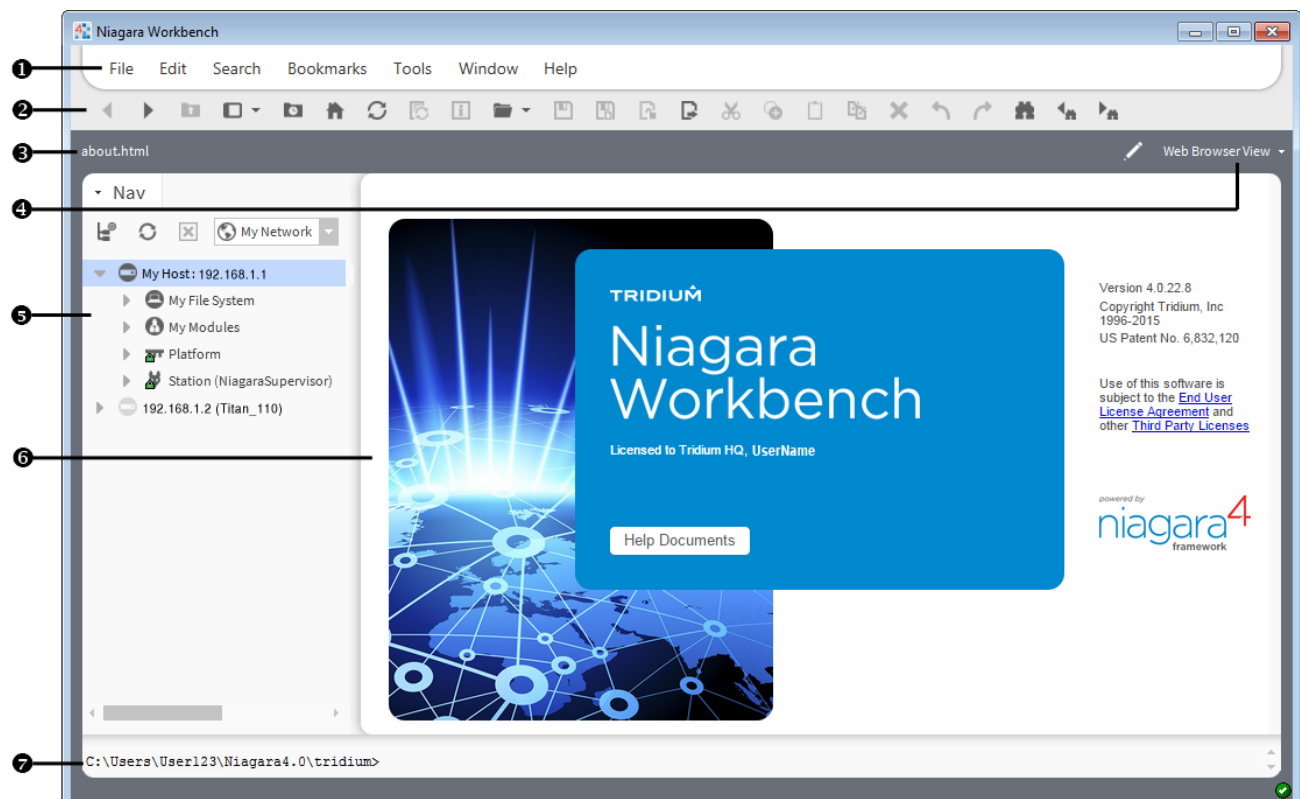
- ◆ Tour of the Workbench GUI
- ◆ Workbench window controls
- ◆ About the side bar panes
- ◆ About popup menus
- ◆ Table controls and options
- ◆ Controls and options for charts
- ◆ Customizing the Workbench environment

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 Workbench welcome screen opens.

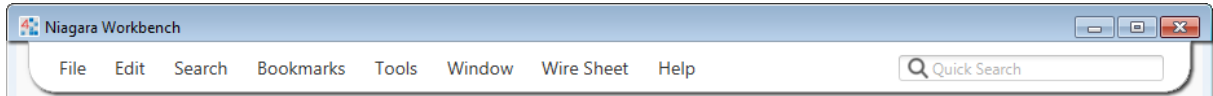
Figure 23 Niagara Workbench



The primary window of the Workbench interface is divided into seven areas:

- 1 Menu bar — contains available program menus.

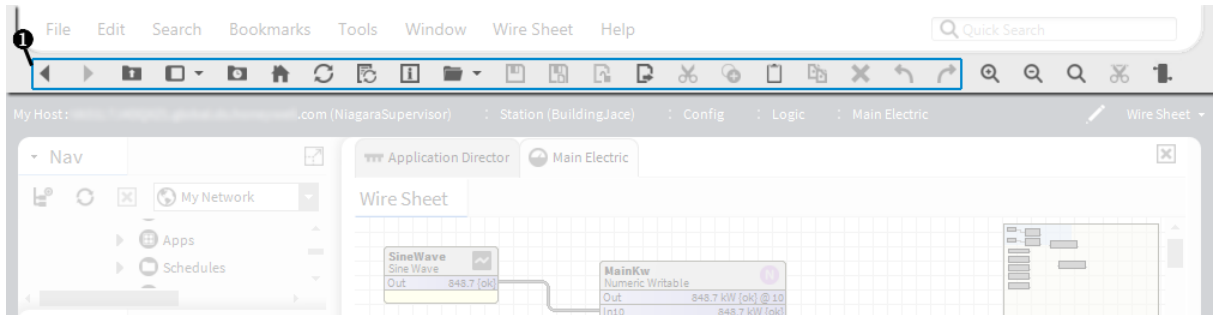
Figure 24 Menu bar



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 25 Toolbar

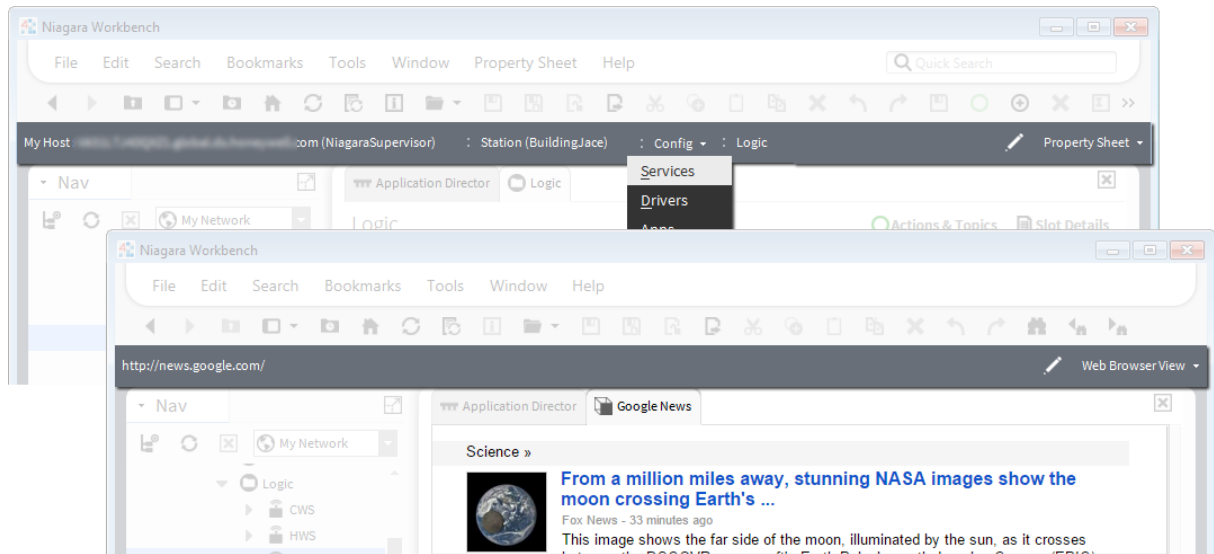


The primary icons (1 in the above image) are always visible, Additional sets of icons are added to the toolbar when you select certain views. For example, when the **Wire Sheet** view is active, the **Delete Links** icon and **Zoom** icons are available.

When an icon is dimmed, it is unavailable. Hovering the mouse pointer over a toolbar icon invokes a popup 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 26 Path bar



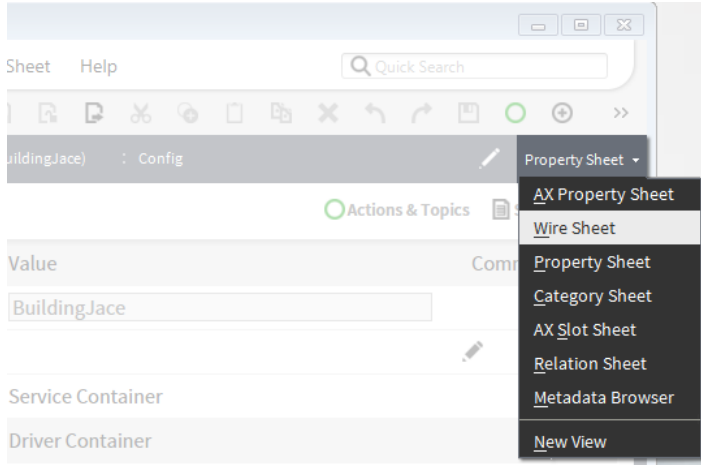
The left side of the path bar shows your current location (Ord or web address). The **View** selector appears on the right side.

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

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

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

Figure 27 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. For details, see "About the side bar panes".

- 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 tabbed views and a thumbnail view.

You may add multiple tabs to the pane by using the tab feature. To change the selected view do any of the following:

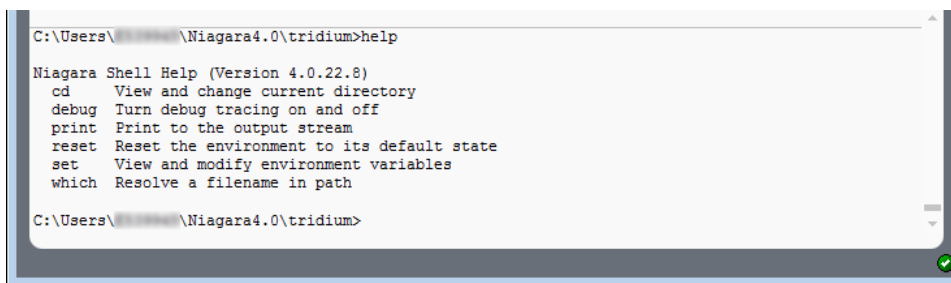
- Double-click on an item in the Nav tree.
- Select a view or action from a Nav tree palette popup menu.
- Select an option from a menu or submenu.
- Select an option from the locator bar.

The thumbnail view, when active, appears in the top right corner of the Wire Sheet window. It helps you find your way around the wire sheet.

- 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 28 Console



```

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

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

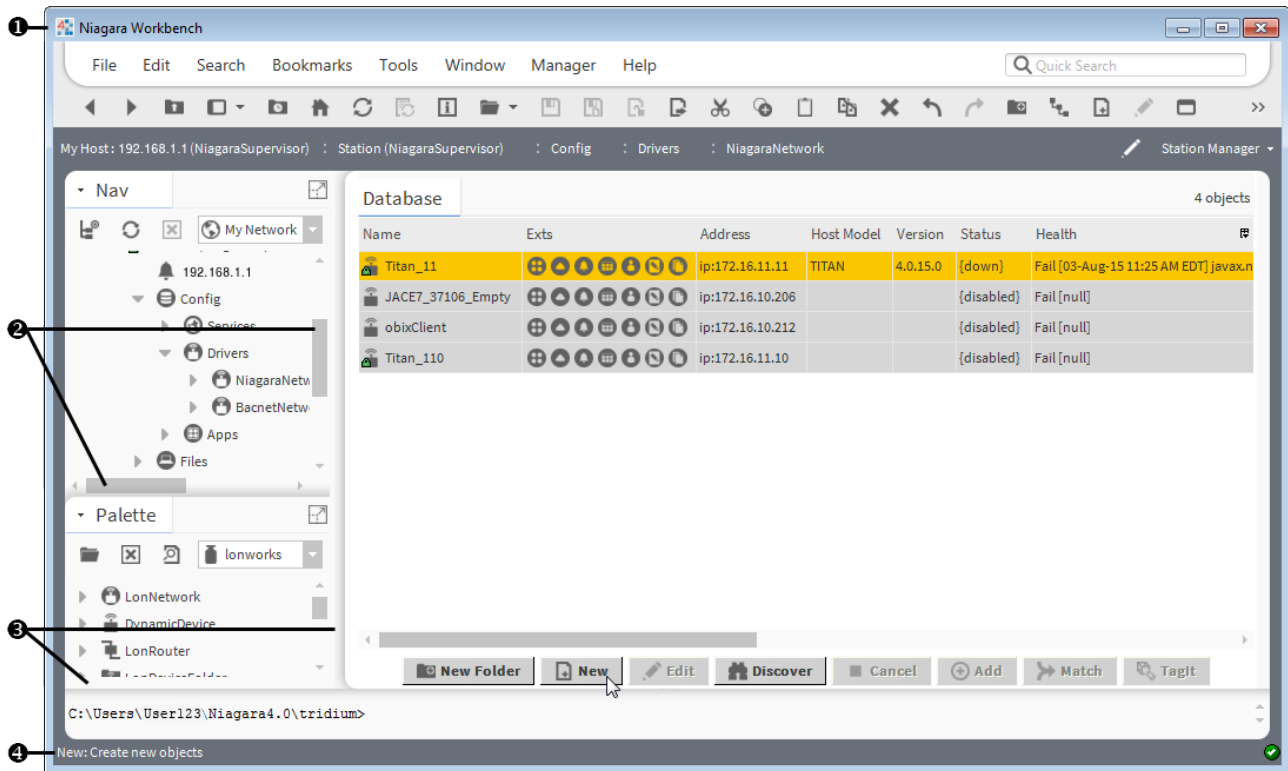
C:\Users\... \Niagara4.0\tridium>
  
```

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

Workbench window controls

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

Figure 29 Window controls

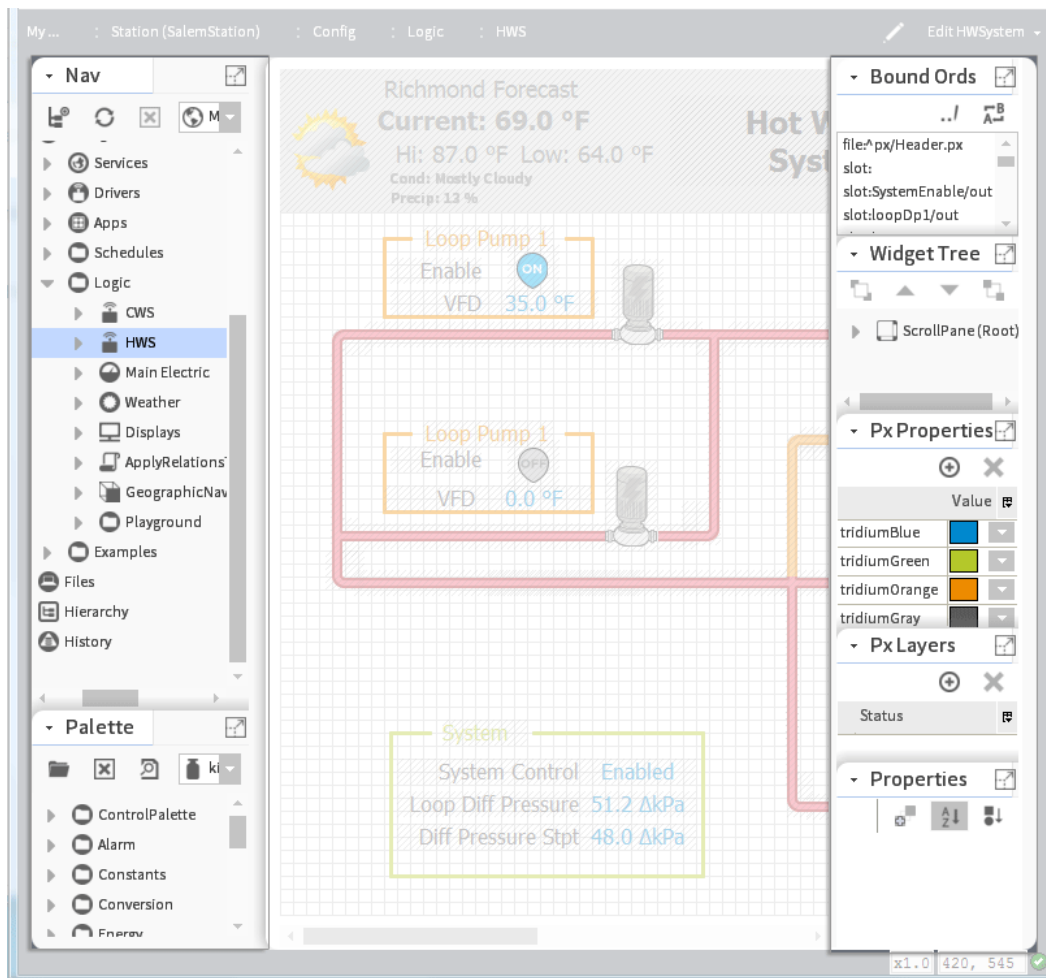


- 1 Title bar — has standard Windows title bar features, including windows name and icons to minimize, maximize, and close. Double-click the title bar to toggle between maximized and a sizable window. You may create additional windows after starting Workbench—all have these basic features:
- 2 Scroll bars — appear in window and window pane areas (side bar, view, or console) when some content portions are not visible. They are along the right and/or bottom portions of a window or pane. Simply drag a scroll slider (highlighted area) to scroll quickly. Or, click an ending scroll arrow to move in incrementally.
- 3 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.
- 4 Status bar — at the bottom left of the Workbench window is a status bar. 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.

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 30 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.

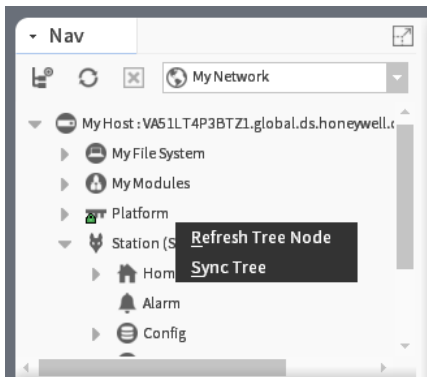
You use a popup menu to perform all operations that are available from inside the side bars (such as cutting or pasting). You use the icons in the title bar to open, close, and expand/restore the side bars. To resize side bar height and width, click and drag on the borders.

- For more information about the side bar title bar, see "About the side bar title bar".
- For more information about types of side bars see "Types of side bars".

About popup menus

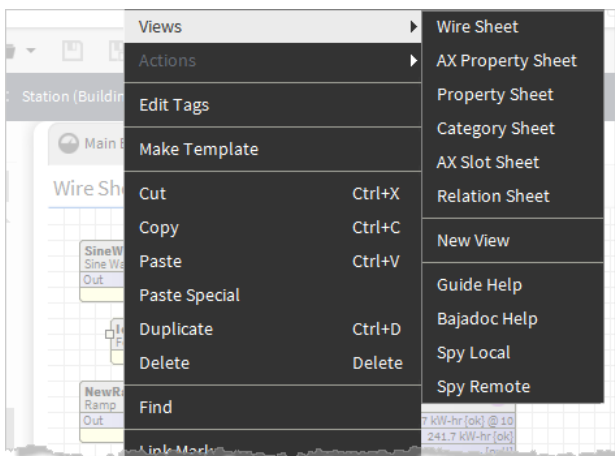
Workbench provides context-specific popup menus for controlling system options. Right-clicking on a component or a view opens these popup menus. The contents of each menu vary depending on the context.

Figure 31 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 32 Wire sheet popup menu



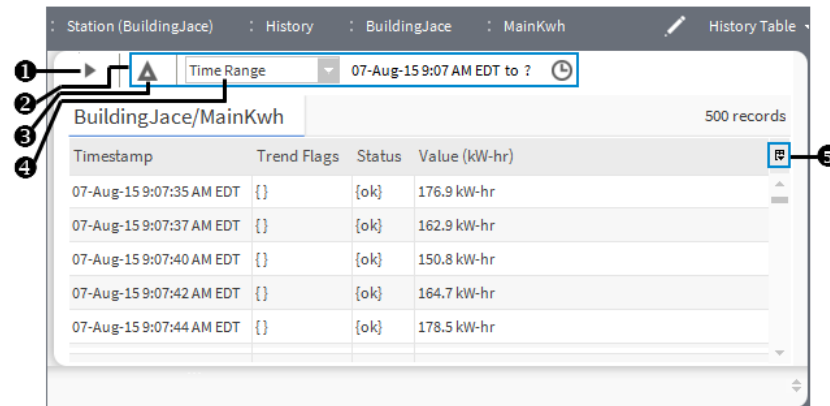
Right-clicking on the Wire sheet displays a popup menu with different options.

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

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

Figure 33 Table controls and options



1	Live Updates — The “play” icon, available for the History Table view, starts Live Updates (On Demand) updating. The icon changes to a “pause” icon while Live Updates is active. See also, “About On Demand history views”.
2	Data parameters — These controls include Delta (for history logging) and Time Range settings.
3	Delta — Useful for history logging, displays value changes (delta) in your table. Refer to “About delta logging” for information about delta logging.
4	Time range — The dropdown option list has a variety of predefined time range options, including an option that allows you to restrict your data presentation to a particular date and time range that you specify.
	Title bar — Displays the name of the data collection on the left side of the title bar and in some tables (collection table, history table, alarm extension manager, and others) displays the total number of records in the table on the right side of the title bar.
	Column headings — Each column of data has a title that indicates the data type.
	Column boundaries — Each column has a movable column boundary that can be used to re-size the column using the mouse control. Stretch or shrink column width by dragging the column boundary, as desired. Use the Reset column widths menu item to reset all column widths to their default size.
5	Table Options — Located in the upper right corner of the table, the dropdown list provides one or more of the following controls and options: <ul style="list-style-type: none"> • Reset column widths — sets all columns in the table to their default widths. This is useful if you manually changed widths of columns, and now some contents are hidden (even after scrolling). • Export — opens the Export dialog box where you can choose to export the table to PDF, text, HTML, or CSV (comma separated variable). • Context-sensitive menu items — additional context-sensitive menu items appear depending on the component that you are viewing.

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.

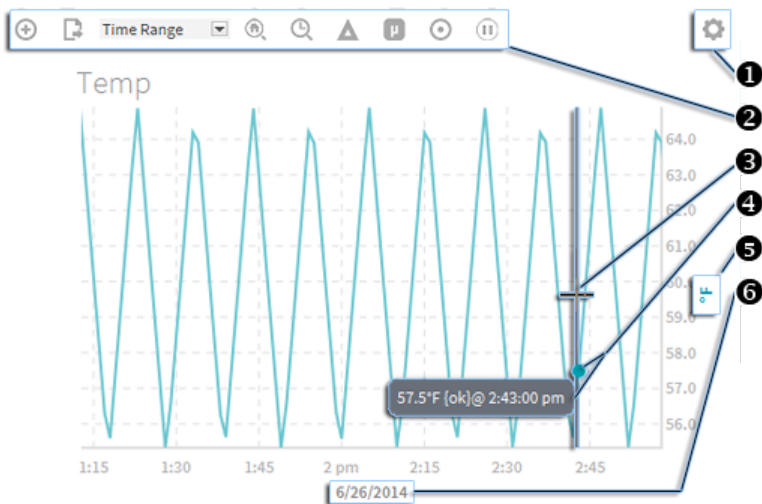
Controls and options for charts

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 in an earlier release. Although the two views have a different look and feel, both offer many of the same controls and options.

Chart view

The **Chart** view (shown below) is the default 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 34 Chart view description



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

Data that can be rendered in a chart includes historical data, live historical data, live data, as well as schedules.

Although chart type is configurable via the **Settings**→**Series** tab dialog, the default chart type is determined by the type of data being presented. For example:

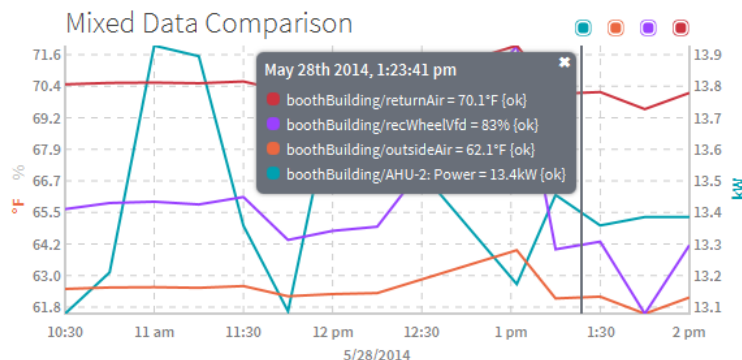
Component type	Default chart type
Numeric histories and points	Render as lines with interpolation, display as a line chart.
Numeric schedules	Render as discrete lines with no interpolation, display as a line chart.
Boolean and Enum points	Render as shaded areas referred to as "swim lanes," displayed as a shaded chart. Opacity of the swim lanes fill is based on the ordinal of the Enum.
Boolean and Enum schedules	Render as shaded areas referred to as "swim lanes," displayed as a shaded chart. Opacity of the swim lanes fill is based on the ordinal of the Enum.

Different types of data (Numeric and Boolean or Enum) can be combined on the same chart. In that situation, the swim lanes representing Boolean and Enum data display with a dimmed opacity to allow you to more clearly view the lines representing the numeric data. 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 make modifications while a chart is rendering. For example, while viewing a chart you can add one or more points, history records, schedules, or 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. On a chart containing data with three or more different units of measure, such as that shown below, the display still shows dual Y-axes. You can switch the units displayed on the secondary Y-axis by clicking on the dimmed axis label. For example, on the left-side Y-axis in the figure below, 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 percent.

You can alternately hide or show specific data or even completely remove data from a chart. Additionally, chart settings permit you to customize the appearance of a chart via selectable data colors and chart type per component, axis orientation, data source zooming, as well as permitting you to turn on or off the chart grid, background color, data popups, and status colors.

Figure 35 Multiple Y-axes accommodate data with different units of measure



Web charts utilize standard Niagara status colors to indicate current status. As shown in the chart below where the Status Coloring command is invoked, a red dot indicating `Alarm` status marks each plot in the `Ramp` line while an orange dot indicating `Fault` status marks each plot in the `FaultHistory` line. Also, status colors shown in the Fixed Data Pop-up dialog confirm the status of charted data.

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

Figure 36 Line chart displaying status colors

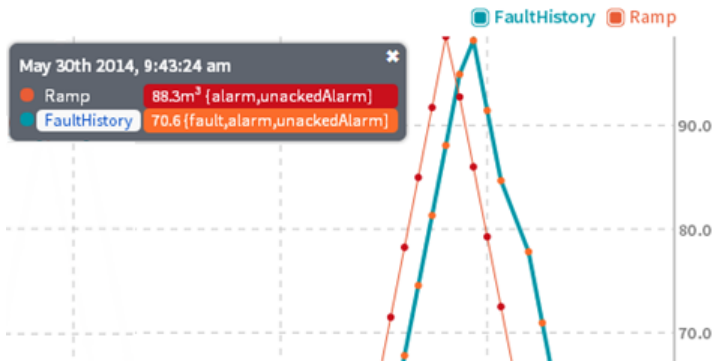


Chart commands





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

Figure 37 Command Bar



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

Command Bar	Options	Description
Add Series		Add components to the chart. Select one or more components via File Chooser. Use Ctrl + Click to select multiple individual components or select a folder that contains multiple components, as shown here. [Either the href or the keyref attribute should be set on graphic elements]
Save		Available only when you open an existing .chart file and make changes. Save — saves file (chart or csv format) to Station space: Files/charts/chart-Name.chart or Files/csv/chartName.csv
Export	Actions tab <ul style="list-style-type: none"> • File Name • Destination • File Type Options tab <ul style="list-style-type: none"> • View On Export • Status Column 	Available in a new chart and when you open an existing chart file. <ul style="list-style-type: none"> • File Name — componentName (default) or type other name using normal file-naming conventions • Destination (Workbench) — StationExports file to station File space Files/charts/chartName.chart or Files/csv/chartName.csv • Destination (Web Browser) —

Command Bar	Options	Description
		<p>Download — exports file to operating system user's file space, for example: C:\Users\userName\Downloads\chartName.chart</p> <p>Station— Exports file to station File space. For example, Files/charts/chartName.chart</p> <p>Print — launches browser Print dialog. Optionally, you can scroll to the bottom of the print dialog and click the link to "Print using system dialog"</p> <ul style="list-style-type: none"> • File Type — chart (default) or csv • View On Export — Displays exported file • Status Columns — Available only for csv exports, exported file includes status data
 Time Range	<ul style="list-style-type: none"> • Auto (default) • Time Range • Today • Last 24 Hours • Yesterday • Week To Date • Last Week • Last 7 Days • Month To Date • Last Month • Year To Date • Last Year 	<p>Specifies time range for data display. Selecting the Time Range option launches a dialog where you can enter custom Start and End times for the range. Leave the End time field blank for live data to continue plotting on the chart.</p>
 Home Zoom	<ul style="list-style-type: none"> • On (default), • Off 	<p>Turns On/Off Home Zoom.</p> <p>On — zooms to the X-axis of the primary data set.</p> <p>NOTE: If the primary data set is numeric, it zooms on the Y-axis.</p> <p>Off —</p>
 Time Zoom	<ul style="list-style-type: none"> • On, • Off (default) 	<p>Turns On/Off Time Zoom.</p> <p>On — zooms X-axis to the time period specified by Time Range dropdown list.</p> <p>Off — reverts to Home Zoom.</p>
 Delta	<ul style="list-style-type: none"> • On , • Off (default) 	<p>Turns On/Off Delta</p> <p>On — plots the rate of change between points.</p>





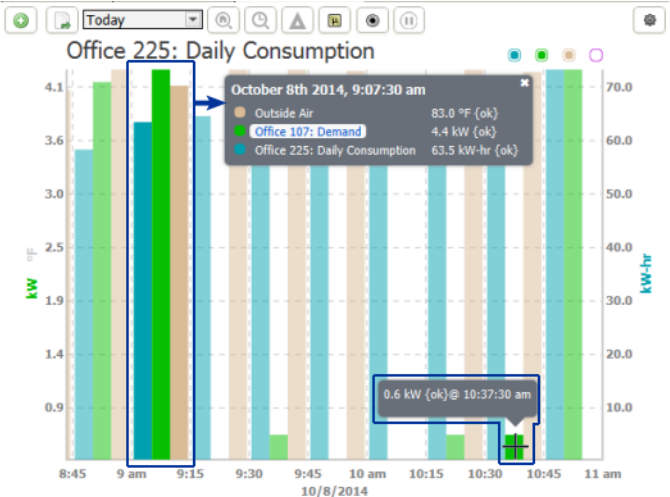
Command Bar	Options	Description
		Off — resumes plotting data points.
 Sampling	<ul style="list-style-type: none"> On , Off (default) 	<p>Turns On/Off Sampling.</p> <p>On — sampling is enabled</p> <p>Off — turns off sampling and disables auto-sampling behavior.</p>
 Status Coloring	<ul style="list-style-type: none"> On Off (default) 	<p>Turns On/Off data Status Coloring.</p> <p>On — displays data points with status colors in a line chart and in shade or bar chart displays a status color band at the top of each bar.</p> <p>Off — hides status coloring, data points/color bands.</p>
 Pause	<ul style="list-style-type: none"> On / Off (default) 	<p>Turns On/Off pause in live data plotting.</p> <p>On — pauses live data plotting. No longer in live mode when paused</p> <p>Off — resumes live data plotting</p>
 Stop	<ul style="list-style-type: none"> On / Off (default) 	<p>Added in Niagara 4.1, this becomes visible only during data loading. Turns data chunking On/Off.</p> <p>On — stops the data chunking process, halts data coming from the server. While stopped, the button displays a red border.</p> <p>Off — reloads all of the data.</p>

Chart settings

Options in the Chart view **Settings** dialog 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.

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"> Line, Discrete line, Shade, Bar 	<p>Line — plots a smooth line with interpolation. The default chart type for Numeric points and histories.</p> <p>Discrete line — plots a “stepped” line without interpolation.</p> <p>Shade — plots shaded areas, known as “swim lanes,” representing state change. The default chart type for Boolean and Enum points.</p> <p>Bar — plots vertical bars. Samples data into common intervals based on available width, When you have more than one component in a chart using bar chart type, they become a Bar Group, where the individual bars are adjacent (no space between).</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> 

Axis tab

Settings	Options	Description
Y-Axis Orientation	<ul style="list-style-type: none"> left right (default) 	Aligns Y-axis of primary data set to the left or right side of the chart.
Data Zoom Scope	<ul style="list-style-type: none"> primary (default) all 	<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>

Settings	Options	Description
Show Grid	<ul style="list-style-type: none"> true (default) false 	Turns on/off the chart grid. true — grid displays in chart behind data. false — grid does not display.
Background Area Color	<ul style="list-style-type: none"> on off (default) 	Turns on/off the Background Area Color for the current theme. On — background area color displays in chart behind data. Off — background area color does not display.

Layers tab

Settings	Options	Description
Data Popup	<ul style="list-style-type: none"> On (default)Displays OffPauses 	Enables/disables the Fixed Data popup. On — clicking on chart data displays the recorded date and time, as well as the name, value and status for each component in the chart at the point where you click. The persistent data popup remains visible until you close it. Off — suspends display of fixed data popup.
Data Mouseover	<ul style="list-style-type: none"> On (default) Off 	Enables/disables the mouseover Data Value popup. On — mouse position on chart data displays the recorded component value, status, and the time for that mouse position. Off — suspends display of mouseover data value popup.
Status Coloring	<ul style="list-style-type: none"> On Off (default) 	Turns On/Off data status coloring. On — displays data points with status colors in a line chart and in a bar chart displays a status color band at the top of each bar. Off — hides status color data points/color bands.

Sampling tab

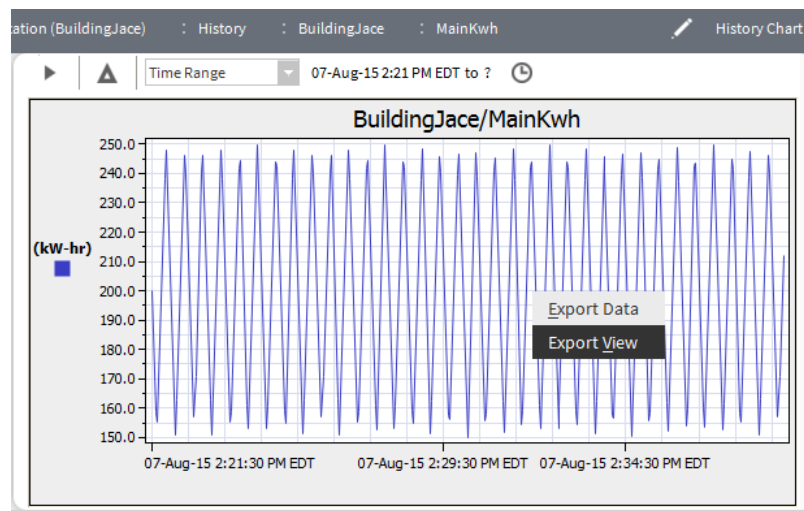
Settings	Options	Description
Auto Sampling	<ul style="list-style-type: none"> true (default) false 	Enables/disables automatic sampling optimizations. true — automatically begins sampling if the focused data set exceeds 2500. false — automatically stops sampling if the focused data set is below 2500.
Sampling	<ul style="list-style-type: none"> true false (default) 	Enables/disables sampling for any size data set. true — turns on sampling false — turns off sampling NOTE: For performance reasons, sampling cannot be turned off once the focused data set exceeds 50,000. This threshold is configurable in the <code>system.properties</code> file. For details, see "About sampling".
Sampling Type	<ul style="list-style-type: none"> average (default) min max sum 	average - samples average values for the selected rollup period. min - samples minimum values for the selected rollup period. max - samples maximum values for the selected rollup period. sum - samples the total of the values in the selected rollup period.
Sample Size	2500 (default)	Specifies the number of points in the data set to sample.


Settings	Options	Description
		NOTE: The default auto sampling size is configurable in the <code>system.properties</code> file. For details, see "About sampling".
Available Data Points	Read only	Displays the maximum number of points in the data set that are available to sample.
Sampling Period	Read only	Visible only once sampling has begun, displays the calculated average of the amount of time between each of the points that have been sampled.


History Chart view


Charts created in an 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 38 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. Refer to "About delta logging" for information about delta logging.
 -  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. See also, "About On Demand history views".
- 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

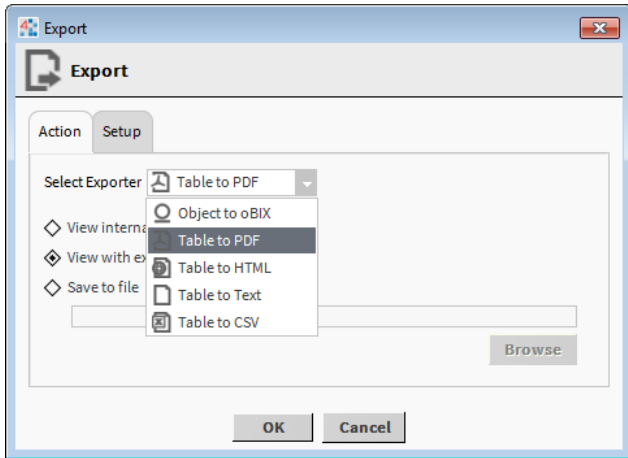
The color of the line and type of line is editable in the Chart Builder view.

- **Export** popup menu

Right-clicking on the History Chart invokes the **Export** popup menu which contains two options:

Export Data opens the **Export** dialog box where you can choose to export the chart to oBIX, PDF, Text, HTML, or CSV (comma separated variable), as shown here.

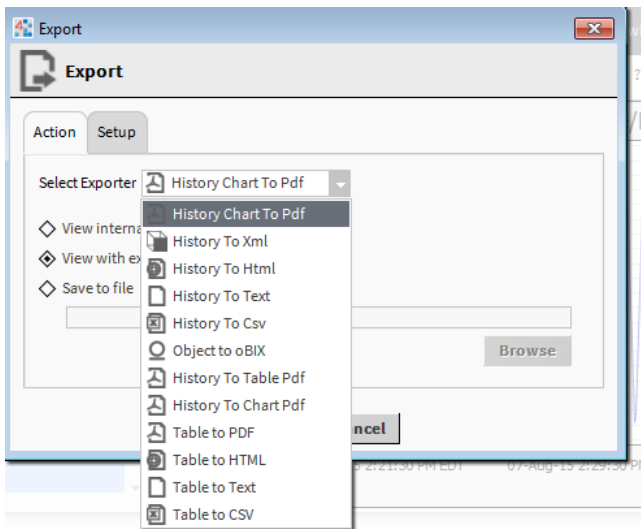
Figure 39 Export Data options



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** dialog box where you can choose from a number of export options, as shown here.

Figure 40 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.

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.

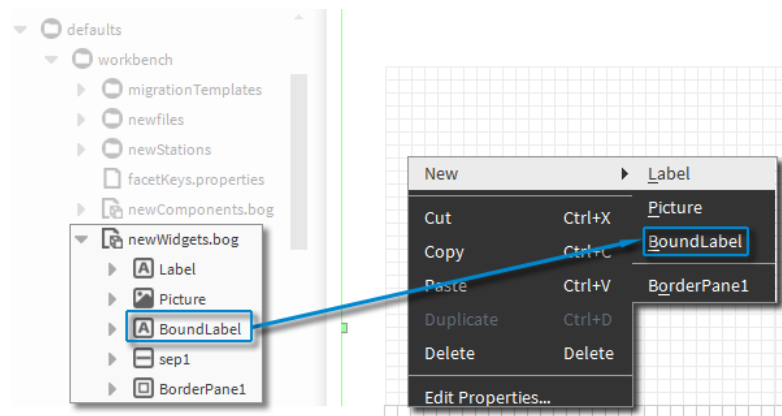
After you create multiple windows, you can then customize each individual window, as needed, to access different information, allowing you to see multiple concurrent views.

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.



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.

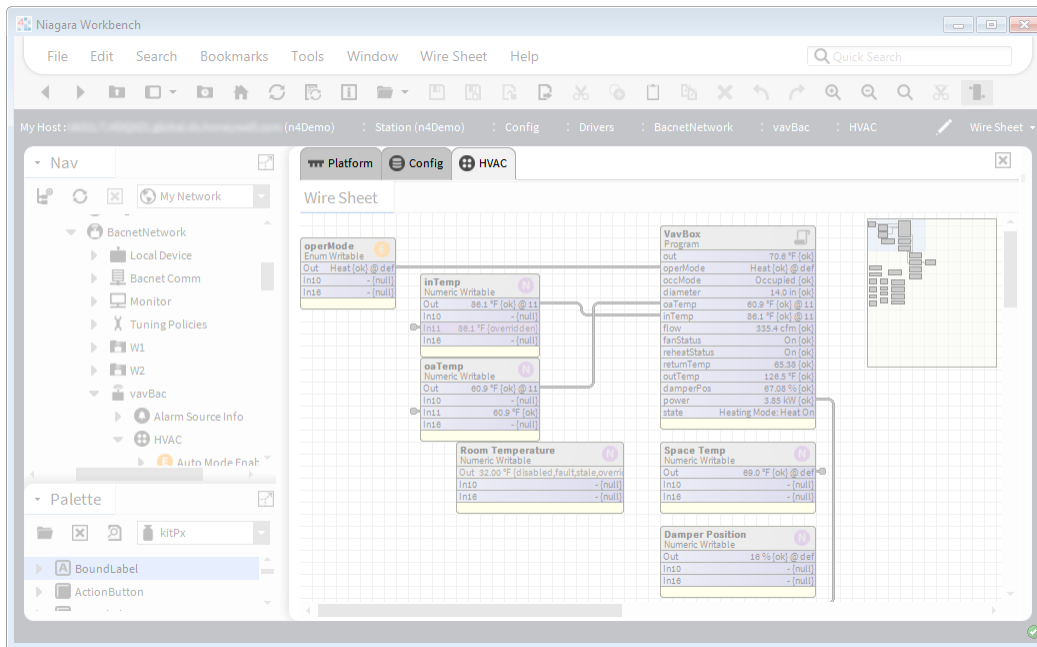
The **Bound Label** option is immediately available in the **New** popup submenu.

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 41 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:

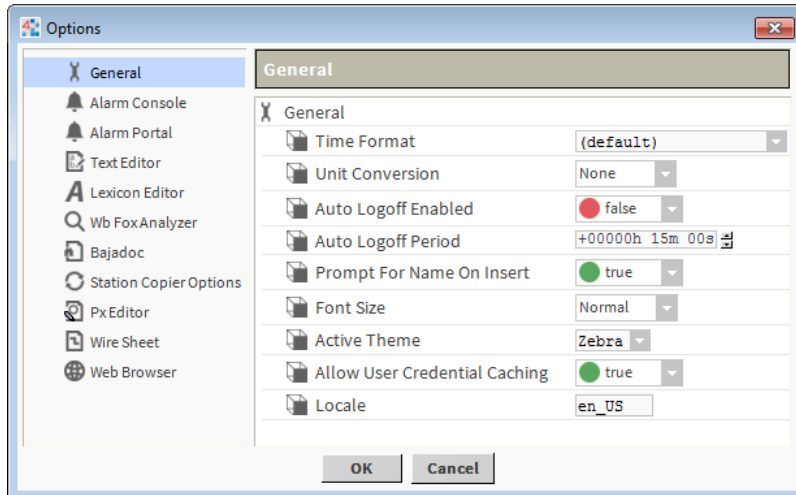
- Click  (Close icon) located in the upper-right corner of the view, just below the View drop-down list.
- Right-click a tab and choose **Close Tab** to close that tab
- Right-click a tab and choose **Close Other Tabs** to close all tabs except that tab.
- From the menu bar, choose **File**:→ **Close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

The currently active tab no longer appears.

Types of Workbench options

Use the Workbench **options** dialog box to customize your Workbench interface and to set other preferences. Open this dialog box by selecting **Tools**→ **Options** from the menu bar.

Figure 42 Options dialog box



The following options are available in the options dialog box:

- General options
- Alarm Console options
- Alarm Portal options
- Text Editor options
- Lexicon Editor options
- Wb FoxAnalyzer options
- Bajadoc options
- Station Copier options
- Px Editor options
- Wire Sheet options
- Web Browser options

General options include the active Workbench “Theme”. For details, see “About Workbench themes”.

General options

General options include settings for a variety of Workbench display and behavior options.

Figure 43 General Workbench options

General	
Time Format	(default)
Unit Conversion	None
Auto Logoff Enabled	<input type="radio"/> false
Auto Logoff Period	+00000h 15m 00s
Prompt For Name On Insert	<input checked="" type="radio"/> true
Font Size	Normal
Active Theme	Zebra
Allow User Credential Caching	<input checked="" type="radio"/> 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.

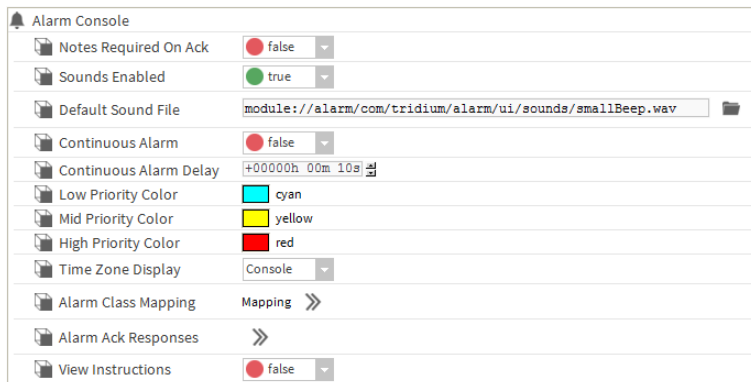
- **Time Format**
Choose from a format option to set the way that time values are displayed by default.
- **Unit Conversion**
Choosing the English or Metric option converts values that are displayed in Workbench to the chosen unit type. Selecting None leaves units in the state that is assigned at the point facet.
- **AutoLogoff Enabled**
Setting this parameter to True will activate the AutoLogoff feature. When activated, the AutoLogoff feature will automatically log off a user from a platform when there is no activity detected in Workbench for the period specified in the AutoLogoff Period field. Setting this parameter to False will disable the AutoLogoff feature.
- **AutoLogoff Period**
Time until Workbench logs off a user when AutoLogoffEnable is set to True.
- **Prompt For Name On Insert**
When set to True, Workbench displays a **Name** dialog, 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. See “About Workbench themes”.
- **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 popup **Authentication** dialog. If selected, the connection credentials are then cached and available in the Workbench “Credentials Manager” (**Tools**→**ManageCredentials**).

This is a “convenience” mechanism. However, for security best practices it is recommended to globally disable user credential caching by setting this property to false. This way that option is unavailable in the **Authentication** dialog. For related details, see “Credentials manager”.
- **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 options

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

Figure 44 Alarm console options



Alarm console options include the following:

- Notes Required on Ack

This option, when set to true, causes the **Notes** dialog box to open whenever you initiate an alarm acknowledgement from the alarm console.

- Sounds Enabled

When set 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

Sets the path to the default sound file.

- Continuous Alarm

When set to true, causes an alarm to repeat continually, until it is acknowledged or cleared. This option works together with the Continuous Alarm Delay property. You can also set this value under the **Alarms** menu in the workbench main menu when the Alarm Console view is active

- Continuous Alarm Delay

When Continuous Alarm is enabled, this property causes a “pause” time between in the continuous alarm sound. The continuous alarm is interrupted for a time equal to the value of this property.

- Alarm priority coloring

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).

- Low Priority Color

Choose a color to designate a low priority alarm.

- Mid Priority Color

Choose a color to designate a mid priority alarm.

- High Priority Color

Choose a color to designate a high priority alarm.

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.

- Time Zone Display

This property 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

Provides a way for you to create alarm classes and map specific alarms to classes.

- Alarm Ack Responses

Use this property to create one or more text entries that you can use to populate the **Notes** dialog box when acknowledging an alarm. When the Notes Required on Ack is set to true, the **Notes** dialog box displays an additional option list containing any entries you create with this property. Use the button to open the associated Edit dialog box and add, edit, or remove response options, as desired. When the Notes Required on Ack is set to false, these Alarm Ack responses are not visible.

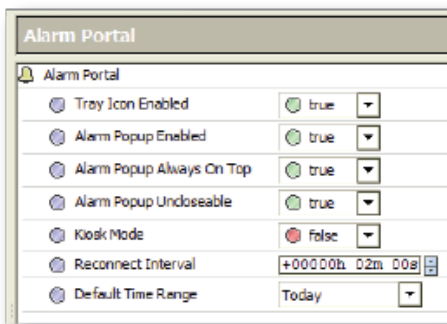
- View Instructions

When set to true, this property causes the alarm Instructions pane to display across the bottom of the Alarm Console. Instructions display in the pane for any single selected alarm that has associated instructions.

Alarm Portal options

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

Figure 45 Alarm portal options



Alarm Portal options include the following:

- Tray Icon Enabled

When set to True, displays an alarm icon in the system tray when the alarm portal is active.

- Alarm Popup Enabled

When set to True, displays an alarm popup window when the alarm portal is active.

- Alarm Popup Always On Top

When set to True, causes the alarm popup window to stay on top of other windows when the alarm portal is active.

- Alarm Popup Uncloseable

When set to True, makes the alarm popup window uncloseable when the alarm portal is active.

- Kiosk Mode

When set to True, the alarm portal opens in Kiosk Mode the next time it is started. For related information, refer to “workbench-KioskService”, the *NiagaraAX Graphics Guide* section “About Kiosk Profiles”.

- Reconnect Interval

The alarm portal checks for “disconnected” alarm consoles. If a console is disconnected, a reconnect is attempted within the Reconnect Interval time.

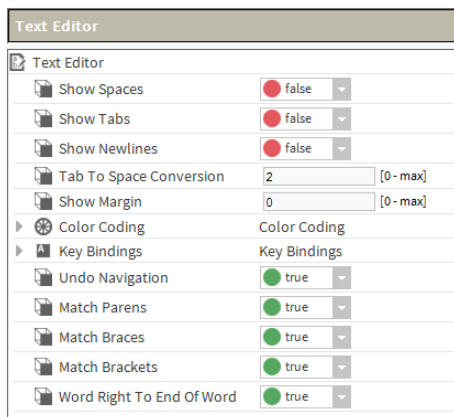
- **Default Time Range**

Allows you to choose the default time range that displays in the Portal Alarm Console pane of the Alarm Console view (Console).

Text Editor options

These options offer a whole range of ways to customize the presentation and behavior characteristics of the text editor tool. Settings include text and symbol color coding options, as well as key bindings for shortcut keys. The text editor options properties are shown .

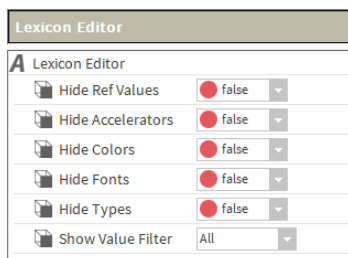
Figure 46 Text editor options



Lexicon Editor options

These options offer ways to customize the default settings of the Lexicon Editor. The Lexicon Editor options properties are shown and described in **Lexicon Editor** view. All of these properties may be overridden using the options that are available in the Lexicon Editor.

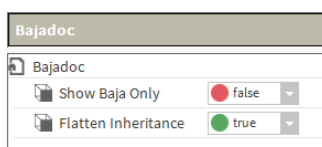
Figure 47 Lexicon editor options



Bajadoc options

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

Figure 48 Bajadoc options



- Show Baja Only

When set to True displays only the reference documentation for slots (Properties, Actions, and Topics). When set to False, documentation on the Java constructors, methods and fields is also displayed.

- Flatten Inheritance

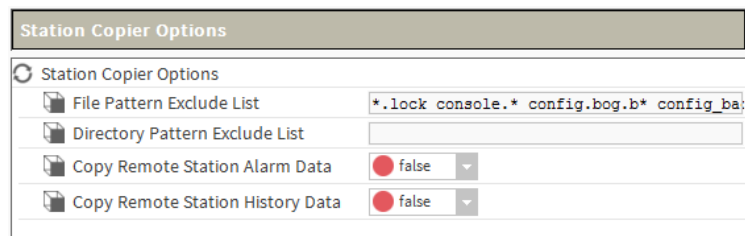
This option allows you to flatten the inheritance hierarchy into a single set of documentation. When set to False only the Java members and Baja slots declared in the specified class are displayed.

When set to True 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 a 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 49 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.

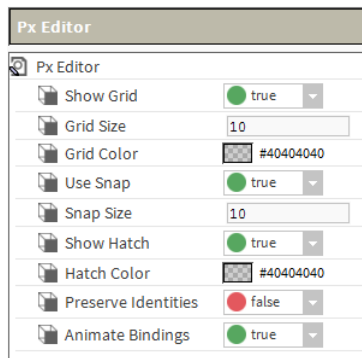
- Copy options: (Remote Station Alarm Data, Remote Station History Data)

These properties allow you to choose to copy (true) or not copy (false) certain station data.

Px Editor options

These options offer ways to customize the presentation and behavior characteristics of the **Px Editor** view. The Px Editor options properties are shown and described in the following list:

Figure 50 Px editor options

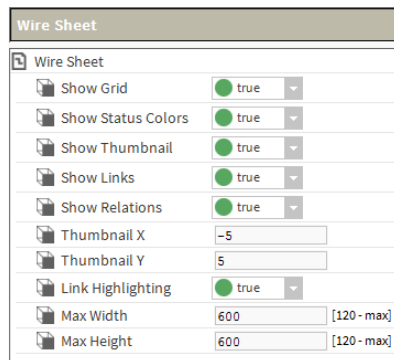


- **Show Grid**
This property sets the default condition of the Px editor grid. Select true to make the grid visible by default or select false to make the grid hidden by default. Either setting may be changed at any time using the PxEditor menu.
- **Grid Size**
This property sets the size of the grid in the Px editor.
- **Grid Color**
This property sets the color of the grid in the Px editor. Click in the color field to display the Color Choose dialog box. Use the Color Chooser to set the color that you want to assign to the grid.
- **Use Snap**
This property 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**
Set an integer value in this field to define the interval between successive snaps.
- **Show Hatch**
This property 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**
This property sets the color of the hatching in the Px editor. Click in the color field to display the **Color Choose** dialog box. Use the Color Chooser to set the color that you want to assign to the Px editor hatching.
- **Preserve Identities**
When set to true, this property allows you to explicitly turn on support for encoding all names and handles on a Px page.
- **Animate Bindings**
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.

Wire Sheet options

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

Figure 51 Wire sheet options



Wire sheet options include the following:

- **Show Grid**
When set to True displays a grid background on the wire sheet view.
- **Show Status Colors**
When set to True, will display a different color for each status when it appears in the wire sheet.
- **Show Thumbnail**
When set to True – provides a small “thumbnail” view of the whole wire sheet for orientation and navigation purposes.
- **Thumbnail X**
A field is provided here for setting the X axis for the default position of the thumbnail view.
- **Thumbnail Y**
A field is provided here for setting the Y axis for the default position of the thumbnail view.
- **Link Highlighting**
When set to True – turns on link highlighting.
- **Max Width**
Use this property to specify a maximum size for the wire sheet width.
- **Max Height**
Use this property to specify a maximum size for the wire sheet height.

For more details, see “Working with the Wire Sheet view”.

Web Browser options

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

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. In addition to permitting you to customize your display, custom themes can also be used to give Workbench a “branded” appearance.

Types of themes

Workbench provides standard “built-in” themes as part of the default application environment. You can also create and use custom themes in Workbench.

Built-in themes

Following are two Workbench themes listed and illustrated below:

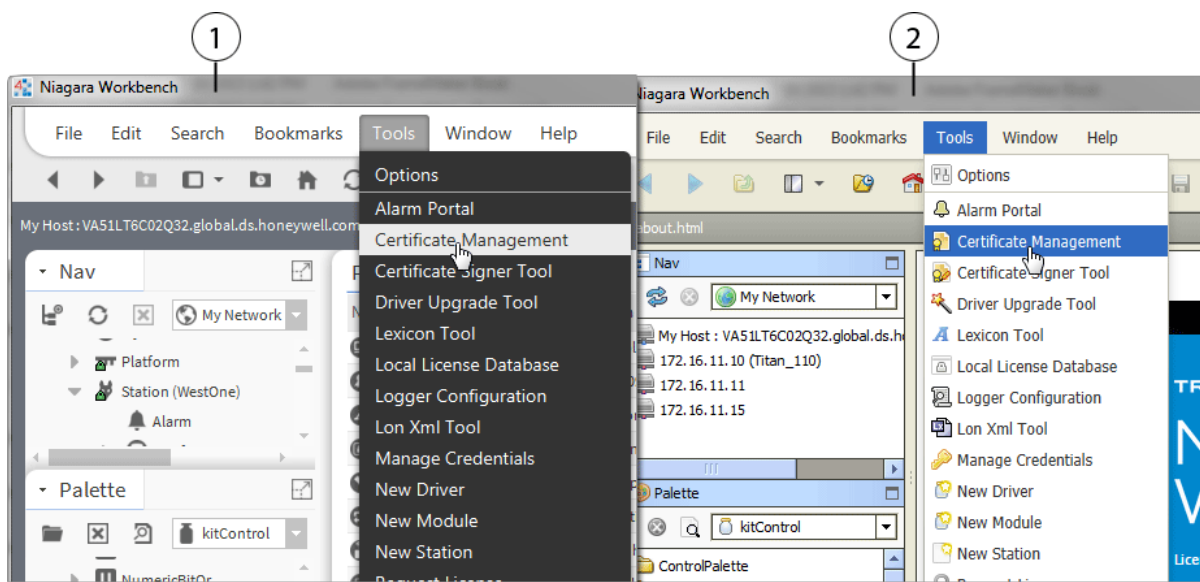
1. Zebra

This theme has a low contrast, grayscale coloring and is the default theme.

2. Lucid

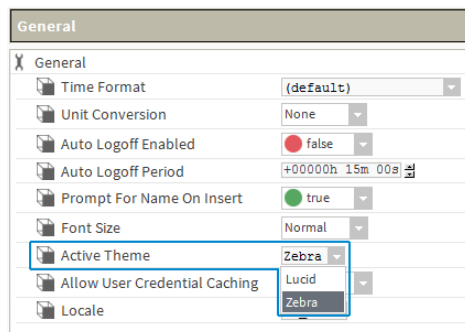
Displays a blue and gray color scheme.

Figure 52 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 53 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.

Custom themes

You can define a unique custom theme for your brand by creating a new theme module that specifies colors (such as, corporate colors), fonts, and icons to be used in the Workbench interface.

Additionally, you can designate your new custom theme as the default active theme in Workbench. Workbench automatically uses this theme unless the user makes another selection. Also, you can also “force” the use of the default theme, effectively locking Workbench into using this theme only.

NOTE: Creating a theme module is a fairly advanced procedure, which typically involves modifications to CSS, NSS, icon images, and building a jar file, and so it is not covered in this guide. For complete details on this, see the Niagara 4 Developers Guide.

Designating a default theme

By default, the Zebra theme is the **Active Theme** in Workbench. You can select a default theme from those shown in the **Tools→Options→Active Theme** dropdown list. Once you have restarted Workbench, the selected theme will be in effect. You can also “lock” the active theme, preventing selection of a different theme.

This procedure explains how to designate a default theme and lock it.

NOTE: Locking the theme hides the **Active theme** dropdown list, normally visible under **Tools→Options**, so that no other theme can be selected.

Step 1 Open the `brand.properties` file in the `SysHome/etc` folder and add the following two lines of text, replacing “`themeModuleName`” with the name of the default theme module, as shown.

- `workbench.theme.default=<themeModuleName>`
- `workbench.theme.locked=true`

```
13 #workbench.splash=module://workbench/com/t;
14
15 # designate default theme
16 workbench.theme.default=myThemeModule
17 # lock active theme
18 workbench.theme.locked=true
```

Step 2 Click **File→ Save** to save your changes in the `brand.properties` file.

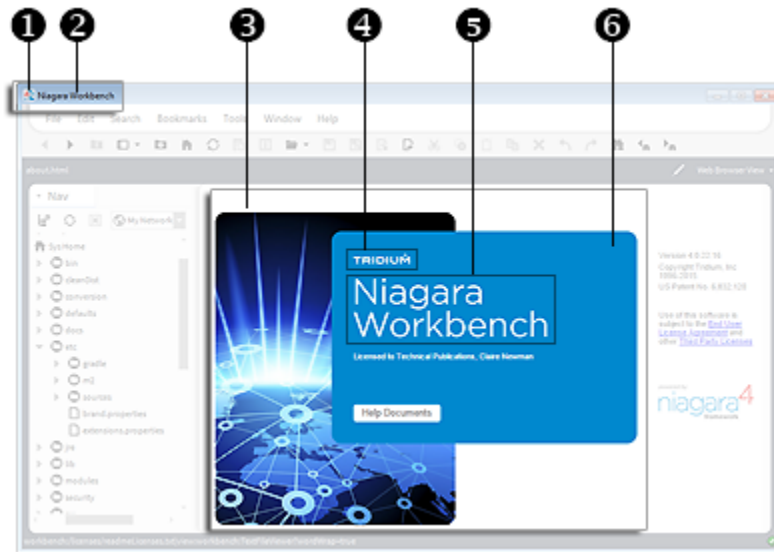
Step 3 Restart Workbench for this change to take affect.

About branding the Workbench splash screen

An alternative to creating a custom theme is to customize elements of the Workbench splash screen to reflect aspects of a corporate brand, such as specific colors, text, and images. The term “splash screen” refers to the initial welcome page that displays when you launch Workbench. This can be useful for OEM and SI partners delivering valid custom branded Niagara systems to their customers.

The basic brand information for Workbench is contained in the `SysHome\etc\brand.properties` file. Elements such as, the Workbench icon, Workbench title (both positioned in upper-left corner), and the splash screen image, text, and logo (all positioned in the right pane) can be defined in the `brand.properties` file.

Figure 54 Customizable elements of the splash screen

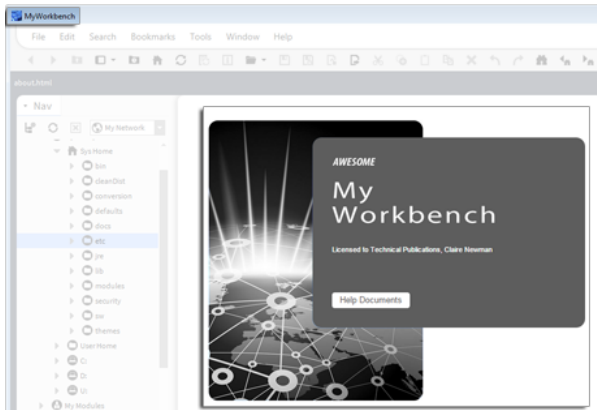


1	<p><code>workbench.icon</code> The default size for this image is 32 x 32 pixels. It is a 32-bit PNG graphic file, where the background is transparent.</p>
2	<p><code>workbench.title</code> This text displays in the upper-left corner, to the immediate right of <code>workbench.icon</code>.</p>
3	<p><code>workbench.splash</code> The default size for this image is 660 x 860 pixels (6"x9"). This dimension can include white space around a smaller image such as that used in the default splash screen. It is a 24-bit JPEG graphic file. The image does not include the solid, round-cornered rectangle in the splash screen which is provided by CSS.</p>
4	<p><code>workbench.splashLogo</code> The default size for this image is 155 x 29 pixels. It is a 32-bit PNG graphic file, where everything except the logo itself is transparent.</p>
5	<p><code>workbench.splashTitleImage</code> The default size for this image is 532 x 161 pixels. It is a 32-bit PNG graphic file, where everything except the title text or image itself is transparent.</p>
6	<p><code>workbench.css</code> The background and foreground colors for the solid, round-cornered rectangle in the splash screen can be modified by overriding the CSS <code>.cover</code> class definitions for these elements. The example shown below sets the background color to medium gray and the foreground color to white.</p>

Example of customized brand.properties

```
workbench.title=MyWorkbench
workbench.icon=file:!etc/icon.png
workbench.css=.cover {background-color: #5D5D5D;color: white;}
workbench.splash=file:!etc/splash.jpg
workbench.splashLogo=file:!etc/splashLogo.png
workbench.splashTitleImage=file:!etc/splashTitle.png
```

Figure 55 Customized splash screen



NOTE: Optionally, you can brand the Niagara Installer application. For details, refer to the Niagara 4 Developers Guide.

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

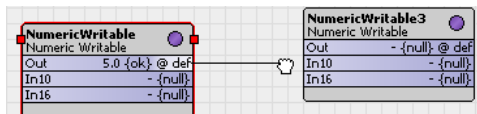
An object on a Wire Sheet view of a data model is a point component that represents a piece of data.

Basic object linking

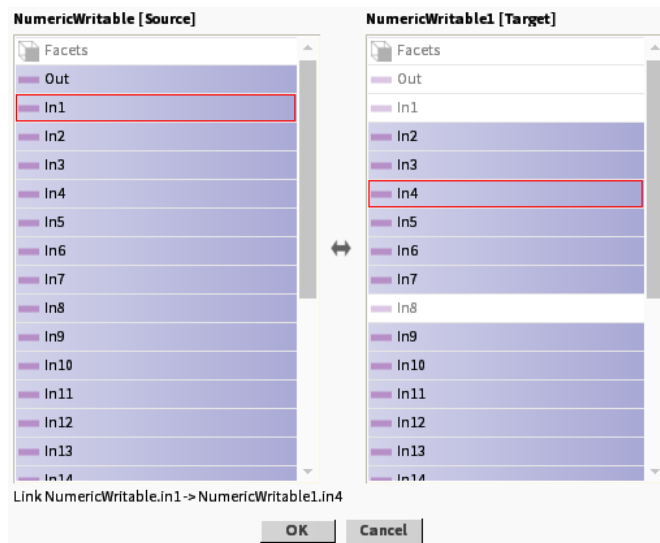
Object links define the direction of data flow.

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.

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** dialog box 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.

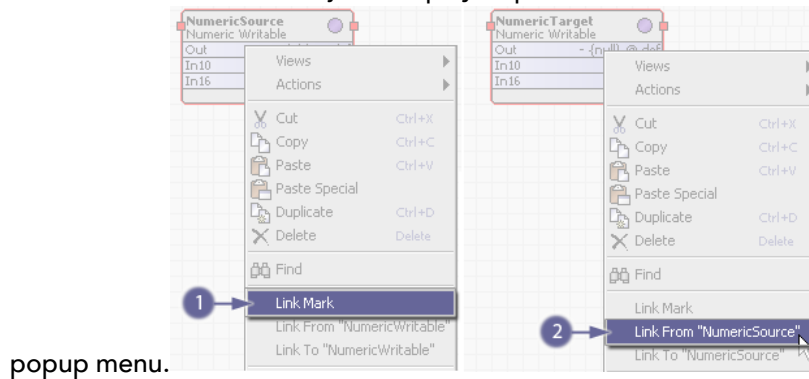
Step 2 To deactivate the link state, release the **Shift** key and click anywhere or touch any key.

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




popup menu.

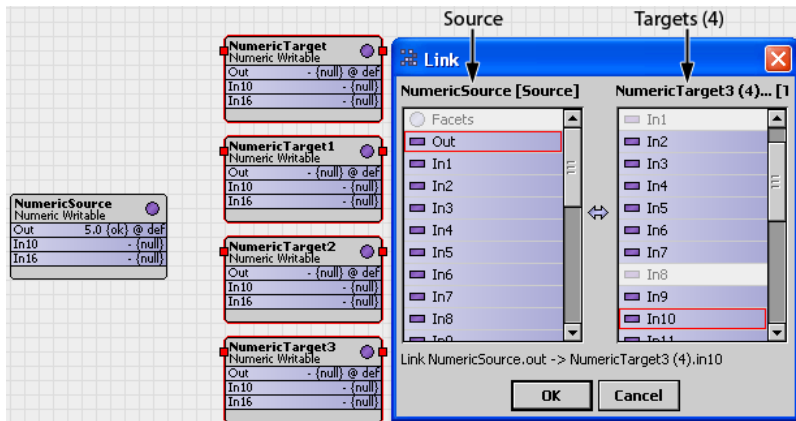
Step 2 To define one or more selected objects as a link source or link target, select the **Link From** command

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

Step 3 To define one or more selected objects as a link source or link target, Sselect 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 dialog box using the **Reverse** button .

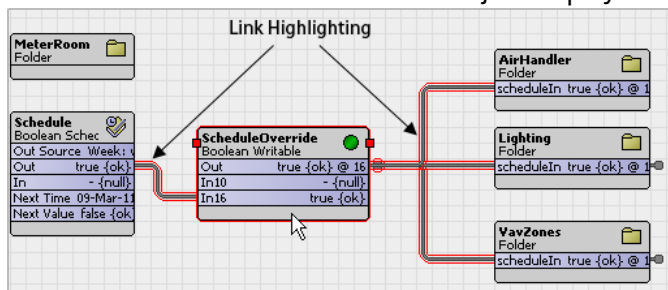


Viewing links

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.



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).

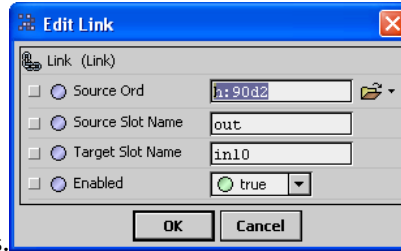
Step 3 To customize the colors, edit the `system.properties` file (in the Nav tree `lib` folder, under **My-Files** → **Sys Home**).

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 Link** window- without having to go to the source component's Link Sheet view

Step 1 To open the **Edit Link** window, right-click a single link (not multiple links or knob links) and select **Edit Link** from the popup menu. The **Edit Link** dialog box displays, as shown below.



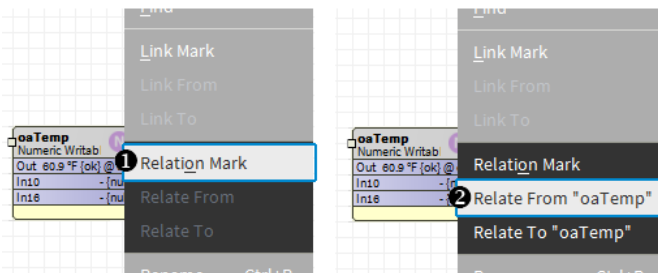
The **Edit Link** window opens.

- 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.



- 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** dialog 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.

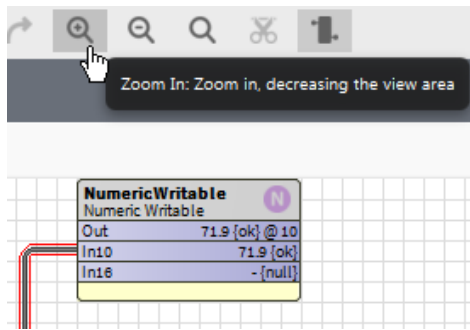
This command opens the **Relation** dialog box with the “marked” object as the target object.

For more details on object relations, see the *Relations Guide*.

Zoom controls

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

Figure 56 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.

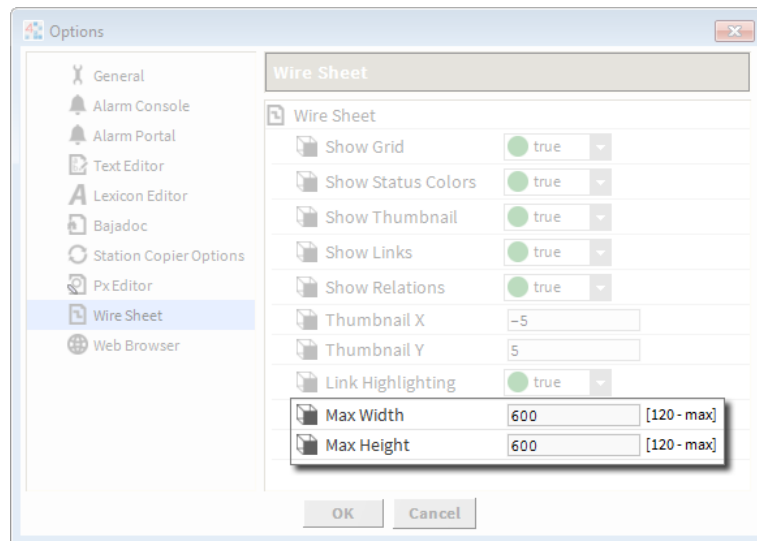
Keeping all objects within view

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.

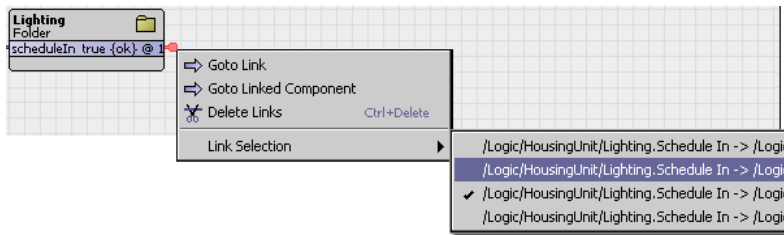


Step 3 Enter a desired maximum value (in pixels) in the **Max Width** and **Max Height** fields and click **OK**.

Link navigation

Link knobs provide easy navigation.

Figure 57 Link Selection menu

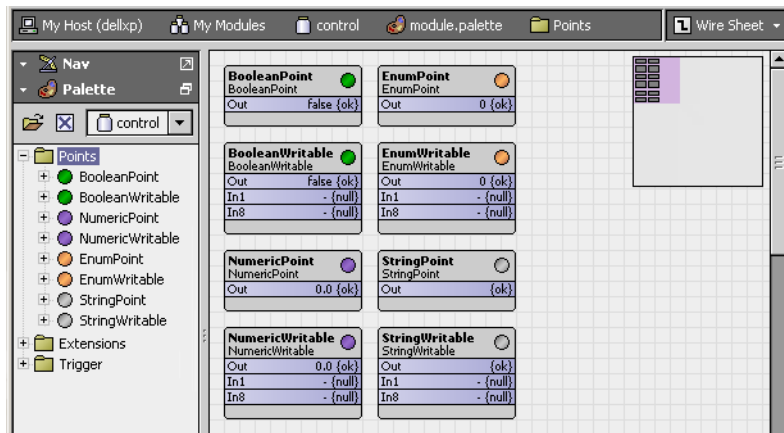


- Off-view linking
Double-clicking on the knob hyperlink at the opposite end of the link, displays the Wire Sheet view with the linked knob highlighted.
- Goto Link command
Right-clicking on the knob displays a **GoToLink** command on the popup menu. Selecting this command hyperlinks to the component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

About control points

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

Figure 58 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, weekly schedules). 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 discretets. 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 “About the proxy extension”, or refer to “About proxy points” in the *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). For more details, see “About point actions”. Writable points also have other features. See, “About writable points”.

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

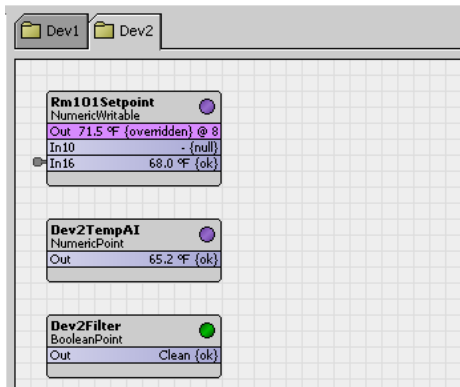
- Extensions, which expand a given point’s functionality. As needed, you can add one or more extensions to a point, each as a child of that point. Extensions add functionality in a modular fashion.
- Time triggers provide periodic actions. These objects do not represent data, but, instead, they regularly 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 section “Application for kitControl components” in the *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 59 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.

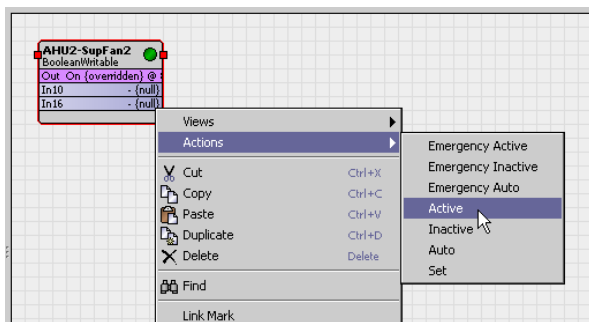
- The currently active priority level (from a 16-level priority scheme) for writable control points only. For more details on priority, see “About the priority scheme”.

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 a Workbench view or in the Nav tree.

Figure 60 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 popup window. See “About override actions”.
- Set the value of the point’s `Fallback` property. See “About set (Fallback) action”.

Often, you modify a writable point's default actions—see “Modifying default actions”.

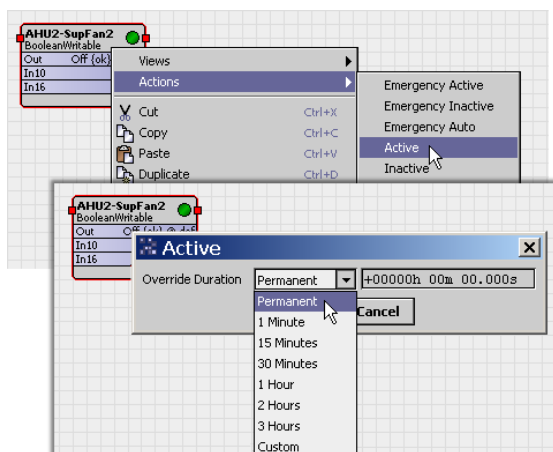
About override actions

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 magenta as shown below. For more point status details, see “How status flags are set”.

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 61 Override action duration (“argument”) dialog



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

If needed, you can limit the maximum duration of manual override using facets—see “Maximum override duration facet”.

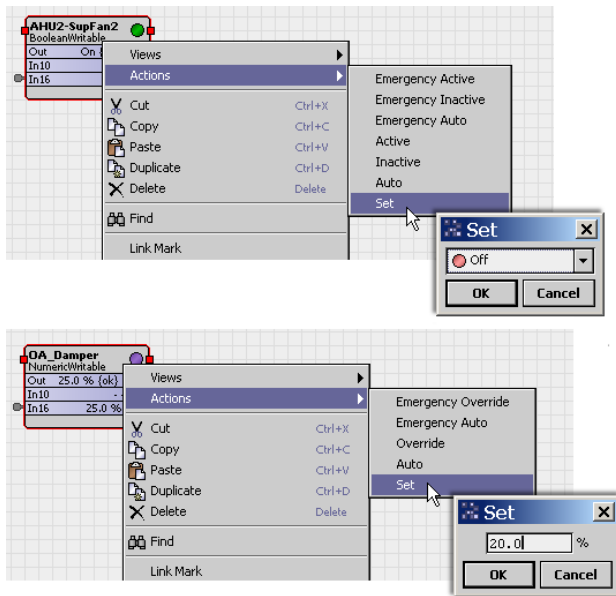
After clicking OK, the override action is issued to the point—if this is the highest effective priority level, the writable point operates under this control. If this is a timed override, the action is automatically auto’ed upon expiration of the override period.

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. For more details, see “About the priority scheme”.

By default, an operator-level user can change the Fallback property, using the “set” action. This produces a popup dialog that displays the current Fallback value.

Figure 62 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 prompt 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 NiagaraNetwork. 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 *NiagaraAX Graphics Guide* section “About Widgets”.

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 “About Constant components” section in the *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. For general information, see “About slots”, and “Using the slot sheet”.

Or, using facets you can limit the duration of manual overrides—see “Maximum override duration facet”.

The following sections provide more details on slot sheet techniques:

- Display names — Change how the point’s popup 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 “Notes on English (en) lexicon usage” in the *Niagara 4 Lexicon Guide*.

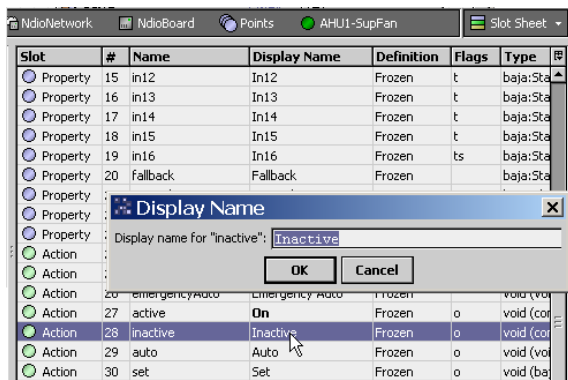
You can also modify the display name of the same action in multiple points in a single operation, using the (default) “Batch Editor” view of the station’s **ProgramService** (Config > Services > ProgramService). This is one of many examples of using the **Batch Editor**.

Also, starting in AX-3.6, you can use the **Batch Editor** to modify a config flag of the same slot on multiple components in a single operation. For details, refer to the *Batch Editor - Engineering Notes* document.

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 63 Editing action display names from slot sheet



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

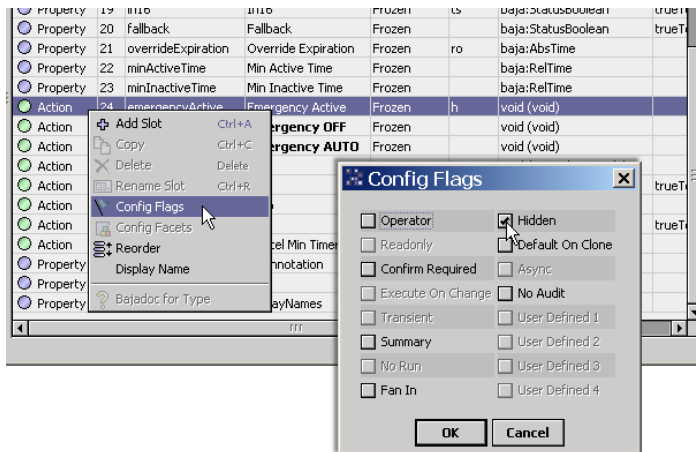
Action access

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 *NiagaraAX Graphics Guide* section “Types of binding properties”.

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

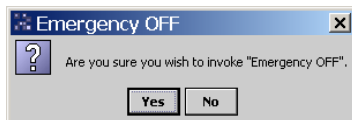
Figure 64 Editing config flags of action to “hide” or change permission level



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

- **Operator**
If checked, only operator-level access is needed to invoke the action. If cleared, admin-level access is needed. For details on permission levels, see “About permission levels”.
- **Confirm Required**
If checked, a second (confirmation) dialog appears after the action is invoked, before the action executes. An example confirmation dialog is shown below. By default, this flag is cleared.

Figure 65 Action confirmation dialog (from “Confirm Required” config flag)



- **Hidden**
If checked, the action does not appear (is hidden) from the action popup 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.)

As previously noted, starting in AX-3.6, the **Batch Editor** lets you modify a slot’s config flag on multiple points in one operation. Refer to the *Batch Editor - Engineering Notes* document for details.

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.

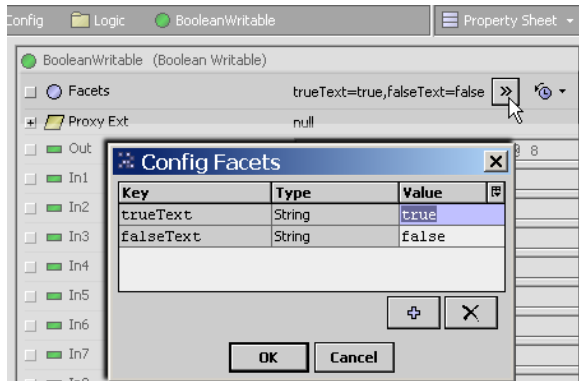
With the exception of proxy points (with possible defined device facets), point facets do not affect how the point’s value is processed. Refer to the *Drivers Guide* section “Effect of facets on proxy points” for related details.

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. As shown, you modify facet values in a point's property sheet, using the **Config Facets** dialog.

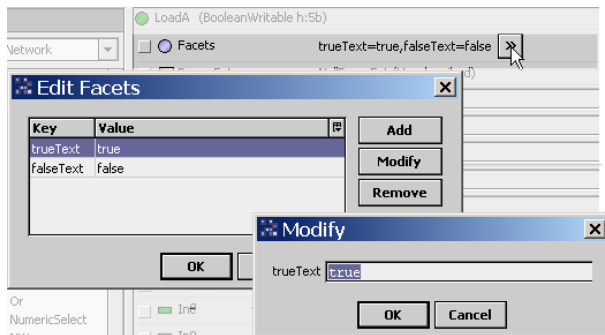
Figure 66 Point facets and edit dialog (AX-3.3 and later shown)



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.

NOTE: If using Workbench AX-3.2 or earlier, you see a slightly different facets editor—where you must click a **Modify** button to edit existing facets. See below. However, the basic operation is the same.

Figure 67 Point facets and edit dialog (Workbench AX-3.2 or earlier)



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 (see "Maximum override duration facet").

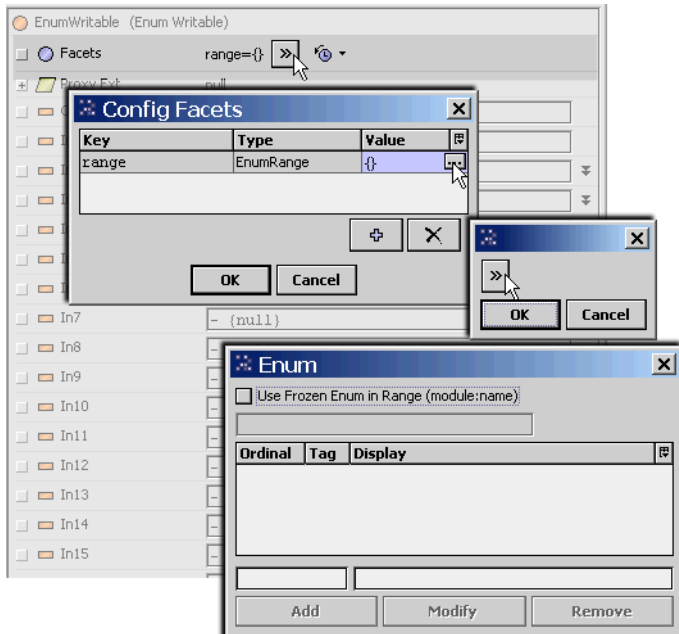
NOTE: For string-type points (StringPoint, StringWritable), facets typically have little practical application. By default, the Facets slot is empty for string-type points.

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.

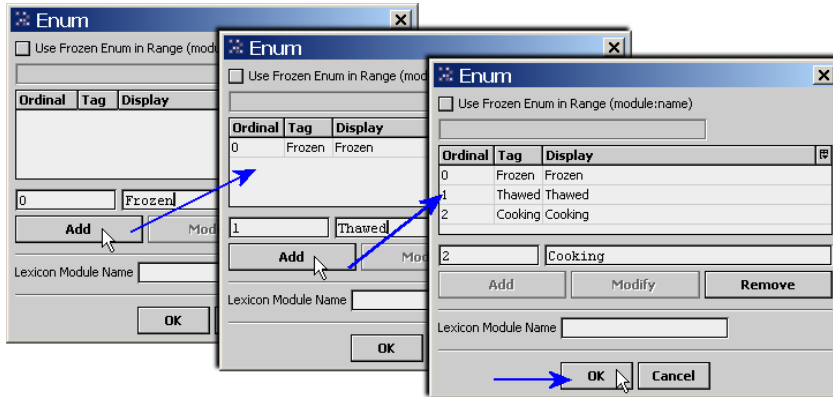
If you add an enum point from the control palette, its Facets slot has a blank range entry. Until you edit this facet and supply the ordinal-tag values, it can display only integer values. As shown, a special **Enum** window appears when you edit range facets.

Figure 68 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 pop-ups.

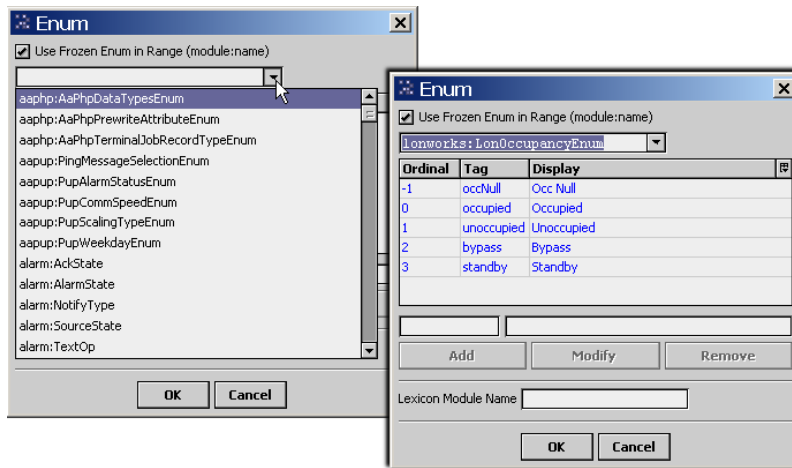
Figure 69 Type unique integer value in Ordinal box and associated text in Tag box



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 70 Frozen enum selection in Enum window



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

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

Effect of facets on point actions

For some points with actions (see "About point 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.

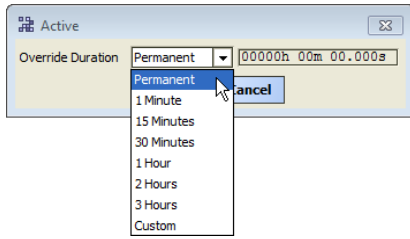
- Maximum override duration (for any writable point type)

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. For more details, see "Maximum override duration facet".

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 71 Default override action menu for writable control point

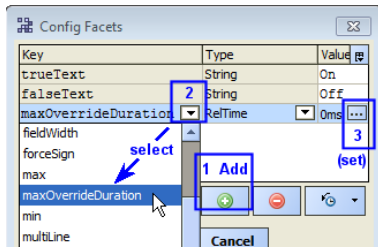


If needed, change this by adding a "maxOverrideDuration" facet (choosing type baja:RelTime), with specified duration time, to either or both:

- 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 72 Config Facets editor when picking maxOverrideDuration



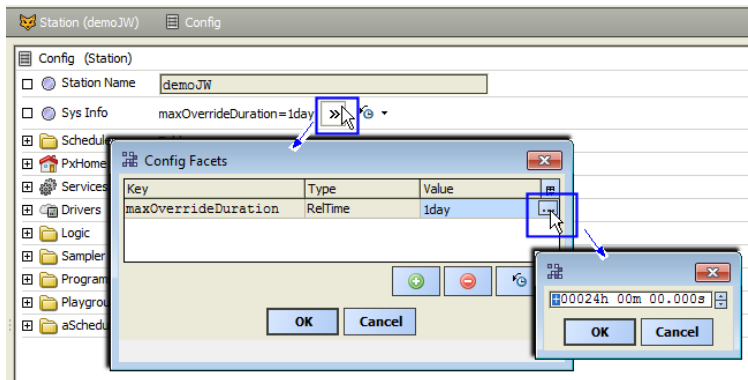
When a writable point is limited by a maxOverrideDuration facet, its action menu adjusts to show the allowable range. See "Action menu examples".

For details about editing facets, see "Accessing and editing facets".

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 73 Global maxOverrideDuration facet added to Sys Info property of station's Config component

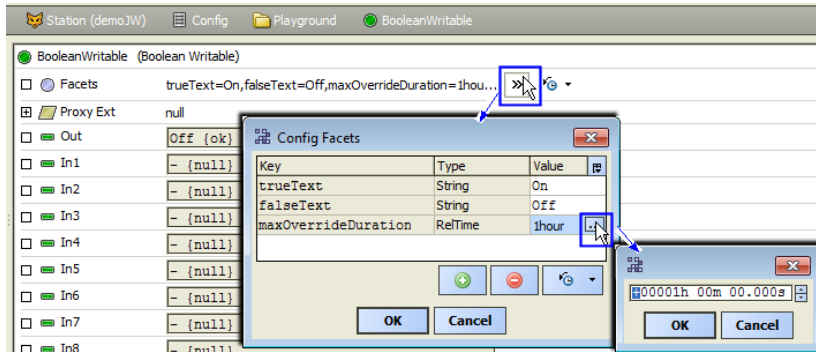


Adding this facet on the Sys Info property acts a global limit (station-wide) to a manual override action to all control points that do not have their own "maxOverrideDuration" facet.

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 74 Added `maxOverrideDuration` facet added at point level (overrides global setting)

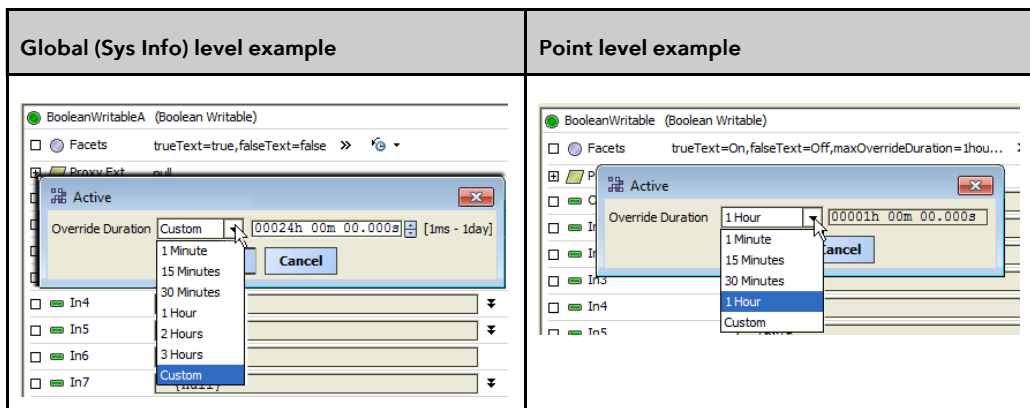


As shown, this `maxOverrideDuration` facet can be added along with any other facets in use by the control point. The example BooleanWritable point above already had configured facets for `trueText` and `falseText`.

Action menu examples

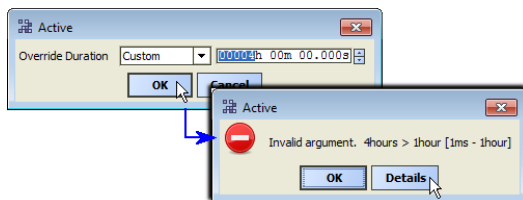
Example action menus from writable points with a `maxOverrideDuration` facet in effect are shown .

Figure 75 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 popup appears that shows the override duration range as shown below.

Figure 76 Custom override attempt over `maxOverrideDuration` limit produces error popup



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, see “About the proxy extension”) and also optional extensions.

There are three main categories of optional extensions:

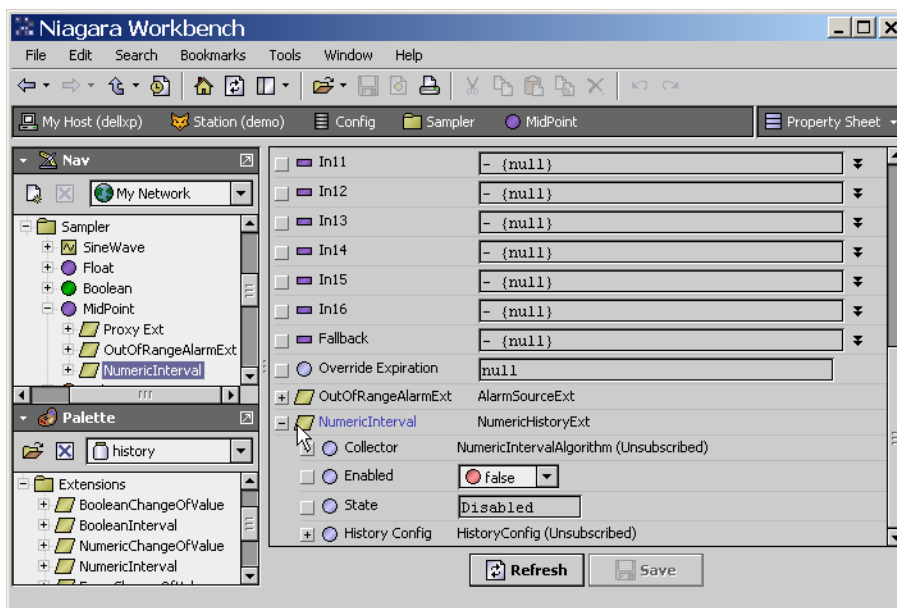
- Control extensions come from the **control** palette.
- Alarm extensions come from the **alarm** and **kitControl** palettes
- history extensions come from the **history** palette

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 77 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. For more details, see “About composites”.

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 “Extensions and kitControl components” in the *NiagaraAX KitControl Guide*.

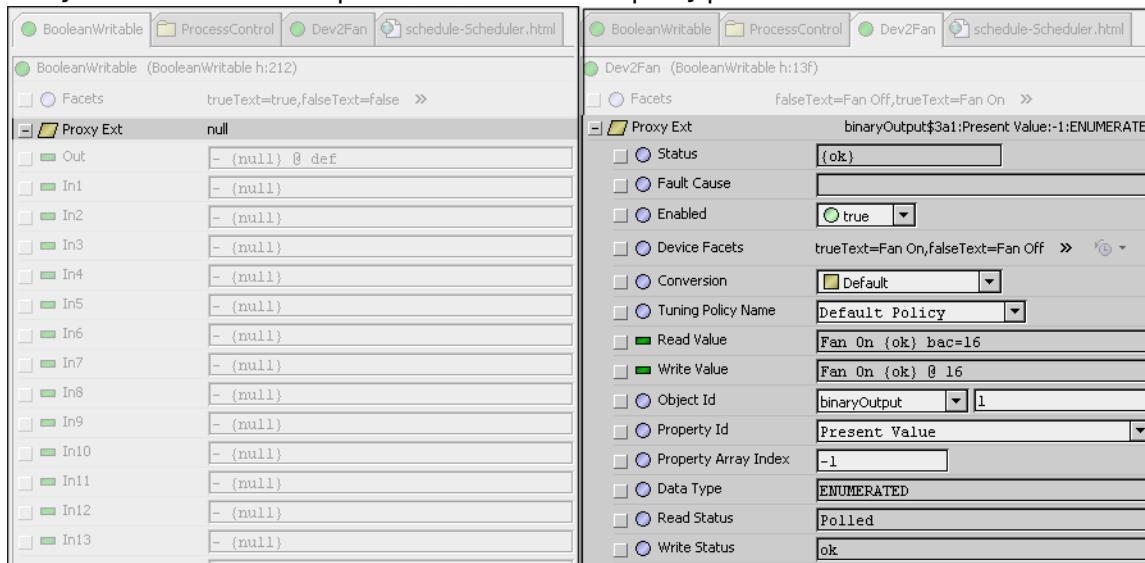
- <DriverType>

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.

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

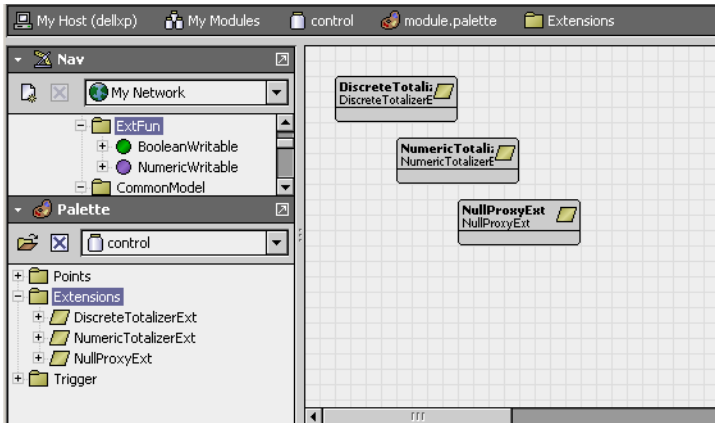


For more details, refer to Drivers Guide sections “About proxy points” and “ProxyExt properties”.

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 78 Control extensions



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

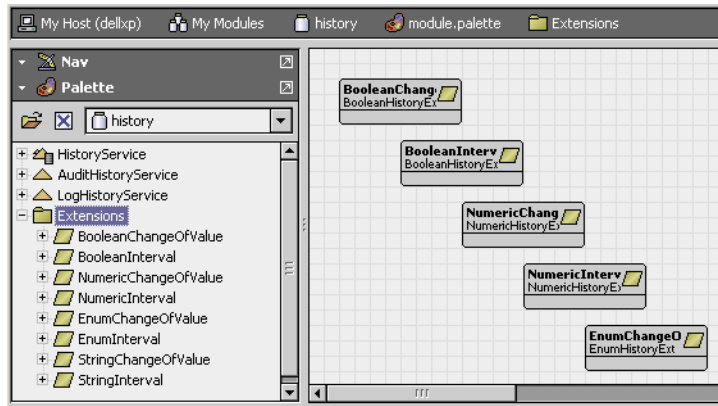
Control extension type (palette:Folder)	Applies to point types		What it does
	(read-only)	Writable	
DiscreteTotalizerExt (control:Extensions)	BooleanPoint	BooleanWritable	Accumulates runtime and change of state (COS) count. Extension actions permit resetting (zeroing) the runtime and COS count.
	EnumPoint	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"	
ProxyExt (control:Extensions)	(standard for any point, see ""About the proxy extension")		Provides methods to driver communications.

About history extensions

Add a history extension to any point for which you want 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.

Find the history extensions in palette **history: Extensions**.

Figure 79 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	
BooleanChangeOfValue	BooleanPoint —	BooleanWritable any object with single Boolean Out, e.g. kitControl: Logic object type "And"	Collects upon each change of Boolean value (state) or status.
BooleanInterval	as above	as above	Collects upon a repeating time interval, as configured.
NumericChangeOfValue	NumericPoint —	NumericWritable any object with single Numeric Out, e.g. kitControl: Math object type "Add"	Collects upon each change of value (outside a specified tolerance) or change of status.
NumericInterval	as above	as above	Collects upon a repeating time interval, as configured.
EnumChangeOfValue	EnumPoint —	EnumWritable any object with single Enum Out	Collects upon each change of enumerated state or status.
EnumInterval	as above	as above	Collects upon a repeating time interval, as configured.
StringChangeOfValue	StringPoint —	StringWritable any object with single String Out	Collects upon each change of string value or status.
StringInterval	as above	as above	Collects upon a repeating time interval, as configured.

All history extensions have an Update History action available. This popup 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

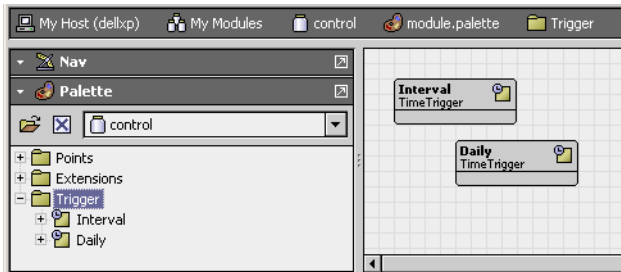
a history extension is set to `%parent.name%`, then the History Config Id is initially named based on this parent display name. However, if you rename the parent component, the History Config Id property does not automatically or immediately change. The Update History Id action invokes a renaming of 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. See “About history names” for more information about history naming.

For more details about history extensions, see “Configure history extensions”.

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 80 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. See “About trigger schedules”.

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. For more details, see "About lexicons" and "Lexicon Editor view".







The following topics explain further details about point status:

- Types of status flags
- Priority of status indication
- How status flags are set
- About "isValid" status check

Types of status flags

Status flag types are listed in Table 3-5, along with associated default colors (if any).

Status flag types

Type	Default Colors, Example	Meaning
alarm	white text, red background 	Point currently has a value in an alarm range, as defined by property in its alarm extension.
fault	black text, orange background 	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 	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 	Originates from a proxy point only. Point (or its parent device or network) has been manually disabled (property enabled = false).
down	black text, yellow background 	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 	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 <i>NiagaraAX KitControl Guide</i> section "Status value to simple value" for more details, including its "About null values" section.
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, refer to "Proxy point status" in the *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: Refer also to the *Drivers Guide* section "Proxy point status" for further details.

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

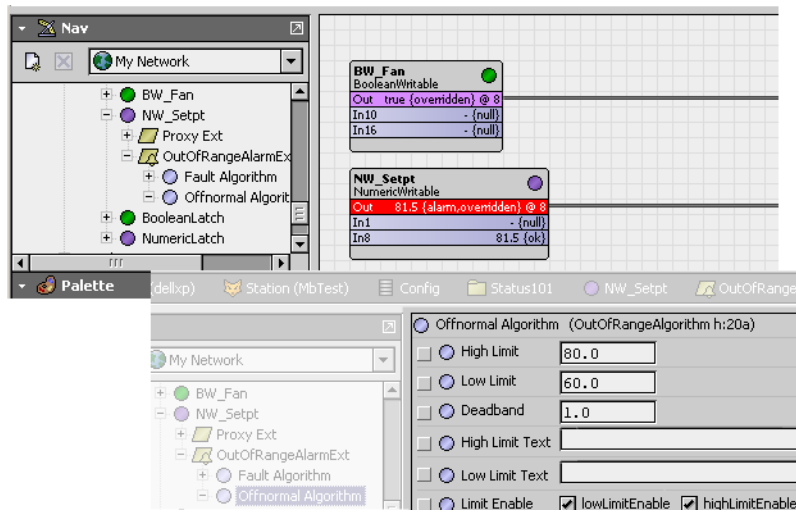
Writable points only, during a user-initiated action at priority levels 8 (override) or 1 (emergency). The override flag clears when the action “times out,” or when a user issues an “auto” action at that same priority level.

- null

Writable point has “null” or otherwise “invalid” value at In1—In16, plus “null” configured as “Fallback” value. See “About “isValid” status check”.

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

Figure 81 Status with simple control points



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

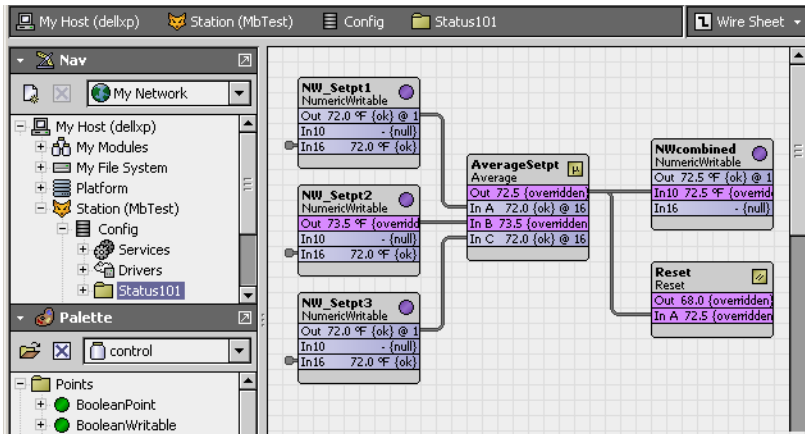
NOTE: If the math or logic object has multiple inputs, and you set the propagateFlags property to select one or more of the statuses above, simple “OR” logic is used across all inputs for the propagation of each selected status.

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. See “About “isValid” status check”.

As an example of status propagation, some number of NumericWritable points are used to establish set-points, 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 82 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 only 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. See “About the priority scheme”.

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 (see “About point actions”). BooleanWritable points also offer built-in “minimum on/off” timers.

- About the priority scheme
- About minimum On and Off times

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

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, note that for most inputs, you may enter a value directly in the point’s property sheet (as an alternative source).

NOTE: 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.

NOTE: 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, note that by default, a “set” action exists on any writable point, which writes directly to the Fallback value. See “About set (Fallback) action”. 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 “set” slot. Otherwise, a user can invoke a right-click command to set Fallback to any value. For more details, see “Modifying default actions”.

Priority linking rules

When linking to the priority inputs of a writable object, you may notice these default 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). See “About point actions”. Priority level 6 in a BooleanWritable is reserved for minimum on/off times. See “About minimum On and Off times”.

NOTE: Both rules vary from the r2 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, using 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 meaning only, see “About minimum On and Off times”)
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 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:

- Min Active Time: 00000h 01m 30s
Specifies that once started, the fan must run at least 90 seconds.
- Min Inactive Time: 00000h 03m 5s
Specifies that once stopped, the fan must remain stopped at 3 minutes, 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. See “Priority level conventions”.

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.

CAUTION: Composites have associated issues—see “Composite 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). See example “Point-level composite”.

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

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

Some composite examples

A few simple examples of composites are included here, as follows:

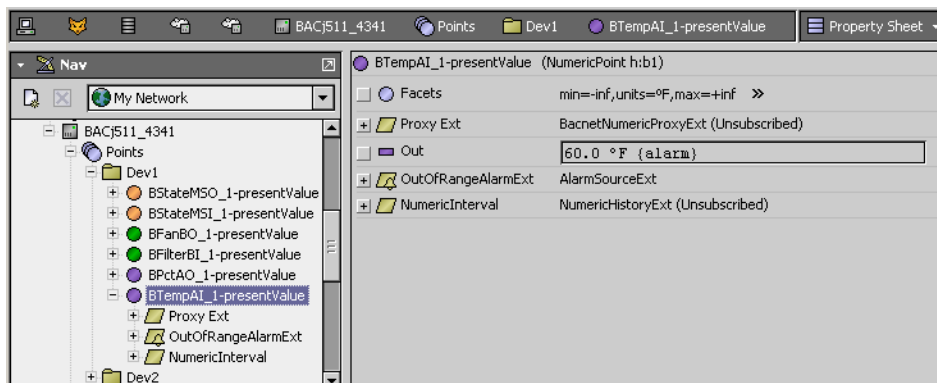
- Point-level composite
- Folder-level composite

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 83 Property sheet of proxy point with two extensions

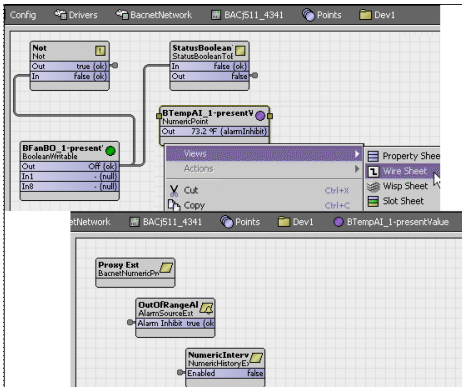


In this example, when another system-related BooleanWritable (representing the system fan) has a false (Off) status, it is desired that this temperature point be:

- 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.

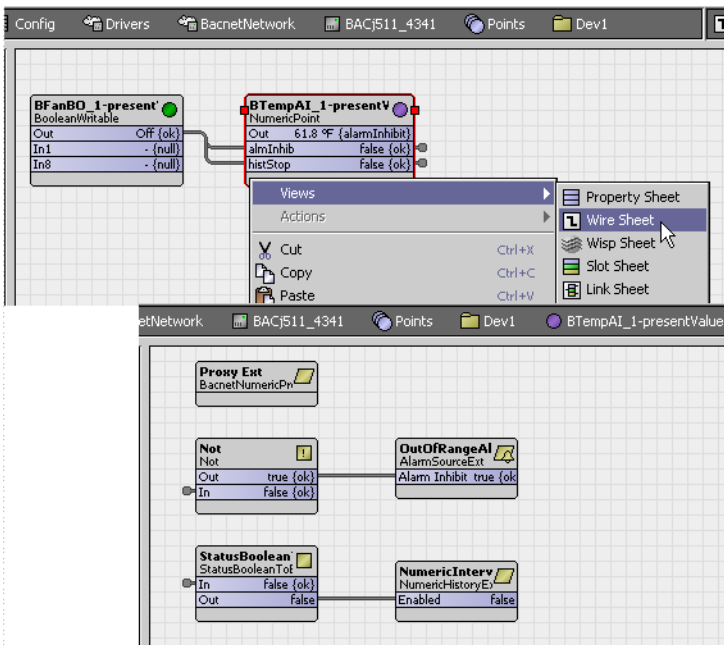
Figure 84 Proxy point example without composite



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.

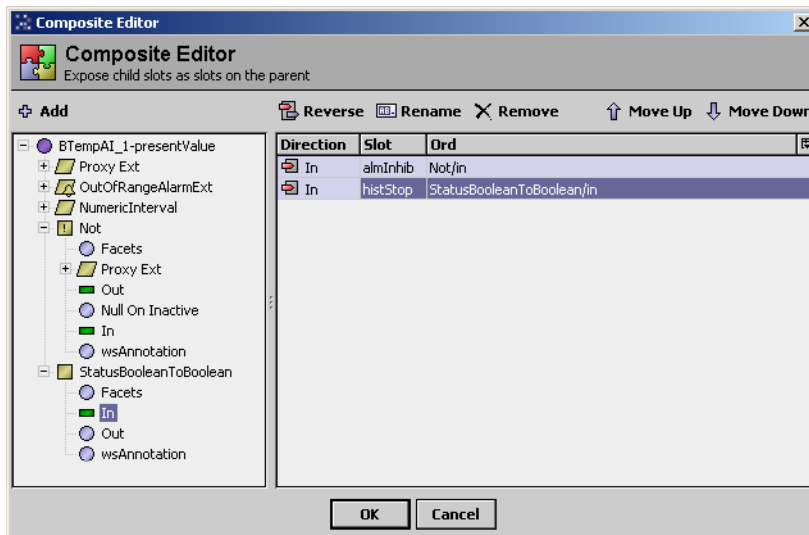
By making a composite of the NumericPoint (acting as the container for both the extensions plus the additional kitControl objects) you can simplify reuse and clarify available links. The following image shows the now-composited NumericWritable linked to the controlling BooleanWritable, and the wire sheet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

Figure 85 Proxy point as composited container



In this example, the exposed input slots in the composite were renamed from “In” to “almInhib” and “histStop” respectively, to clarify what each does. When looking at the Composite Editor for this example NumericPoint, it appears.

Figure 86 Composite Editor for example NumericPoint

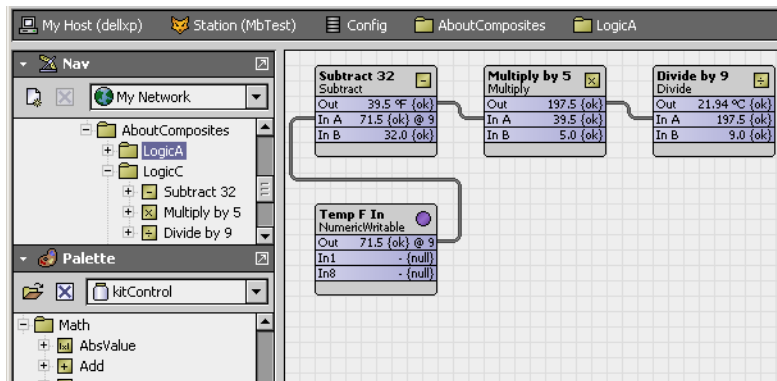


Folder-level composite

NOTE: Please see the related Caution

The following image shows a simple example of three Math objects chained together, 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.

Figure 87 Simple example of folder before compositing

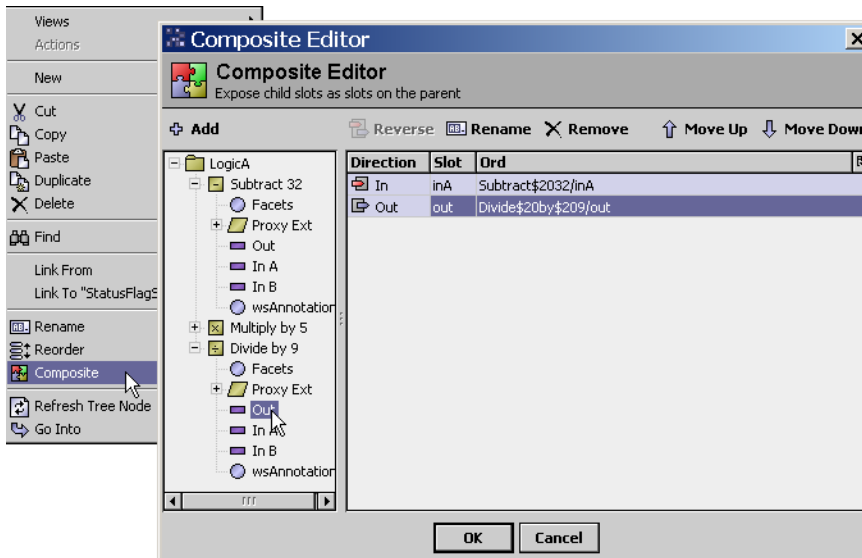


If this application was needed later, you could copy all 3 linked objects again and insert in another (perhaps already crowded) wire sheet. However, the middle "Multiply" object reveals an intermediary result that is distracting. Or, you could just create a new subfolder with only the 3 linked objects 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.

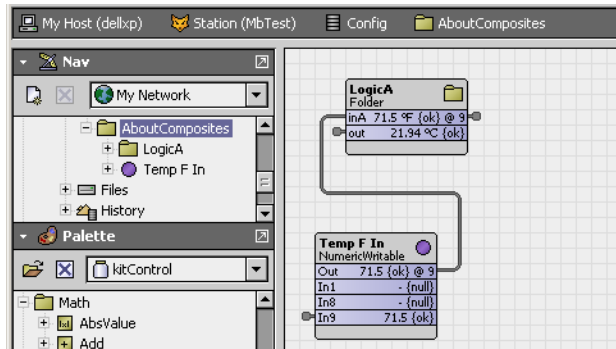
Figure 88 Launching and using the Composite Editor



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. (For more details, see Making a composite.)

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 89 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 “pinned” into the “subscribed” state. This means that proxy point will always be polling (regardless of any other usage).

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

Chapter 4 About Workbench tools

Topics covered in this chapter

- ◆ Types of Workbench Tools
- ◆ Alarm portal
- ◆ Bacnet Service
- ◆ Lexicon Tool
- ◆ Local License Database
- ◆ Logger Configuration tool
- ◆ Lon Xml Tool
- ◆ Lonworks Service
- ◆ Credentials manager
- ◆ 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 Job 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 in this section are available from the **Tools** menu. However, navigational links to some of the tools also appear in the Nav side bar and in the **Nav Container** views.

Some, but not all of the tools that are available under the **Tools** menu appear in the Nav side bar when you select them. 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 will be 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 more typically use the Job Service, BACnet service, and Lonworks Service under the “Services” node of a specific station.

Refer to “Types of Workbench Tools” for a list of the tools that are available from the **Tools** menu.

Types of Workbench Tools

The following tools are available in Workbench:

- Alarm portal
Explains the options that appear in the alarm portal. Refer to “Alarm portal”.
- Bacnet Service
Explains the options that appear in the Bacnet service. Refer to “Bacnet Service”.
- Certificate Management
Explains the Certificate Management view which you use to manage PKI (Public Key Infrastructure) and self-signed digital certificates to secure communication within a NiagaraNetwork. Certificates secure TLS connections to the host.
Refer to the document “*Station Security Guide*” for complete details.

- **Certificate Signer Tool**

Explains the options in the Certificate Signer Tool. You use this tool to sign intermediate digital certificates. Refer to the document "*Station Security Guide*" for complete details.
- **Driver Upgrade Tool**

(Appears if the `driverUpgrade` module is in the local `!modules` folder). Provides a wizard to upgrade driver modules in a remote JACE host, including making any necessary changes in the installed station's database (`config.bog` file) that may be required by the upgraded modules. Usage requires entering valid platform credentials and station credentials for the remote JACE.
- **Lexicon Tool**

Explains the options that appear in the lexicon tool. For general information, see "Lexicon Tool". For detailed information, see the *Niagara 4 Lexicon Guide*. For information about the lexicon installer, refer to the "Lexicon Installer" section of the *Niagara 4 Platform Guide*.
- **Logger Configuration**

An effective tool for troubleshooting and debugging station communications problems. Refer to "About the Logger Configuration tool".
- **Local License Database**

Explains the Workbench License Manager view. Refer to "Local License Database".
- **Lonworks Service**

Explains the options that appear in the lonworks service manager. Refer to "Lonworks Service".
- **Lon Xml Tool**

Describes the Lon Xml Create view. Refer to "Lon Xml Tool".
- **Manage Credentials**

Explains the options that appear in the credentials manager. Refer to "Credentials manager".
- **New Driver wizard**

Explains the options that appear in the new driver wizard. Refer to "New Driver wizard".
- **New Module wizard**

Explains how to modify the attributes of a new module. Refer to "New Module wizard".
- **New Station wizard**

Explains the options in the new station wizard how to change the default attributes of a new station. Refer to "New Station wizard".
- **Request License**

Explains the options that appear in the license request dialog box. Refer to "Request License".
- **Resource Estimator**

Explains the options that appear in the resource estimator dialog box. Refer to "Resource Estimator".
- **Time Zone Database Tool**

Provides ways to explore the contents of the local `timezones.jar` available to Workbench (if a station is running locally, say on a Supervisor, it uses these time zone definitions as well). Refer to "Time Zone Database Tool".
- **Todo List**

Explains the options that appear in the todo list dialog box. Refer to "Todo List".
- **Workbench Job Service**

Explains the options that are available in the Workbench Job Service view. Refer to “Workbench Job Service”.

- Workbench Service Manager

Explains the options that are available in the Workbench Service Manager view. Refer to “Workbench Service Manager”.

Alarm portal

The Alarm Portal tool allows you to view and acknowledge alarms from many different stations using a single viewer (portal). The alarm portal, shown, is divided into two resizable panes, and a set of buttons along the bottom of the alarm portal window: You can add alarm consoles from different stations by right-clicking in the Alarm Console Monitor (top pane) and choosing the **Add Alarm Console** menu selection to initiate the **Add Alarm Console** wizard.

From the Alarm Portal, double click on any alarm listed in the Open Alarm Sources pane to see the **Alarm Details** dialog box and the alarm record dialog box, as described in “About the alarm console”.

Figure 90 Alarm portal

The screenshot shows the Alarm Portal interface with two main panes and a control bar at the bottom.

Alarm Console Monitor (2 objects)

Station	Port	Console Ord	Status	Last Connected Time	Last Disconnected Time
ip:10.10.8.160	1912	slot:/Services/Alarming/ConsoleRecipient	Connected	17-Nov-04 9:06 AM	null
ip:10.10.8.209	1911	slot:/Services/AlarmService/ConsoleRecipient	Connected	17-Nov-04 9:07 AM	17-Nov-04 9:07 AM

Open Alarm Sources (15 objects)

Timestamp	Source State	Ack State	Ack Required	Source
17-Nov-04 8:31:13 AM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
17-Nov-04 7:21:13 AM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
17-Nov-04 6:53:13 AM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
17-Nov-04 1:59:13 AM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
17-Nov-04 12:59:43 AM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 11:46:13 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 9:26:13 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 9:08:43 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 7:41:13 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 7:06:13 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 6:52:13 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 5:59:43 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
16-Nov-04 4:56:43 PM	Normal	0 Acked / 1 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange
17-Nov-04 6:25:13 AM	Offnormal	0 Acked / 2 Unacked	true	local: station: slot:/AlarmLab/SimJace_Numeric/OutOfRange

Control bar buttons: Acknowledge, Hyperlink, Notes, Silence, Filter

- Alarm Console Monitor

This pane displays information about the consoles that are being monitored and contains the following information areas:

- Title bar

Displays “Alarm Console Monitor” title on the left and the number of consoles that are connected to the alarm portal on the right side of the title bar.

- Column headings

Displays the title of each of the columns that are displayed in the alarm console monitor.

- Alarm console list

Each row in the alarm console listing contains information about alarm consoles that are currently listed as available to monitor from the alarm console. Station identity information is displayed, as well connection status (connected or disconnected) and connect and disconnect times.

- Open Alarm Sources

This pane displays information about individual alarm sources, as described below. If you double-click on a single alarm source, the Open Alarm Sources view opens. This view is described in "About the alarm console".

- Title bar

Displays "Open Alarm Sources" title on the left and the total number of points that are displayed in the list on the right side of the title bar.

- Column headings (click column heads to sort)

Displays the title of each of the columns in the open alarm sources list, as follows:

- Timestamp

displays the latest time that the point generated an alarm.

Source State

displays the current state of the point (for example, Normal or Offnormal).

Ack State

displays how many alarms have been generated at the point and how many of those alarms have been acknowledged.

Ack Required

displays "true" or "false" indicating that an acknowledgement is or is not required for that alarm.

Source

displays the path to the point that is generating the alarm.

- Alarm portal buttons

- Acknowledge

Acknowledges all alarms at the selected source when an alarm is selected in the alarm source pane.

- Hyperlink

is active when an alarm extension property (refer to "About alarm extension properties". When clicked, displays the alarm source.

- Notes

displays a dialog box that allows you to add notes to an alarm record (see "About the console recipient").

- Silence

when clicked, this button quiets the alarm sound for the selected point.

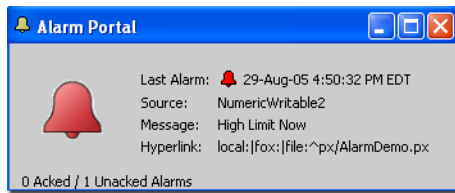
- Filter

displays the alarm filter dialog box for limiting the alarms that are displayed. See "About the alarm console".

The alarm portal tool, when enabled, also places the alarm icon in your system tray and an alarm popup window.

You can set options for the alarm portal in the **Tools→Options** menu, as described in "Alarm Portal options".

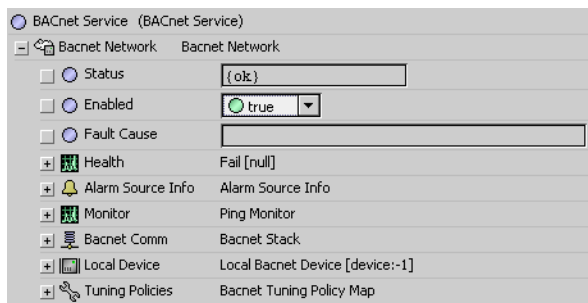
Figure 91 Alarm portal popup window



Bacnet Service

Following is an example of the **Bacnet Service** property sheet. Select **Tools**→**BACnet Service** from the Workbench main menu to add this service to the “My Tools” node of the nav tree. The default view is the property sheet view.

Figure 92 Bacnet service property sheet view



NOTE: You can use this Workbench Tools service to configure a local Bacnet device in the same way as a you would under a station, as described in the *Bacnet Guide*. However, any configuration that you do using the Workbench Bacnet Service (under the “My Tools” node) will be lost when you close Workbench. Use Bacnet Service under a station to save your Bacnet Service settings to a database.

Refer to the *Bacnet Guide* for a detailed description of the Bacnet service.

Lexicon Tool

For general information on lexicons, see “About lexicons”. For detailed information on using the Lexicon Tool, see the *Niagara 4 Lexicon Guide*. For details on installing (edited) file-based lexicons in remote JACE platforms, see the *Niagara 4 Platform Guide* section “Lexicon Installer”. For details on installing (new/edited) module-based lexicons in remote JACE platforms, see the *Niagara 4 Platform Guide* section “Software Manager”.

For an overview of the Lexicon Tool views, see the following sections:

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 93 Lexicon report view

Check boxes at the top of report view allow you to “hide” values that could affect “missing” status counts, as described in the *Niagara 4 Lexicon Guide* section “Lexicon components”.

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 detailed information using the lexicon report view, see the *Niagara 4 Lexicon Guide* section “Lexicon Report View”.

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 94 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 Workbench 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 clicking the **Save** button in the Workbench toolbar.

For detailed information on using the lexicon editor view, see the *Niagara 4 Lexicon Guide*.

Lexicon Module Builder

The **Lexicon Module Builder** view (shown) available in the allows you to bundle multiple lexicon files into a module for ease of distribution. The lexicons available for use are those lexicon files (modulename.lexicon) located in the !lexicon folder. You can build new lexicon modules or replace existing ones.

Figure 95 Lexicon module builder view

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.

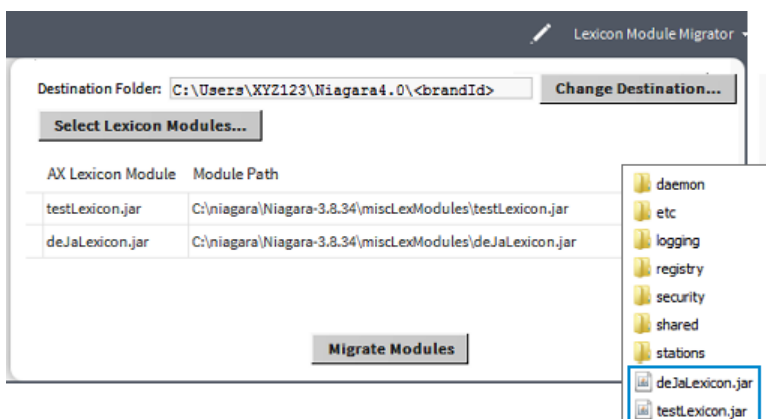
On completion of the build, you will see a confirmation message indicating that the module was constructed and placed in the !modules folder.

For detailed information on using the lexicon module builder view, see the *Niagara 4 Lexicon Guide* section “Lexicon Module Builder View”.

Lexicon Module Migrator

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

Figure 96 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.

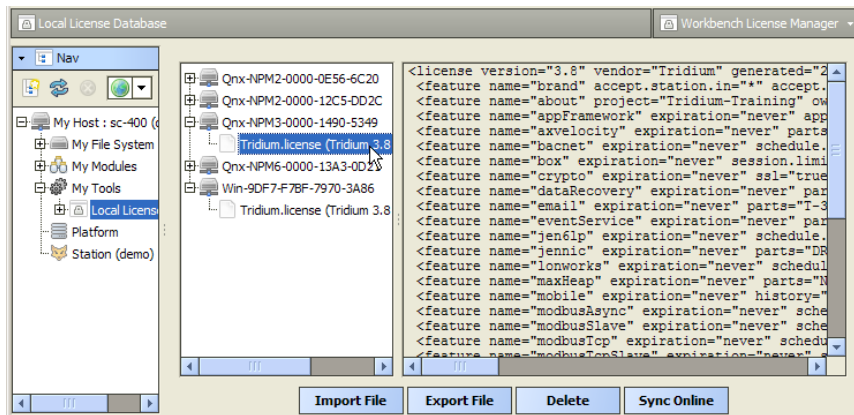
NOTE: For any modules that are new in Niagara 4.0, such as webEditors, etc., you need to build new lexicon modules using the **Lexicon Module Builder** view.

For detailed information on using the Lexicon Module Migrator, see the *Niagara 4 Lexicon Guide*.

Local License Database

The **Workbench License Manager** view is available via the Workbench Tools menu, by selecting **Local License Database**.

Figure 97 Workbench License Manager



As shown, this view lets you browse and manage the contents of your “local license database.”

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 File**
Always available, this allows you to add license file(s) from a local license file or license archive (.lar) file.
- **Export File**
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 Workbench local license database.
- **Sync Online**
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 more complete details about the license database, see “Workbench license manager” in the *Niagara 4 Platform Guide*.

Logger Configuration tool

The Workbench **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 a Workbench connection, this capability allows you to enable/disable loggers using a web browser.

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

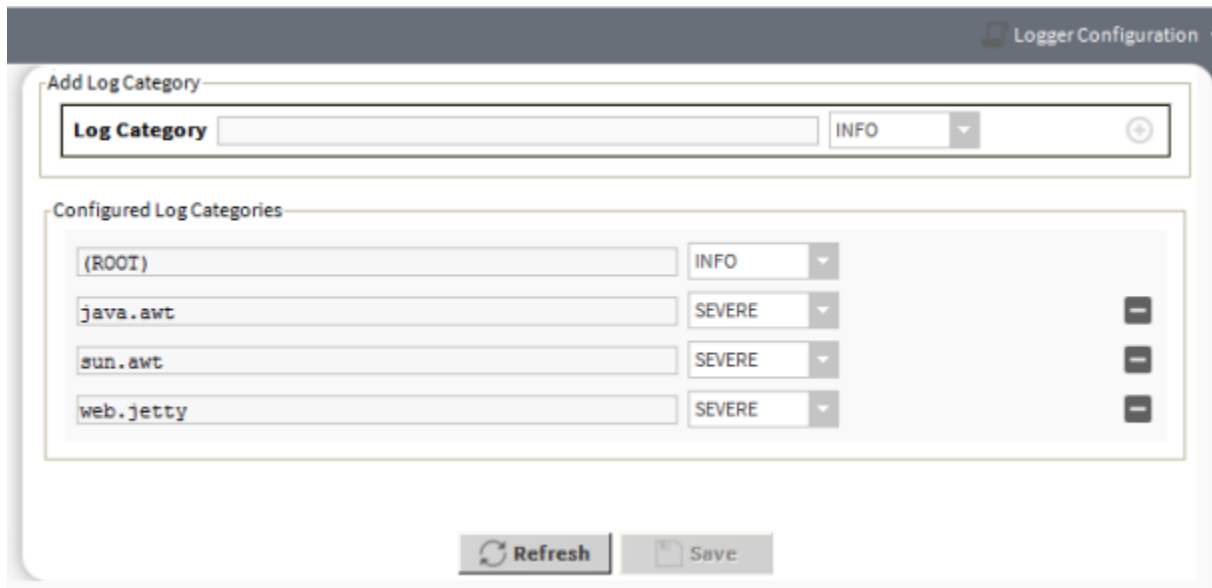
The Logger Configuration tool has seven severity levels which differs from the number of log levels used in NiagaraAX. Log levels used are mapped as shown here:

Table 1 NiagaraAX vs. Niagara 4 Severity Level Mapping

NiagaraAX Log Levels	Niagara 4 Log Levels
	Off
Error	Severe
Warning	Warning
Message	Info
	Config
Trace	Fine
	Finer
	Finest
	All

The **Logger Configuration** view is the main view for the Workbench Logger Configuration tool, as well as for the station DebugService.

Figure 98 Logger Configuration view



Logs for the connected station are visible in the platform **Application Director** view. While logs for Workbench are visible in a 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" then 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 99

Index | logSetup | DEFAULT-alarm Sample [!]

The Log alarm has been set to default Log level

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:`

Configuring settings for local Workbench logs


Use the **Logger Configuration** tool in Workbench Tools 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 Workbench, select **Tools**→**Logger Configuration**
The **Logger Configuration** view appears.
2. Click  (remove), located to the right of the log category you wish to remove.
The log category is immediately removed.
3. 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

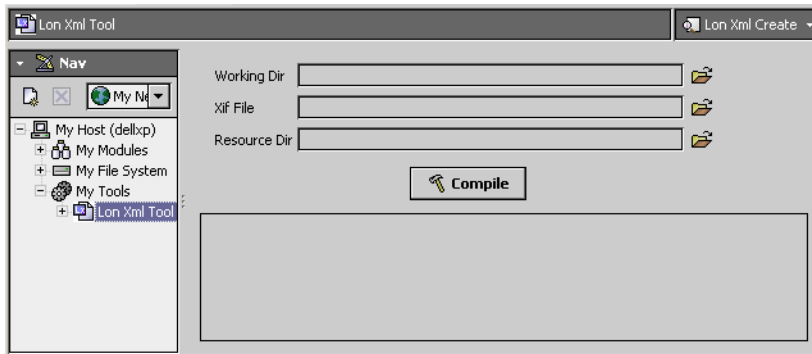
To view logged data for Workbench:

- Open a 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 (.lnml) file for a device, using the source .xif file and (if necessary) other resource files, as available from the device’s manufacturer. The tool’s dialog provides the necessary input fields.

Figure 100 Lon Xml Create tool view



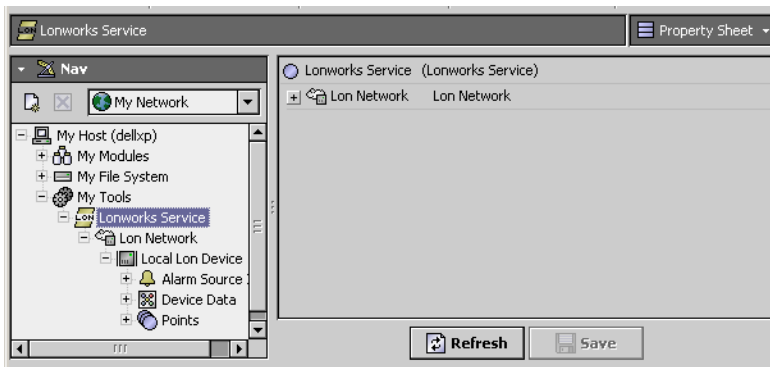
Refer to the *Lonworks Guide* for details about using this view and about the following:

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

Lonworks Service

The Lonworks Service is applicable if your Workbench PC has a Lonworks adapter. You can view it by selecting **Lonworks Service** from the **Tools** menu. It provides the identical “LonNetwork” access as if you had a local (PC) station running.

Figure 101 Lonworks Service tool in Workbench



For example, you can use the **Lon Device Manager** to discover, add, and upload Lon devices, examine nvs and ncis as LonComponents, and even perform writes and do other configuration (make bindings between devices, commission devices, etc.).

CAUTION: Any configuration you perform is not stored (persisted) in a station database, as this is “station-less” access (modeled completely in RAM). When you close Workbench, all modeling is lost—consider the Lonworks Service like a “hand held device” in this respect.

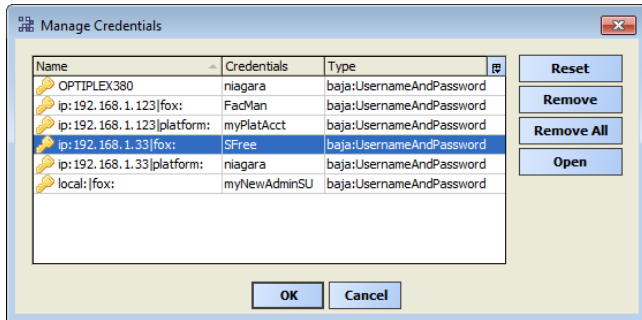
For this reason, do not use this tool to access a Lonworks network already managed by a station, but instead open that station, and access its LonNetwork.

Refer to the *Lonworks Guide* for more details.

Credentials manager

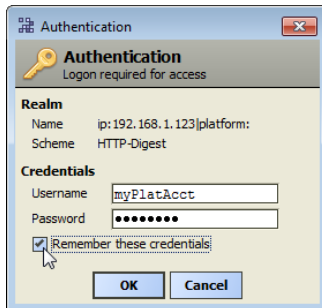
The **Manage Credentials** tool, or “Credentials Manager” provides a dialog box to access and open any previously “remembered” (cached) connections from your Workbench, including both platform and station connections.

Figure 102 Manage Credentials dialog (credentials manager)



You can also remove or reset any cached credentials. Note that your credentials cache is populated whenever you have the login (Authentication) dialog option “Remember these credentials” (check box) enabled.

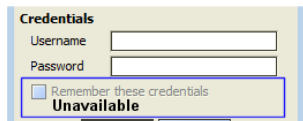
Figure 103 Authentication (Remember credentials)



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 of Workbench, or going to the credentials manager. Then, the login authentication dialog has the cached credentials already entered, and you simply click **OK**.

If you want tighter security for any platform or station connection from your Workbench PC, you should clear this check box whenever opening (logging on) a platform or station. Furthermore, you should remove any related entries using the credentials manager. This way, that platform or station connection always requires full entry of both user name and password.

NOTE: You can globally disable user credentials caching in Workbench via **Tools→Options**, in the **General** menu. When **Allow User Credentials Caching** is set to false, the **Remember these credentials** check-box remains unavailable (dimmed) in any **Authentication** dialog.

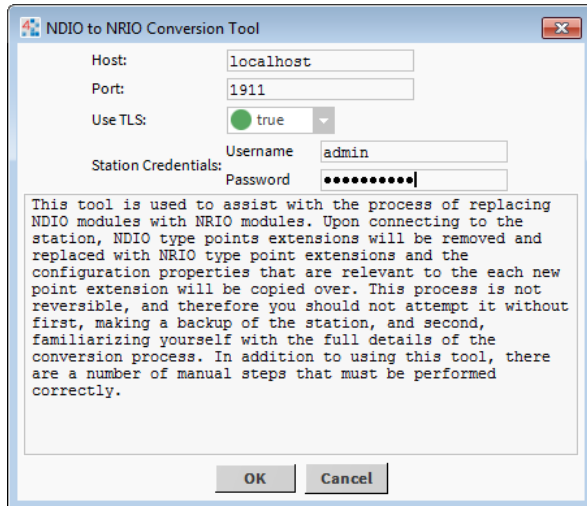


For related details, see “General options”.

NDIO to NRIO Conversion Tool

The **NDIO to NRIO Conversion Tool** is available when you select **Tools→NDIO to NRIO Conversion Tool** from the menu bar. This wizard provides a convenient way to replace an NDIO driver module with an NRIO driver module. This is useful if replacing a legacy JACE-2/3/6/7 with a JACE-8000. Legacy JACE models support both NDIO and NRIO, however the JACE-8000 only supports NRIO.

Figure 104 NDIO to NRIO Conversion Tool



Although NRIO and NDIO points have similar configuration properties, the points are not interchangeable. NRIO proxy extensions use different software than those in NDIO. So you cannot simply move points from NDIO modules to NRIO modules.

When run, the wizard replaces all NDIO type point extensions in the network with NRIO type point extensions, and the configuration properties that are relevant to each new point extension are copied as well. After running the tool there are several manual steps that you must perform in order to complete the conversion. Specifically, you need to move the converted points to the NRIO network, and then check for invalid addresses and correct those. Finally, you should verify that everything works as intended. For more details on how to use the tool, see *Converting NDIO modules to NRIO*.

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 a JACE-8000 which does not support NDIO.

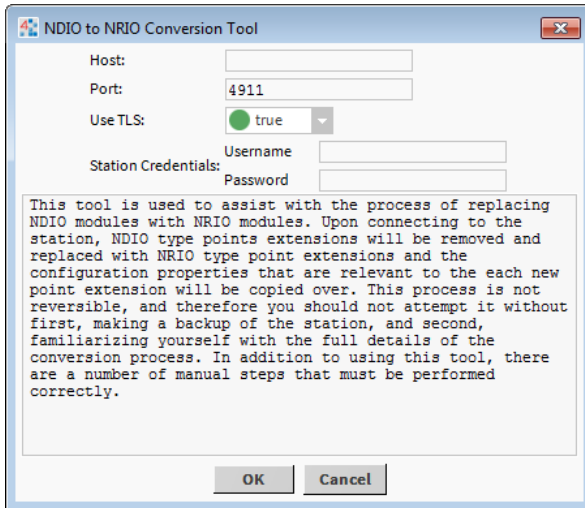
Prerequisites:

- nrioConversion-wb module installed
- Existing station running on Niagara 4 (regardless of whether or not the host supports NDIO). The station has an NDIO network with points.

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

- Step 1 Backup the N4 JACE station that contains the NDIO module(s) to be converted.
- Step 2 Add an NRIO network to the station, an NRIO device, and at least one NRIO CounterInputPoint (hardware point).
- Step 3 Save and reboot the JACE.
- Step 4 Once the reboot has finished, expand the Nav tree in Workbench and select an NDIO module to convert, and click **Tools→NDIO to NRIO Conversion Tool**.

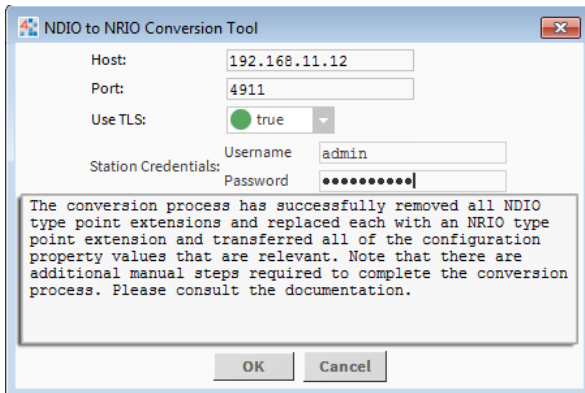
The **NDIO to NRIO Conversion Tool** wizard displays, as shown here.



Step 5 In the wizard dialog, enter the following connection values and click **OK**:

- **Host:** <IP address>
- **Username**
- **Password**

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



NOTE: The wizard actually converts the points for all of the NDIO modules even though you selected only one.

Step 6 Click **Cancel** to close the wizard.

At this time, all of the NDIO points are converted.

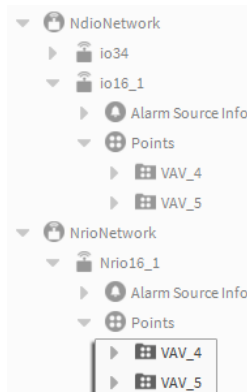
Step 7 Restart the station (if necessary, reboot JACE).

NOTE: A subsequent step requires that you manually move the converted NDIO points into the NRIO modules. Some designers prefer to add point folders to separate specific points by function. When using this approach, use the following steps to create the folders prior to moving the points from the NDIO network to the NRIO network.

Step 8 Expand an NRIO device, right-click on the **Points** folder and select **Views→Points Manager**. For example, the **Nrio16PointManager** view.

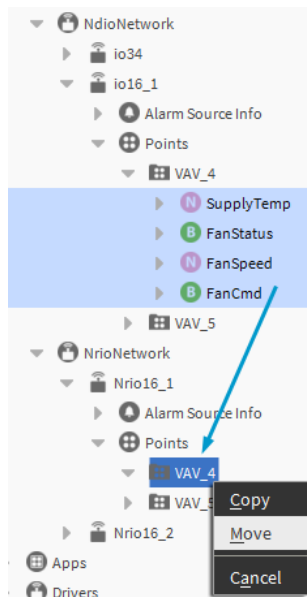
- In the **Points Manager** view, click **New Folder** and in the **Name** dialog enter the name for a corresponding NDIO Points Folder.

- b. Repeat the prior substep as needed to create all the necessary NRIO Points Folders, as shown in the following image.



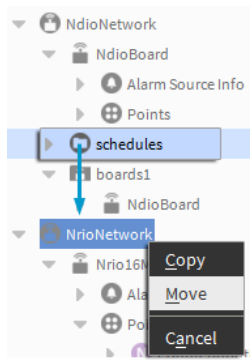
Step 9 In the Nav tree, expand an NDIO module's /Points/NdioPointFolder, select all of those points, and drag them while holding down the mouse right-button.

Step 10 Drop the points onto the corresponding NRIO Points Folders in an NRIO module, and click **Move** in the popup menu, as shown here.



IMPORTANT: Be sure to use the **Move** operation because it preserves data links, while cut-and-paste (or copy-and-paste) does not.

Step 11 Repeat the Move operation, as needed, to move all of the NDIO points and related non-point objects, i.e. schedules and control logic, to NRIO modules.



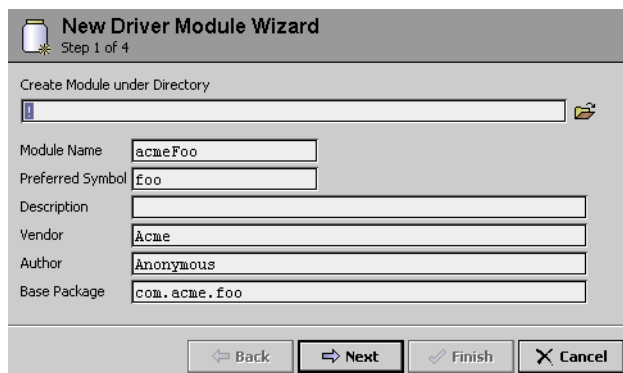
Step 12 Connect to your live station, access the **NRIO Device Manager** view and confirm that all modules and points are addressed correctly.

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 dialog boxes is shown below.

Figure 105 New driver wizard

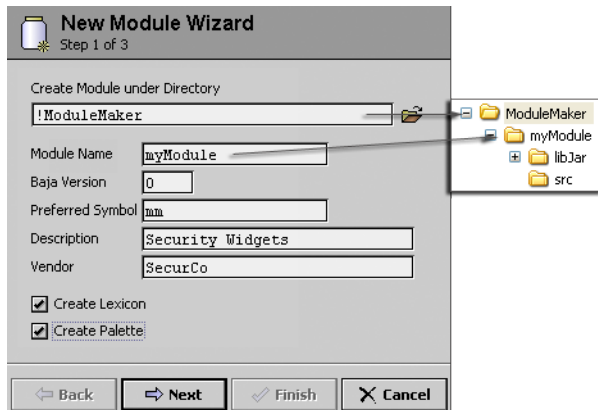


This wizard is provided as a convenient way to perform some of the steps necessary for creating a driver module. The four wizard dialog boxes allow driver developers to specify some basic driver module options and finish with the wizard creating a set of folders and files under a station directory that you assign during the wizard process. Also see the *Niagara 4 Developers Guide* for more detailed information about building modules.

New Module wizard

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

Figure 106 New module wizard creates folders under working directory



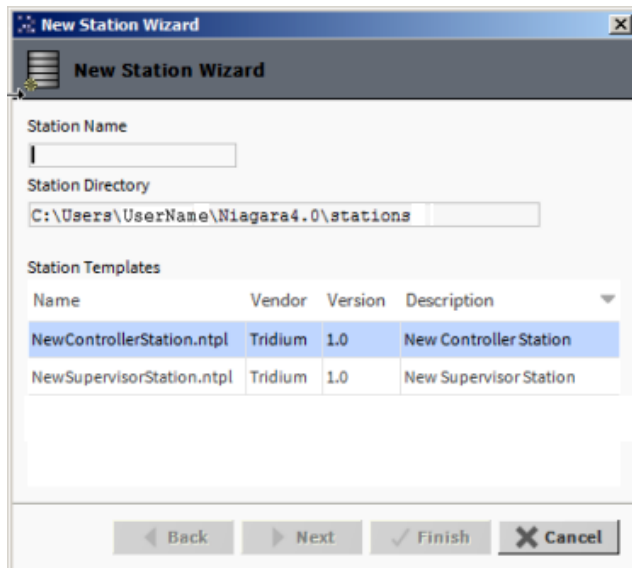
This wizard creates a set of folders and files under a station directory that you assign during the wizard process. After creating these files, you need to finish the process, as described in “Working with modules”. Also see the *Niagara 4 Developers Guide* for more detailed information about building modules.

New Station wizard

The **New Station Wizard** available in the **Tools** menu, uses templates to define the starting contents of a new station. By default, the wizard provides two station templates from which you can choose to create either a new controller (JACE) station or a new Supervisor station. Additionally, the list of templates shown in the wizard dialog 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: At present, there is no difference between the two default station templates (aside from the names), they create identical stations. However, the configuration of these templates may vary in future Niagara distributions or an OEM may distribute different default templates.

Figure 107 Workbench New Station tool



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 passwords”. 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 Workbench 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 Workbench 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 dialog appears notifying you of successful station creation.

Figure 108 Default station templates contain 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 dialog, 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.

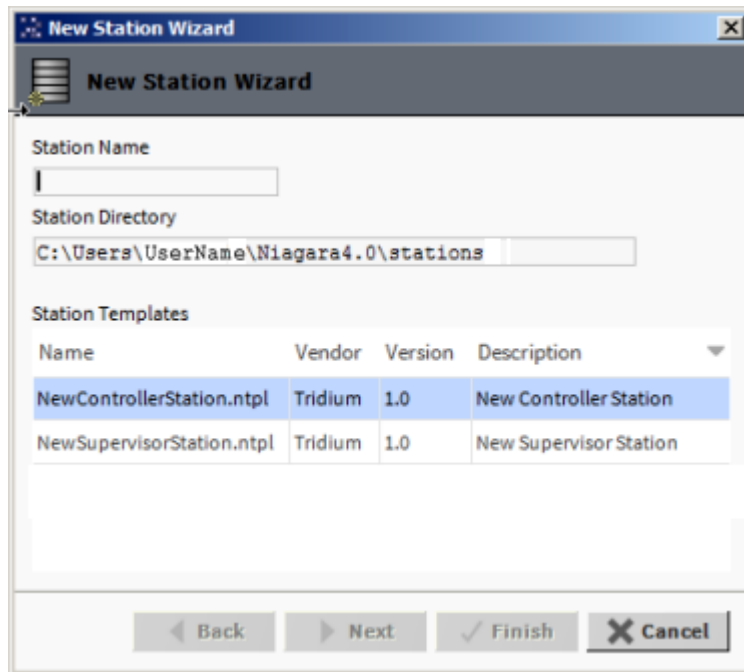
Both of the default station templates (controller, supervisor) have these two port configuration parameters. However, you (or an OEM who distributes their own workbench) can make a station template that provides different parameters or none at all. Whether the wizard dialog displays alternative parameters, 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 in Workbench **Tools** to create a new controller station or Supervisor station. The new station is automatically configured with appropriate services.

Step 1 In Workbench, select **Tools**→**New Station**

The **New Station Wizard** opens.



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 here. You can either delete the existing station or provide a different station name.

New Station Wizard

Station Name
myNewStation

Station Directory
C:\Users\UserName\Niagara4.0\stations\myNewStation

Delete existing station myNewStation? Yes

Station Templates

Name	Vendor	Version	Description
NewControllerStation.ntpl	Tridium	1.0	New Controller Station
NewSupervisorStation.ntpl	Tridium	1.0	New Supervisor Station

Back Next Finish Cancel

- b. 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**

Step 3 The second dialog prompts for the admin user password, configuration information relevant to the station template, and allows you choose an action to take once the station creation is completed.

- a. Enter **Admin Password** and **Admin Password Confirm**
- b. Modify the Foxs and HTTPS port numbers as needed.

NOTE: If the new Station Template selected in Step 1 contains any exposed configurable properties, such as the ports indicated in the above image, this dialog presents those properties allowing you to modify the values as needed.

- c. Select an option for the preferred action on completion.
- d. Click **Finish**.

The **New Station Wizard** closes. If the default option, `open it in user home`, is selected (as shown in the above image) a **Property Sheet** view of the new station `config.bog` file displays.

Creating a station template

Create a user-defined station template from a configured station that contains everything needed for the initial starting point of a new station. Once created, the station template is added to the list of selectable templates presented in the Workbench **New Station Wizard**.

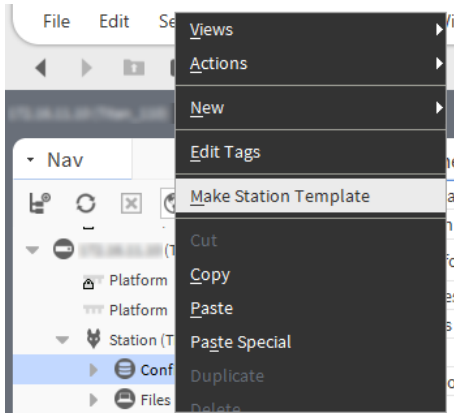
Prerequisites:

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

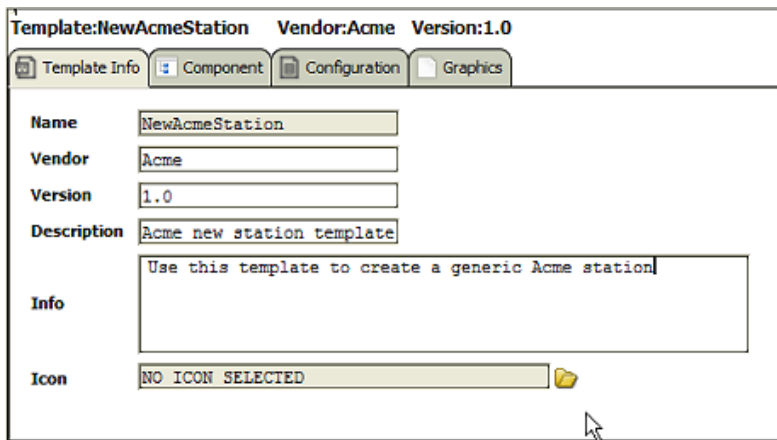
Station templates are automatically saved in the `stationTemplates` sub-directory of your Workbench User Home. User-defined station templates stored in this location are automatically included in the **Station Templates** table shown in the Workbench **New Station Wizard** and are available for selection when using the tool.

This procedure is one the Systems Integrator might perform on a fully configured “basic” station for purposes of reuse and standardization.

- Step 1 In the Nav tree, right-click on the station’s Config node and select **Make Station Template** (as shown here) to open the **Template** view.

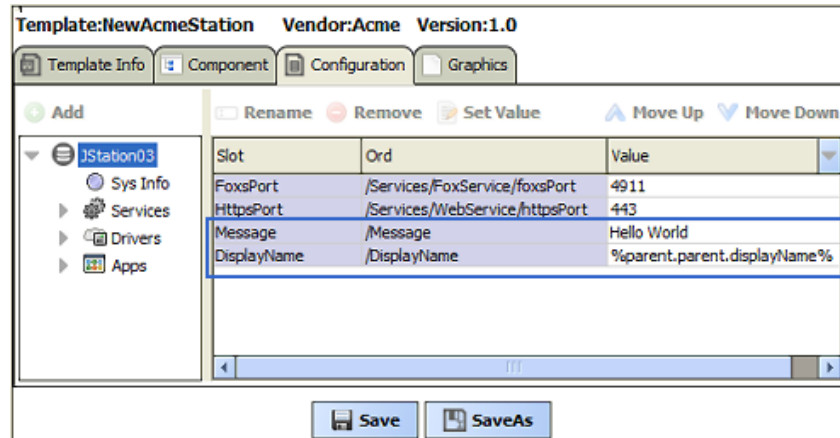


- Step 2 In the **Template** view, click each of the tabs to edit the station template as needed.
- a. On the **Template Info** tab, fill-in desired info about the template, as shown here.



- b. On the **Component** tab, make desired changes for the default station template.
- c. On the **Configuration** tab, add configurable parameters as needed. For example, to add the **Foxs Port**:
 - i. In the left pane, expand **Services**→**FoxService**→**Foxs Port**
 - ii. Double-click **Public Server Port** to add it to the right pane.
 - iii. Change the default value as needed and click **Set Value**.
 - iv. Click **Rename** and change the name to "Foxs Port".

NOTE: The value entered into the stationTemplate for the exposed property becomes the default value upon creating a new station.



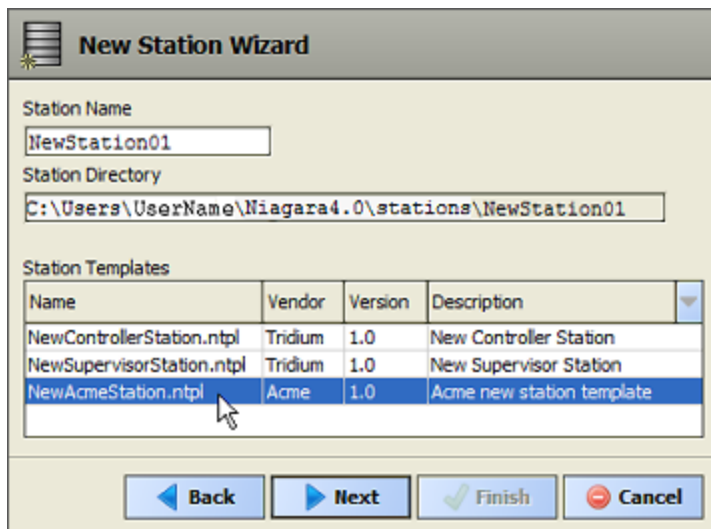
NOTE: Any exposed component properties defined in the template displays as configurable parameters in the **New Station Wizard** when this template is used.

- d. On the **Graphics** tab, make desired changes for the default station template.

Step 3 When finished modifying the template, click **Save**.

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

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

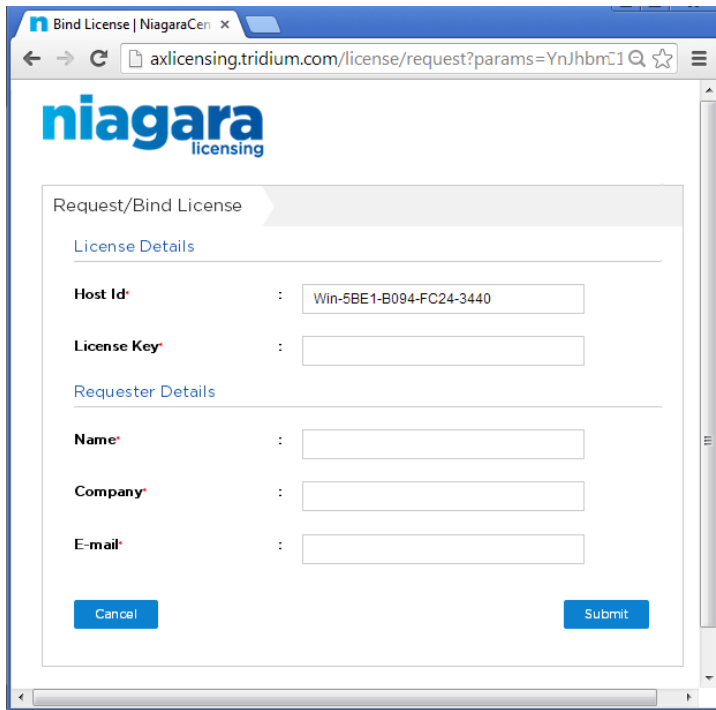


NOTE: A Workbench restart is not necessary in order for the user-defined station template to appear in the **New Station** wizard. The file simply needs to be saved to the `~stationTemplates` directory.

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 Workbench PC. See below.

Figure 109 License request form in browser (from Workbench, Tools > Request License)



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.

NOTE: For more information about licensing, see “License Tools and Files” in the *Niagara 4 Platform Guide*.

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 110 Resource estimator view

Additional Multiplier	Medium (30%)			
Device Networks	0	× 100,000 + Multiplier		0.000 kRU
Devices	0	× 5000 + Multiplier		0.000 kRU
Proxy Points	0	× 250 + Multiplier		0.000 kRU
Program Components	0	× 3000 + Multiplier		0.000 kRU
Histories				
Count * (250 + Size/10)				
Config. 1	Count	0	Capacity	Record Count
		0		Size
		0		0.000 kRU
Config. 2	Count	0	Capacity	Record Count
		0		Size
		0		0.000 kRU
Config. 3	Count	0	Capacity	Record Count
		0		Size
		0		0.000 kRU
Config. 4	Count	0	Capacity	Record Count
		0		Size
		0		0.000 kRU
Config. 5	Count	0	Capacity	Record Count
		0		Size
		0		0.000 kRU
AlarmDb Capacity	0	× 1		0.000 kRU
Total				
				0.000 kRU
<input type="button" value="Calculate"/>				

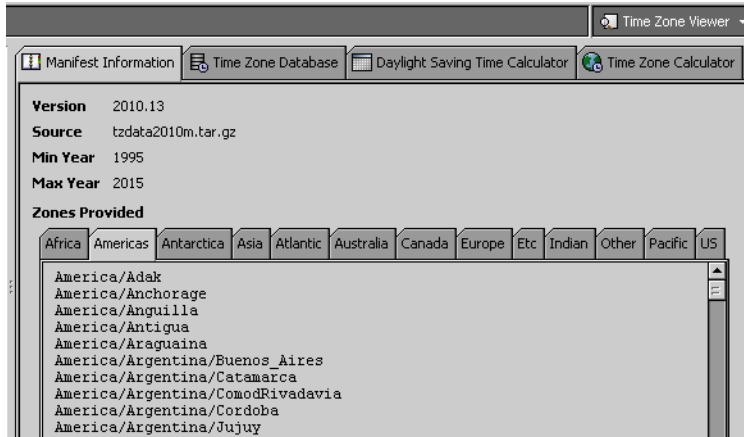
Time Zone Database Tool

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, refer to “Time Zones and Niagara” in the *Niagara 4 Platform Guide*.

Figure 111 Time Zone Database Tool in Workbench



Manifest Info

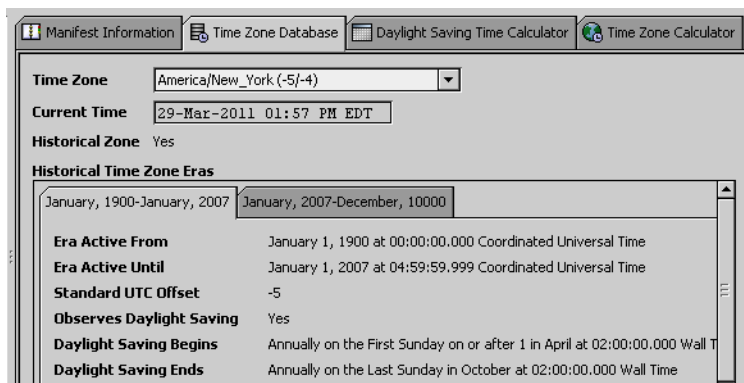
This tab in the **Time Zone Database Tool** shows you the `timezone.jar` file version, and which “Olsen Time Zone DB” version the file is based upon. The “Min Year” says how far back historically time zones are guaranteed to behave correctly. Conversely, “Max Year” is how far forward behavior should be correct.

The main view is a tabbed breakdown of all time zone definitions by world region, where the list of time zones in each region is listed alphabetically.

Time Zone Database

This tab in the **Time Zone Database Tool** lets you see how any selected time zone behaves.

Figure 112 Time Zone Database tab in Time Zone Database Tool



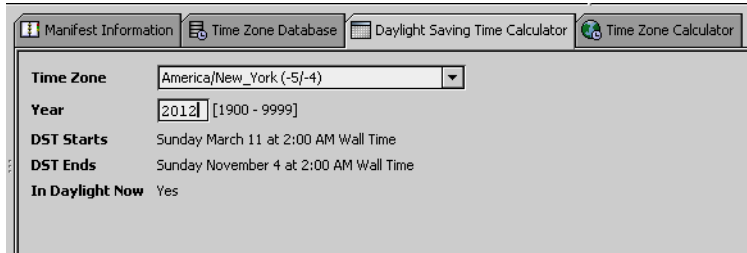
For example, what is the current time? (note that a refresh is required for update)

Is this a “historical zone”? (meaning have time zone “rules” changed for this zone at some point?)
 Note that if “Yes”, there will be multiple “era” tabs in the lower definition area, each with the parameters for things like UTC offset, DST (daylight savings time) changeover, and so on—specific to that time era.

Daylight Savings Time Calculator

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

Figure 113 Daylight Savings Time Calculator tab in Time Zone Database Tool

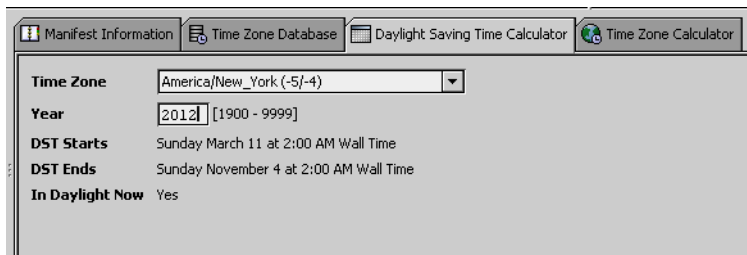


To use, select a time zone and type in a year in the **Year** field.
 Press Enter to update the “DST Starts” and “DST Ends” time fields.

Time Zone Calculator

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

Figure 114 Time Zone Calculator tab in Time Zone Database Tool



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 115 Todo list view

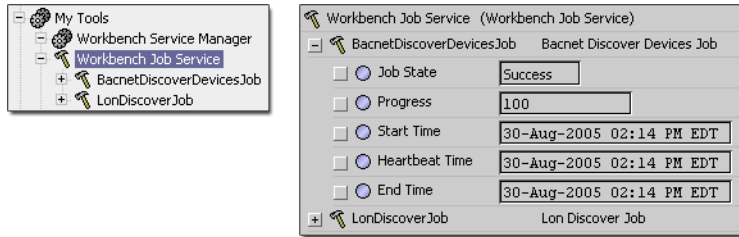
Summary	Group	Priority	Date
Commission Jace	Extra	0	17-Aug-05 11:46 AM EDT
Create New Station	General	1	17-Aug-05 11:46 AM EDT
Set up Alarm Portal	Extra	2	17-Aug-05 11:46 AM EDT
Reschedule Energy Slowdown	General	3	17-Aug-05 11:46 AM EDT

The Todo list is a tabular view with standard table controls and options, as described in “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 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 116 Job service nav tree and property sheet view

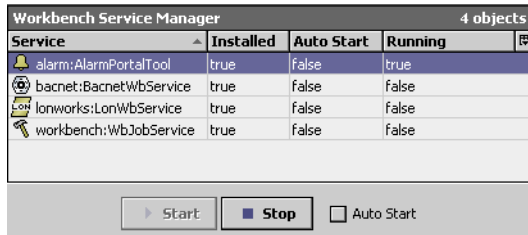


NOTE: Workbench jobs are jobs that are initiated and run under the Workbench - not under a station. Jobs that are run under a station are monitored and displayed in the Station Job Service (under the station Services node in the nav side bar).

Workbench 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 117 Workbench service manager view



The Workbench service manager is a tabular view with standard table controls and options, as described in “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 alarm module
- ◆ Components in alarmRdb module
- ◆ Components in backup module
- ◆ Components in baja module
- ◆ Components in chart module
- ◆ Components in control module
- ◆ Components in converters module
- ◆ Components in crypto module
- ◆ Components in email module
- ◆ Components in file module
- ◆ Components in help module
- ◆ Components in history module
- ◆ Components in net module
- ◆ Components in onCall module
- ◆ Components in program module
- ◆ Components in the Sms module
- ◆ Components in schedule module
- ◆ Components in timesync 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 “Alphabetical list of kitControl components” section in the *Kit-Control Guide*.

Summary information is provided on components in the following modules:

- alarm
- alarmRdb
- backup
- baja
- chart
- control
- converters
- crypto
- email
- file
- flr
- help
- history
- net
- onCall

- program
- schedule
- sms
- timesync
- web
- workbench

Components in alarm module

- AlarmClass
- AlarmClassFolder
- AlarmConsoleOptions
- AlarmPortalOptions
- AlarmService
- AlarmSourceExt
- AlarmSourceInfo
- BooleanChangeOfStateAlarmExt
- BooleanCommandFailureAlarmExt
- ConsoleRecipient
- FaultAlgorithm
- EnumChangeOfStateAlarmExt
- EnumCommandFailureAlarmExt
- LinePrinterRecipient
- MemoryAlarmService
- OffnormalAlgorithm
- OutOfRangeAlarmExt
- PrinterRecipient
- StationRecipient
- StatusAlarmExt
- StringChangeOfValueAlarmExt

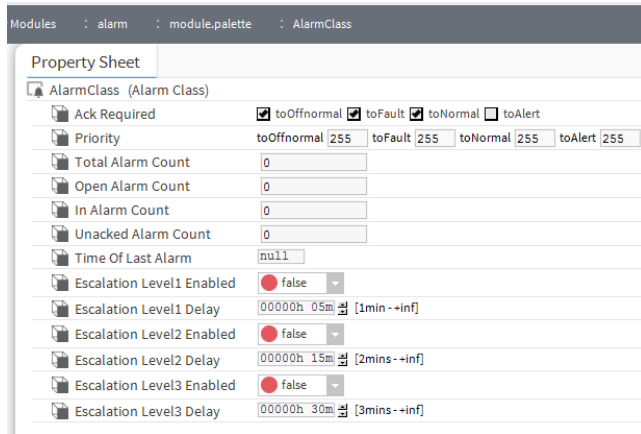
alarm-AlarmClass (DefaultAlarmClass)

An AlarmClass object is used to group alarms that have the same routing and handling characteristics. The AlarmClass is available in the alarm palette.

The alarm class:

- Routes alarms with some similar set of properties along common routes that serve as channels for like data.
- Manages the persistence of the alarms as needed via the alarm archive, but otherwise merely chains alarms from the alarm source via a topic.
- Manages which alarms require acknowledgement.
- Is the basis for visual grouping in the alarm console.

Figure 118 Alarm Class



Property	Value	Description
Ack Required	true or false	The alarm must be acknowledged.
Priority [on-call contact]	1–255 for each transition, default: 255; %priority% on a report	Specifies the order in which the <code>OnCallService</code> sends alarm notifications to the <code>OnCallContact</code> . Priority levels are indicated graphically by colors and are set up using the alarm options dialog box. The contact with the lowest number (highest priority) receives notification first. An alarm that is not acknowledged within the designated time is forwarded to the next contact in the list. NOTE: Contacts may share the same Priority number. The <code>OnCallService</code> sends an identical notification to all contacts that have the same priority number
Total Alarm Count	read-only	Displays the total number of alarms assigned to the alarm class from all sources.
Open Alarm Count	read-only	Displays the current total number of alarms that are unacknowledged and normal or unacknowledged and an alert.
In Alarm Count	read-only	Displays the total number of alarm conditions.
Unacked Alarm Count	read-only	Displays the total number of unacknowledged alarms.
Time of Last Alarm	read-only	Displays the time that system generated the last alarm assigned to this alarm class.
Escalation Level(n) Enable	false check box, where n is 1, 2 or 3	The escalation level defaults to “true” (enabled). Selecting this check box turns this escalation level off.
Escalation Level(n) Delay	hours and minutes; one minute is the smallest increment you can set for this property.	Sets the time between alarm generation and escalation. It is not the time between escalation levels. Set a time to allow an unacknowledged alarm to remain unacknowledged before you escalate it to the next level.

alarm-AlarmClassFolder

This is a container object provided for organizing groups of alarm class objects. The `AlarmClassFolder` is available in the alarm palette.

alarm-AlarmConsoleOptions

This component configures the alarm console options. You access these options by in Workbench clicking **Tools**→**Options**→**Alarm Console** .

The component is stored under the `/users/{user}/options` directory.

Property	Value	Description
Notes Required on Ack	true or false	true opens the Notes window when you initiate an alarm acknowledgement from the alarm console.
Sounds Enabled	true or false	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	true causes an alarm to repeat continually until it is acknowledged or cleared. This option works together with the Continuous Alarm Delay property. You can also set this value under the Alarms menu in the Workbench main menu when the Alarm Console view is active.
Continuous Alarm Delay	hours minutes seconds	true interrupts the sound of the alarm for a time equal to the value of this property.
Low Priority Color	color	Defines the color to use for the least important alarms. Alarm priorities are numbered from 1–255. The system assigns colors to alarm priorities that fall between priority levels on a color-scale along a path defined by the three assigned colors.
Mid Priority Color	color	Defines the color to use for an alarm of medium importance. Alarm priorities are numbered from 1–255. The system assigns colors to alarm priorities that fall between priority levels on a color-scale along a path defined by the three assigned colors.
High Priority Color	color	Defines the color to use for the highest priority (priority 1). Alarm priorities are numbered from 1–255. The system assigns colors to alarm priorities that fall between priority levels on a color-scale along a path defined by the three assigned colors.
Time Zone Display	Console Source	Displays the alarm record timestamp in the time zone of the alarm console view (Console) or in the time zone of the alarm source (Source).
Alarm Class Mapping	additional options	Opens the Alarm Class Mapper window, which allows you to add alarm classes and map them.

Property	Value	Description
Alarm Ack Responses		Creates one or more text entries that you can use to populate the Notes window when acknowledging an alarm. When 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. To add, edit, or remove response options, use the <code>>></code> button to open the associated Edit window. When Notes Required on Ack is set to <code>false</code> , these Alarm Ack Responses are not visible.
View Instructions	<code>true</code> or <code>false</code>	<code>true</code> causes the alarm Instructions pane to display across the bottom of the Alarm Console. Instructions display in the pane for any single selected alarm that has associated instructions.

alarm-AlarmPortalOptions

This component allows you to configure the alarm portal parameters. The **Options** dialog box displays when you select the Workbench **Tools**→**Options**→**Alarm Portal** menu item.

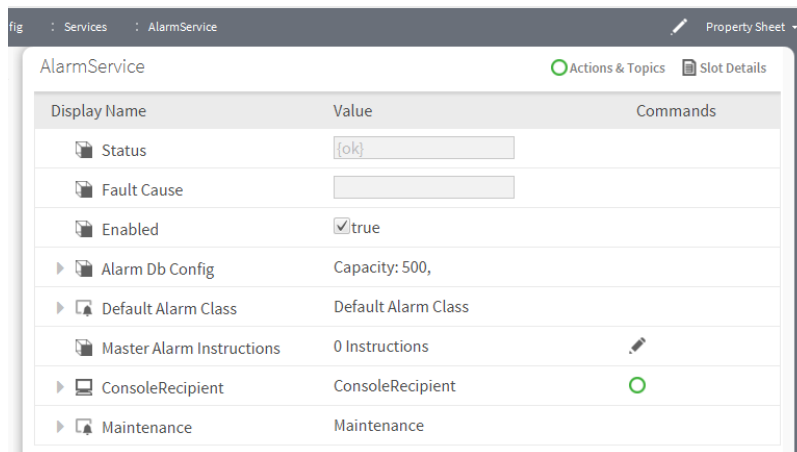
The component is stored under the `/users/{user}/options` directory.

Property	Value	Description
Tray Icon Enabled	<code>true</code> or <code>false</code>	<code>true</code> displays an alarm icon in the system tray when the alarm portal is active.
Alarm Popup Enabled	<code>true</code> or <code>false</code>	<code>true</code> displays an alarm popup window when the alarm portal is active.
Alarm Popup Always on Top	<code>true</code> or <code>false</code>	<code>true</code> causes the alarm popup window to stay on top of other windows when the alarm portal is active.
Alarm Popup Uncloseable	<code>true</code> or <code>false</code>	<code>true</code> prohibits the operator from closing the alarm popup window when the alarm portal is active.
Kiosk Mode	<code>true</code> or <code>false</code>	the alarm portal opens in Kiosk Mode the next time it is started. This changes the display.
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 Time Range	list of options	Selects the most common interval for displaying alarm data.

alarm-AlarmService

This component uses **AlarmClasses** to route all alarm messages between **AlarmSources** and **AlarmRecipients**. Each station contains a single **AlarmService**, which is available in the **alarm** palette.

Figure 119 AlarmService property sheet



Property	Value	Description
Status [component]	read-only text: ok disabled fault	Displays the current state of the component or extension. The platform connection will be in <code>fault</code> if any of the following occurs: <ul style="list-style-type: none"> Supervisor has no <code>ProvisioningNwExt</code> under its <code>NiagaraNetwork</code> (for example, it has been deleted). Supervisor is not licensed for provisioning. <code>NiagaraStation</code> is in <code>fault</code>. The station's platform daemon rejects the platform connection's credentials. The extension is <code>disabled</code> if its <code>Enabled</code> property is set to <code>false</code> or the <code>ProvisioningNwExt</code> is disabled.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Enabled [general]	<code>true</code> or <code>false</code>	Activates and deactivates use of the function.
Db Config	heading	Provides access to database configuration properties. See Db Config, page 137 .
Default Alarm Class	heading	Defines basic alarm properties, reports alarm counts and establishes escalation levels. See Default Alarm Class, page 137 .
Master Alarm Instructions	app	Clicking the edit icon (pencil) to the right opens a window for adding and managing alarm instructions. See Master Alarm Instructions, page 138 .
Console Recipient	heading	Provides access to console recipient properties. See Console Recipient, page 139 .
Maintenance	heading	Provides access to maintenance properties. See Maintenance, page 140 .

Db Config

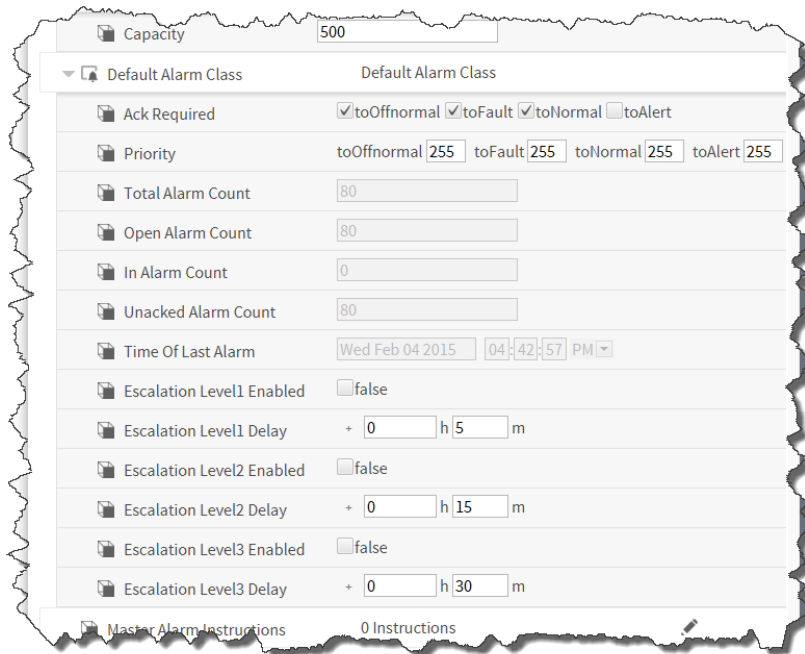
Figure 120 AlarmService DbConfig properties



Property	Value	Description
Capacity	1–250,000 records	Defines the number of alarm records to store in the histories database. When the capacity is reached, newer alarm records overwrite the oldest records.

Default Alarm Class

Figure 121 AlarmService DefaultAlarmClass properties

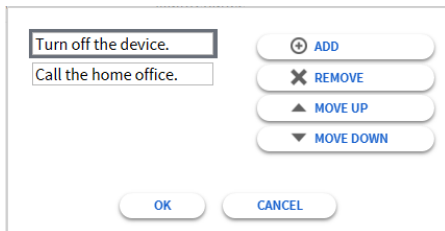


Property	Value	Description
Ack Required	true or false	The alarm must be acknowledged.
Priority [on-call contact]	1–255 for each transition, default: 255; %priority% on a report	Specifies the order in which the OnCallService sends alarm notifications to the OnCallContact. Priority levels are indicated graphically by colors and are set up using the alarm options dialog box. The contact with the lowest number (highest priority) receives notification first. An alarm that is not acknowledged within the designated time is forwarded to the next contact in the list. NOTE: Contacts may share the same Priority number. The OnCallService sends an identical notification to all contacts that have the same priority number

Property	Value	Description
Total Alarm Count	read-only	Displays the total number of alarms assigned to the alarm class from all sources.
Open Alarm Count	read-only	Displays the current total number of alarms that are unacknowledged and normal or unacknowledged and an alert.
In Alarm Count	read-only	Displays the total number of alarm conditions.
Unacked Alarm Count	read-only	Displays the total number of unacknowledged alarms.
Time of Last Alarm	read-only	Displays the time that system generated the last alarm assigned to this alarm class.
Escalation Level(n) Enable	false check box, where n is 1, 2 or 3	The escalation level defaults to "true" (enabled). Selecting this check box turns this escalation level off.
Escalation Level(n) Delay	hours and minutes; one minute is the smallest increment you can set for this property.	Sets the time between alarm generation and escalation. It is not the time between escalation levels. Set a time to allow an unacknowledged alarm to remain unacknowledged before you escalate it to the next level.

Master Alarm Instructions

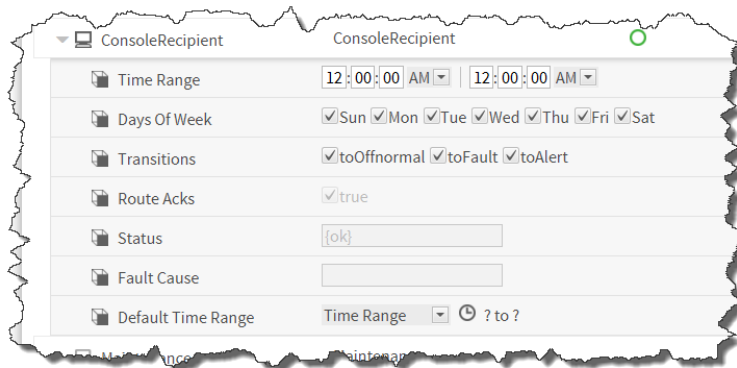
Figure 122 Master instructions



Button	Value	Description
Add	n/a	Adds an instruction to the list.
Remove	n/a	Deletes the selected instruction from the list.
Move Up/Down	n/a	Moves the selected instruction higher or lower in the list.

Console Recipient

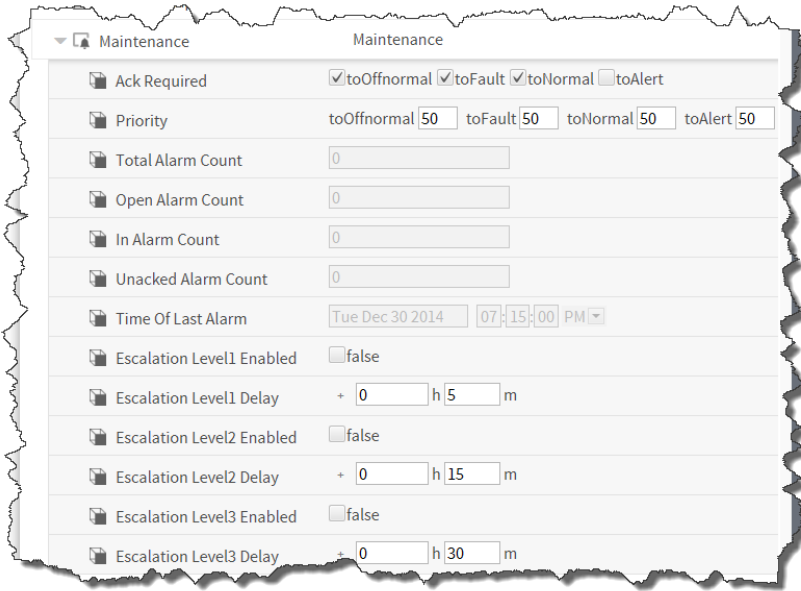
Figure 123 AlarmService ConsoleRecipient properties



Property	Value	Description
Time Range	Start Time and End Time	Start Time sets the time of day to begin the function (for example, trigger schedule, alarm event)
Days of the Week or Days of Week		
Transitions	Option boxes	Allow selection of specific alarm transitions to display in the console. Only those transitions that are selected will be displayed in the console - even though the alarms are still saved into the alarm history.
Route Acks	true or false	Enables and disables the routing of alarm acknowledgements to the recipient.
Status [component]	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> {ok} indicates that the component is polling successfully. {down} indicates that polling is unsuccessful, perhaps because of an incorrect property. {disabled} indicates that the Enable property is set to false. fault indicates another problem.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Default Time Range	drop-down list of time options	Provides a list of options for controlling how much information to display on the alarm console. If you select Time Range , the system prompts you for a beginning and ending time.

Maintenance

Figure 124 AlarmService Maintenance properties



Property	Value	Description
Ack Required	true or false	The alarm must be acknowledged.
Priority [on-call contact]	1–255 for each transition, default: 255; %priority% on a report	Specifies the order in which the OnCallService sends alarm notifications to the OnCallContact . Priority levels are indicated graphically by colors and are set up using the alarm options dialog box. The contact with the lowest number (highest priority) receives notification first. An alarm that is not acknowledged within the designated time is forwarded to the next contact in the list. NOTE: Contacts may share the same Priority number. The On-CallService sends an identical notification to all contacts that have the same priority number
Total Alarm Count	read-only	Displays the total number of alarms assigned to the alarm class from all sources.
Open Alarm Count	read-only	Displays the current total number of alarms that are unacknowledged and normal or unacknowledged and an alert.
In Alarm Count	read-only	Displays the total number of alarm conditions.
Unacked Alarm Count	read-only	Displays the total number of unacknowledged alarms.
Time of Last Alarm	read-only	Displays the time that system generated the last alarm assigned to this alarm class.

Property	Value	Description
Escalation Level(n) Enable	false check box, where n is 1, 2 or 3	The escalation level defaults to "true" (enabled). Selecting this check box turns this escalation level off.
Escalation Level(n) Delay	hours and minutes; one minute is the smallest increment you can set for this property.	Sets the time between alarm generation and escalation. It is not the time between escalation levels. Set a time to allow an unacknowledged alarm to remain unacknowledged before you escalate it to the next level.

Types of alarm extensions

Find the alarm extensions in palettes **alarm: Extensions** and **kitControl:Alarm**. This table lists all alarm extension types and the applicable point parents.

Alarm extension type(palette: Folder)	Applies to point types		General description
	Read-only	Writable	
OutOfRangeAlarmExt (alarm: Extensions)	NumericPoint	NumericWritable	Provides alarming based upon numeric alarm high and low limits. Includes configurable deadband.
	—	Any object with single numeric Out	For example, kitControl:Math object "Add"
StringChangeOfValueAlarmExt(alarm:Extensions)	StringPoint	StringWritable	Provides alarming based upon either inclusion or exclusion of the entered string value (or "regular expression," as needed).
	—	Any object with single String Out	For example, kitControl:String object "StringSubString"
BooleanChangeOfStateAlarmExt(alarm:Extensions)	BooleanPoint	BooleanWritable	Provides alarming based upon one of two possible values (states) as an alarm condition.
	—	Any object with single Boolean Out	For example, kitControl: Logic object "And."
BooleanCommandFailureAlarmExt (alarm:Extensions)	—	BooleanWritable	Provides alarming based upon mismatch between commanded value and actual (sensed) value. Extension has feedbackValue input property for linking.
EnumChangeOfStateAlarmExt (alarm:Extensions)	EnumPoint	EnumWritable	Provides alarming based upon one of multiple possible values (states) as an alarm condition.
EnumCommandFailureAlarmExt (alarm:Extensions)	—	EnumWritable	Provides alarming based upon mismatch between commanded value and actual (sensed) value. Extension has feedbackValue input property for linking.
StatusAlarmExt (alarm: Extensions)	Any type that accepts extensions	Any type that accepts extensions	Provides alarming based upon any combination of status flags, including overridden, null, etc.

Alarm extension type(palette: Folder)	Applies to point types		General description
	Read-only	Writable	
LoopAlarmExt (kitControl: Alarm)	—	LoopPoint	Sliding alarm limit for Loop-Point based upon controlled process deviation from setpoint.
ElapsedActiveTimeAlarmExt (kitControl:Alarm)	BooleanPoint with DiscreteTotalizerExt	BooleanWritable with DiscreteTotalizerExt	Provides alarming based upon accumulated runtime (elapsed active time). References a specific DiscreteTotalizerExt under same parent point.
	—	any object with single Boolean Out (also with a DiscreteTotalizerExt)	For example, kitControl: Logic object "And."
ChangeOfStateCountAlarmExt (kitControl:Alarm)	BooleanPoint with DiscreteTotalizerExt	BooleanWritable with DiscreteTotalizerExt	Provides alarming based upon accumulated COS (change of states). References a specific DiscreteTotalizerExt under same parent point.
	—	any object with single Boolean Out (also with a DiscreteTotalizerExt)	For example, kitControl: Logic object "And"

alarm-AlarmSourceExt

This component is the abstract super-class of all Baja control alarming algorithms. It is available in the alarm module. Alarm extensions are contained in the **alarm** palette.

To set up alarming on a component you add an alarm extension to the component’s property sheet. Alarm extension types must match their parent component type. For example, an **OutOfRangeAlarmExt** goes with a Numeric point type and a **BooleanChangeOfStateAlarmExt** goes with a Boolean point type.

Each alarm extension shares the same set of properties that allow you to specify the alarming conditions and certain routing options. Alarm extension properties define items such as alarm enable (annunciation) transition types, alarm delay times, associated alarm class, and alarm display text for different transition types. You define the actual alarm limits or state(s) in properties in the extension’s “**Offnormal Algorithm**” slot.

Property	Value	Description
Alarm Inhibit	true or false	<p>true prevents all alarm generation due to any transition or state change, thus preventing unintended alarms in after-hours situations when a piece of equipment is turned off. Inhibit Time qualifies this behavior.</p> <p>For example, if set to true and an Offnormal state is reached, a toOffNormal status is not communicated. When the state returns to Normal, a toNormal status also is not communicated.</p> <p>A difference between Alarm Inhibit and Alarm Delay is that the former is a boolean value (true/false) and may be controlled by another device (for example, the ON/OFF value of a fan).</p> <p>false allows alarm generation. This value prevents alarms from being inhibited (even if an Inhibit Time is set).</p>
Inhibit Time	hours minutes seconds	Controls the length of time that the current Alarm Inhibit state remains in effect after an Alarm Inhibit state change.

Property	Value	Description
		<p>When an Alarm Inhibit value changes from <code>true</code> to <code>false</code>, alarm generation continues to be inhibited for the time specified by the value set for Inhibit Time</p> <p>When an Alarm Inhibit value changes from <code>false</code> to <code>true</code>, alarm generation may continue to be inhibited for a time that is dependent on the point type. For discrete points, the system increases the Inhibit Time value by a factor of three. If the point is a numeric point, nothing changes.</p>
Alarm State	Normal Low Limit High Limit or Fault	Displays the current state of the alarm s
Time Delay	hours: minutes: seconds	<p>Displays the minimum time period that an alarm condition must exist before the object alarms. In other words, the object status must meet the alarm criteria for a continuous period equal to or greater than defined in the this property before an alarm is generated. Time Delay provides a way to prevent nuisance alarms that may be caused by a momentary change in a state value (Normal, Low Limit, High Limit).</p> <p>NOTE: Time Delay does not affect alarms generated by a fault. There is no delay when transitioning in or out of a Fault generated alarm.</p>
Time Delay to Normal	hours: minutes: seconds	Sets the minimum time period that a normal condition must exist before the object may return to normal status.
Alarm Enable	<code>toOffnormal</code> or <code>toFault</code>	<p><code>toOffnormal</code> turns on the ability of the alarm to transition from normal to the alarm state <code>Offnormal</code>.</p> <p><code>toFault</code> turns on the ability of the alarm to transition from normal to the alarm state <code>Fault</code>.</p>
To Offnormal Times	text	<ul style="list-style-type: none"> • Alarm Time (defaults to null, which means that the event has not occurred) • Ack Time (defaults to null) Displays the time that the alarm was acknowledged. • Normal Time (defaults to null) Displays the time that the To Normal event occurred. • Count (defaults to zero (0)) Displays the total number of Offnormal events.
To Fault Times	text	<p>Alarm Time (defaults to null, which means that the event has not occurred) displays the time that the To Fault event occurred.</p> <p>Ack Time (defaults to null) displays the time that the alarm was acknowledged.</p> <p>Normal Time (defaults to null) displays the time that the To Normal event occurred.</p> <p>Count (defaults to zero (0)) displays the total number of Off-normal events.</p>
Time in Current State	hours: minutes: seconds	Displays the elapsed time since the component transition to the current state occurred.

Property	Value	Description
Source Name	%parent.display-Name% (default)	Displays the name of the alarm source. If you use the default script setting, the source name field shows the display name of the alarm extension parent. You can edit this script or type in a multi-line literal string to display.
To Normal Text	text	The text to display when the component transitions to a Normal status. When applicable, text entered for Fault Algorithm , High Limit Text and/or Low Limit Text may override this text.
Hyperlink Ord or Hyperlink	Ord, BQL Query or path	Associates an ord, BLQ query or path with an alarm state on the component. When an alarm is reported in the console, the Hyperlink button activates. Clicking this button links to the location you specify here.
Sound File	ord	The path to a sound file that plays when the current component is in an alarm state. Use the folder icon to browse to the file. Click the arrow icon to the right of the folder icon to test the path.
Alarm Icon	ord	Defines the path to a graphic file to add to the display in the timestamp column of the alarm table in the Console Recipient view.
Alarm Instructions	text	Opens a window in which you can provide customized instructions to the building attendant concerning how to handle the alarm.
Offnormal Algorithm	additional properties	Displays Offnormal options that depend on the alarm extension.
Ordinal	read-only	Provides a unique identifier for the particular OnCallList . The OnCallService tracks the next free ordinal number.
Alarm Class	List, console column, or field or %alarmClass% on a report.	Specifies the alarm routing option for the component.
Meta Data [alarms]	text	Allows you to enter new facets for the extension.

alarm-BooleanChangeOfStateAlarmExt

This extension implements a change of state alarm detection algorithm for Boolean objects as described in BACnet Clause 13.3.2. It is available in the **Extensions** folder of the **alarm** palette.

alarm-StringChangeOfValueAlarmExt

This extension generates an alarm upon inclusion or exclusion of a particular string value, or more accurately, regular expression (regex) of a point's Out slot (string type). This alarm extension is available in the **Extensions** folder of the **alarm** palette.

By default, matching is case sensitive, but this attribute may be configured using the **Case Sensitive** property in the extension's **Offnormal Algorithm** and **Fault Algorithm** container slots.

In addition to the standard alarm properties, this extension supports these properties.

Property	Value	Description
Expression	a value of: .*	This is the regexp value for any text.
Normal On Match	true, false	
Case Sensitive	true (default), false	Matching defaults to case sensitive.

Thus, by default status remains `ok` until an edit is made to one or both properties above.

Simple string example

A hospital emergency room desires an alarm created whenever the moon enters a "full moon" phase. A `StringWritable` is created and given a `StringChangeOfValueAlarmExt`. In this extension:

- In the `Offnormal Algorithm`'s **Expression** property, the following string is entered: `Full Moon`.
- The `Offnormal Algorithm`'s **Normal On Match** is set to `false`, and **Case Sensitive** is left at `true`.

In the station's `WeatherService`, a `WeatherProvider` has a `MoonPosition` component, which serves as the link source. A link is made from the `MoonPosition`'s `Phase` property to the `In16` slot of the `StringWritable`. On all phases of the moon but one, the `StringWritable` has a normal status. When `MoonPosition`'s phase changes to `Full Moon`, the `StringWritable` alarms, and remains in alarm until the next moon phase (`Waning Gibbous`).

NOTE: If the **Expression** string entry was simply: `Moon`, alarms would occur during both phases that include the string "Moon", namely "Full Moon" and "New Moon".

Regex examples

The **Expression** property in both the `Offnormal Algorithm` and `Fault Algorithm` containers can process a simple string value, as in the example. The **Expression** property also processes a value using regular expression (regexp) syntax. This provides even more flexibility, such as with use of "or" operators, among others.

Regex syntax is beyond the scope of this document, but a few regexp examples are listed below:

- Contains the word "alarm":
`(.*) (alarm) (.*)`
- Contains the word "offnormal" or "fault":
`(.*) (offnormal) | (fault) (.*)`
- Eight "1" or "0" characters, with the fourth and eighth characters being 1:
`(1|0){3}(1)(1|0){3}(1)`
- Empty text:
`^$`
- Any text:
`.*`

This is the default **Expression** property value, that is in an extension copied from the `alarm` palette.

alarm-FaultAlgorithm

This component is the super-class of all fault detection mechanisms and contains properties that specify fault conditions. The default implementation does not generate any `toFault` alarms. A `FaultAlgorithm` is under each type of alarm extension, along with an `OffnormalAlgorithm` container.

alarm-EnumChangeOfStateAlarmExt

This extension implements a change of state alarm detection algorithm for enum objects as described in BACnet Clause 13.3.2. Each algorithm instance defines a set of enumerated values that should be considered offnormal conditions and, therefore, should generate an alarm. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

alarm-EnumCommandFailureAlarmExt

This extension implements a command failure alarm detection algorithm for enum objects as described in BACnet. If the feedback and output values of the enum point are not equal for more than **timeDelay**, the system generates an offnormal alarm. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

alarm-OffnormalAlgorithm

This super-class of algorithm extension checks for off normal conditions. You access this extension under each type of alarm extension along with a **FaultAlgorithm** container.

This extension's properties specify which alarm conditions to check for.

alarm-OutOfRangeAlarmExt

This extension implements a standard out-of-range alarming algorithm, and applies to points with a status numeric output. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

Algorithm properties

These properties are unique to the **OutOfRangeAlarmExt**.

Property	Value	Description
Fault Algorithm, High Limit	true, false	Enable and disable high limits.
Low Limit	true, false	Enable and disable low limits.
Deadband		
High Limit Text		
Low Limit Text		
Offnormal Algorithm, High Limit	true, false	Enable and disable high limits.
Low Limit	true, false	Enable and disable low limits.
Deadband		
High Limit Text		
Low Limit Text		

alarm-StatusAlarmExt

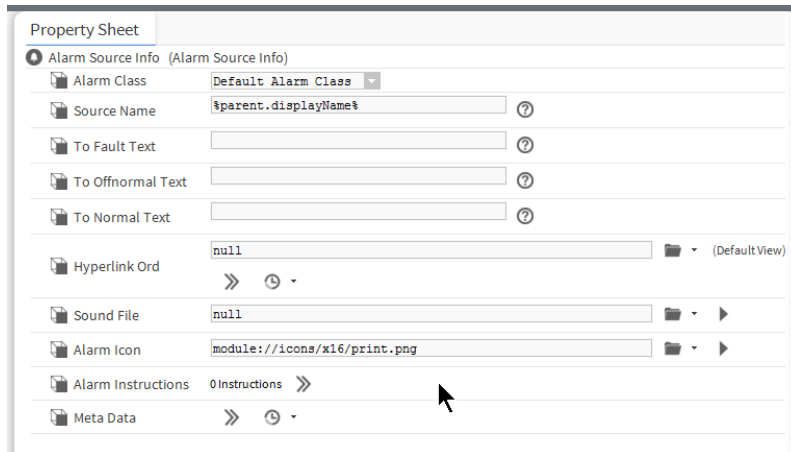
This extension provides alarming based upon any combination of status flags, and applies to all points and objects that accept extensions. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

alarm-AlarmSourceInfo

This container slot is available on any network component, and each child device component. The slot's properties populate the alarm record when the network or device does not respond to a monitoring ping. This ping is configured at the network level.

Each parent and child device object has its own **Alarm Source Info** slot with identical (but independently maintained) properties.

Figure 125 Alarm Source Info



Property	Value	Description
Alarm Class	List, console column, or field or % alarmClass% on a report.	Specifies the alarm routing option for the component.
sourceName	text	Displays the name in an alarm record that identifies the source of the alarm.
To Fault Text	text	The text to display when the component transitions to a Fault status. When applicable, text entered for Fault Algorithm , High Limit Text and/or Low Limit Text may override this text.
To Offnormal Text	text	The text to display when the component transitions to an Off-normal (alarm) state. When applicable, text entered for Fault Algorithm , High Limit Text and/or Low Limit Text may override this text.
To Normal Text	text	The text to display when the component transitions to a Normal status. When applicable, text entered for Fault Algorithm , High Limit Text and/or Low Limit Text may override this text.
Hyperlink Ord or Hyperlink	Ord, BQL Query or path	Associates an ord, BLQ query or path with an alarm state on the component. When an alarm is reported in the console, the Hyperlink button activates. Clicking this button links to the location you specify here.
Sound File	ord	The path to a sound file that plays when the current component is in an alarm state. Use the folder icon to browse to the file. Click the arrow icon to the right of the folder icon to test the path.

Property	Value	Description
Alarm icon	text	Defines the path to a graphic file the system includes in the Timestamp column of the alarm table in the Console Recipient view. Use the folder icon to browse for the file. Use the right-arrow to test the location you entered.
Alarm Instructions	text	Advice that accompanies the alarm notification (Alarm Record window) that provides important information for the operator. Click the right-pointing arrow to view the instructions.
Meta Data [alarms]	text	Allows you to enter new facets for the extension.

Types of alarm recipients

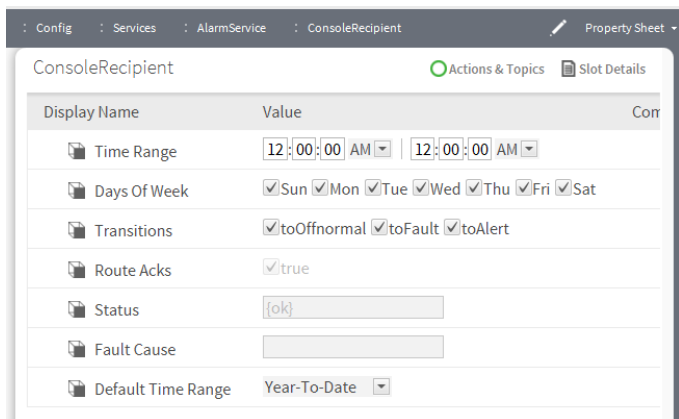
Alarm recipients are linked to an alarm class (from the alarm topic on the alarm class to the routeAlarm action on **AlarmRecipient**). Recipients may be configured to receive alarms at certain times of the day, certain days of the week, and to receive alarms of only specified transitions. There are several subclasses of the alarm recipient.

alarm-ConsoleRecipient

This component manages the transfer of alarms between the alarm history and the alarm console. The **ConsoleRecipient** is available in the **alarm** palette. For example, the console recipient gets unacknowledged alarms from the alarm history and updates the history when they are actually acknowledged. To view this property sheet, right-click the **ConsoleRecipient** component in the Nav tree and click **Views→Property Sheet**.

The default view of the console recipient is the alarm console view.

Console recipient properties are displayed and edited in the console recipient property sheet.



Property	Value	Description
Time Range	Start Time and End Time	Start Time sets the time of day to begin the function (for example, trigger schedule, alarm event)
Days of the Week or Days of Week		
Transitions	Option boxes	Allow selection of specific alarm transitions to display in the console. Only those transitions that are selected will be displayed in the console - even though the alarms are still saved into the alarm history.

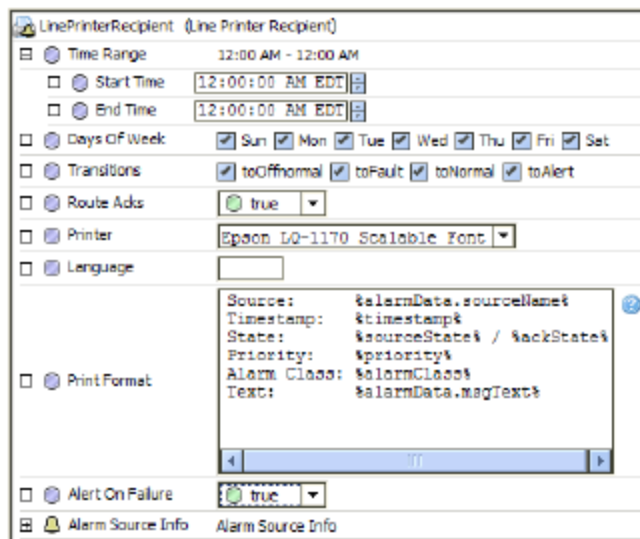
Property	Value	Description
Route Acks	true or false	Enables and disables the routing of alarm acknowledgements to the recipient.
Status [component]	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> {ok} indicates that the component is polling successfully. {down} indicates that polling is unsuccessful, perhaps because of an incorrect property. {disabled} indicates that the Enable property is set to false. fault indicates another problem.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Default Time Range	drop-down list	Selects which records to display based on the date.

alarm-LinePrinterRecipient

This component prints alarms to a lineprinter that is attached to a station running on a Windows platform, or to a remote (networked) printer known to its Windows OS. To access this property sheet right-click the LinePrinterRecipient component in the Nav tree and click **Views→Property Sheet**.

Alerts may be generated if the printing of an alarm fails, but the line printer recipient does not print alarms that it generates itself. The station must have permission to print on any printer chosen (which is typical).

A **PrinterRecipient** component is also available. It provides more formatting options, applicable to most modern printers.



Property	Value	Description
Time Range	Start Time and End Time	Start Time sets the time of day to begin the function (for example, trigger schedule, alarm event)
Days of the Week or Days of Week		

Property	Value	Description
Transitions	Option boxes	Allow selection of specific alarm transitions to display in the console. Only those transitions that are selected will be displayed in the console - even though the alarms are still saved into the alarm history.
Route Acks	true or false	Enables and disables the routing of alarm acknowledgements to the recipient.
Printer	drop-down list	Shows the printers that are available (both locally attached and remotely networked) through the host platform's Windows operating sys on the first line. NOTE: For more information about how to format this information, click on the help icon to the right of the field.
Language		Identifies the language to use for the
Print Format	text	<ul style="list-style-type: none"> Source: %alarmData.sourceName% These field definitions determine what prints for each alarm beginning with the Source, which prints the name of the entity that is responsible for generating the alarm on the first line. For more information about how to format this information, click on the help icon to the right of the field. Timestamp: %timestamp% Prints the time the alarm occurred on the second line. State: %sourceState% / %ackState% Prints the current alarm state on the third line. Priority: %priority% Prints the alarm priority on the fourth line. Alarm Class: %alarmClass% Prints the alarm class on the fifth line. Text: %alarmData.msgText% Prints any text associated with the alarm on the sixth line.
Alert on Failure	true or false	Enables and disables the generation of an alert if the printer fails to print an alarm.

alarm-StationRecipient

This component manages the transfer of alarms between the **AlarmService** and a remote station. For example, a station may send alarm notifications to a supervisor station – or any other remote station in the system. The **StationRecipient** is available in the **alarm** palette.

The station recipient component provides a place to specify the location and other details about that remote station. The properties on a station recipient include a field for selecting the remote station, as well as alarm collection options.

StationRecipient (StationRecipient h:228a)

Time Range 12:00 AM - 11:59 PM

Days Of Week Sunday Monday Tuesday Wednesday Thursday Friday Saturday

Transitions toOffnormal toFault toAlert

Status {ok}

Last Send Time Nov 03 2004 09:46:30.349 AM

Last Failure Time Nov 01 2004 11:30:19.355 AM

Last Failure Cause java.lang.NullPointerException

Retry Interval +00000h 00m 15s

Queued Alarm Count 3610

Remote Station demoSimJace

Property	Value	Description
Time Range	Start Time and End Time	Start Time sets the time of day to begin the function (for example, trigger schedule, alarm event)
Days of the Week or Days of Week		
Transitions	Option boxes	Allow selection of specific alarm transitions to display in the console. Only those transitions that are selected will be displayed in the console - even though the alarms are still saved into the alarm history.
Status [component]	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> {ok} indicates that the component is licensed and polling successfully. {down} indicates that polling is unsuccessful, perhaps because of an incorrect property. {disabled} indicates that the <code>Enable</code> property is set to false. {fault} indicates another problem.
Last Send Time	time	The date and time the system sent the last alarm to the station.
Last Failure Time	time	The date and time of any last failure.
Last Failure Cause	text	The reason for the failure.
Retry Interval	hours and minutes	In the case of a failed alarm transmission, the amount of time the system waits before attempting to send the alarm to the station again.
Queued Alarm Count	number	The number of alarms that are ready to be sent.
Remote Station		Displays a list of eligible remote stations. Valid stations have a valid network connection between the Supervisor and the station. The properties configured in the alarm class in the Alarms component of the remote station's NiagaraNetwork determine which station(s) receive the alarms.

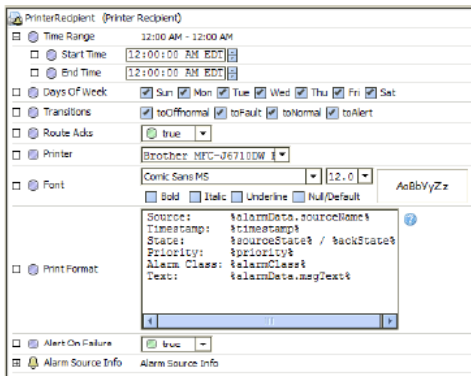
alarm-PrinterRecipient

This component can be used to print alarms to most types of printers (including laser printers) attached to a station running on a Windows platform, or to remote (networked) printers known to its Windows OS. It is available in the **Recipients** folder of the alarm module palette.

PrinterRecipient differs from the older **LinePrinterRecipient** component, originally intended to work only with line printers, where alarms print without a new page feed on each alarm, and the native font of the target printer is always used.

Like the **LinePrinterRecipient**, **PrinterRecipient** applies to Windows hosted stations only. The printer must be known to the host platform's Windows OS and selected from the printer drop down-list. Alerts may be generated if the printing of an alarm fails, but the printer recipient does not print alarms that it generates itself.

The main differences between the **LinePrinterRecipient** and **PrinterRecipient** are additional font property settings, which allow the selection of font type, size, and various style overrides. Combined with multi-line alarm message text properties that are available in various alarm extensions, the **PrinterRecipient** provides flexibility for alarm printing.



Property	Value	Description
Time Range	Start Time and End Time	Start Time sets the time of day to begin the function (for example, trigger schedule, alarm event)
Days of the Week or Days of Week		
Transitions	Option boxes	Allow selection of specific alarm transitions to display in the console. Only those transitions that are selected will be displayed in the console - even though the alarms are still saved into the alarm history.
Route Acks	true or false	Enables and disables the routing of alarm acknowledgements to the recipient.
Printer	drop-down list	Shows the printers that are available (both locally attached and remotely networked) through the host platform's Windows operating sys on the first line. NOTE: For more information about how to format this information, click on the help icon to the right of the field. tem.

Property	Value	Description
Print Format	text	<ul style="list-style-type: none"> • Source: %alarmData.sourceName% These field definitions determine what prints for each alarm beginning with the Source, which prints the name of the entity that is responsible for generating the alarm on the first line. For more information about how to format this information, click on the help icon to the right of the field. • Timestamp: %timestamp% Prints the time the alarm occurred on the second line. • State: %sourceState% / %ackState% Prints the current alarm state on the third line. • Priority: %priority% Prints the alarm priority on the fourth line. • Alarm Class: %alarmClass% Prints the alarm class on the fifth line. • Text: %alarmData.msgText% Prints any text associated with the alarm on the sixth line.
Alert on Failure	true or false	Enables and disables the generation of an alert if the printer fails to print an alarm.

alarm-FaultAlgorithm

This component is the super-class of all fault detection mechanisms and contains properties that specify fault conditions. The default implementation does not generate any toFault alarms. A **FaultAlgorithm** is under each type of alarm extension, along with an **OffnormalAlgorithm** container.

alarm-EnumChangeOfStateAlarmExt

This extension implements a change of state alarm detection algorithm for enum objects as described in BACnet Clause 13.3.2. Each algorithm instance defines a set of enumerated values that should be considered offnormal conditions and, therefore, should generate an alarm. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

alarm-EnumCommandFailureAlarmExt

This extension implements a command failure alarm detection algorithm for enum objects as described in BACnet. If the feedback and output values of the enum point are not equal for more than **timeDelay**, the system generates an offnormal alarm. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

alarm-MemoryAlarmService

This component provides an alternative to the standard file-based **AlarmService**. When you use this service, alarms are not stored persistently on the station's host as they are with the standard file-based **AlarmService**. The **MemoryAlarmService** is available in the **alarm** palette.

This service coordinates the routing of alarms within the framework.

NOTE: A station should have only one alarm service. Do not enable both the standard **AlarmService** and **MemoryAlarmService** on the same station.

Choosing **MemoryAlarmService** might be appropriate for situations where you do not want to keep a large store of alarms on your host and are looking primarily for immediate alarm notification. Alarm records are stored in memory and are lost in the case of a power failure.

Like the **AlarmService**, the **MemoryAlarmService** may contain one or more alarm classes. An alarm class may route alarms to one or more alarm recipient types.

The routing process and views are the same as those for the standard file-based alarm service, including alarm acknowledgements from the recipients back to the source, as well as alarm notifications from the source to the recipients. The default view (wire sheet view) of the **AlarmService** makes it easy to visualize the relationships between the alarm class and the alarm recipient. These relationships are created by linking the alarm class to the alarm recipient.

alarm-OffnormalAlgorithm

This super-class of algorithm extension checks for off normal conditions. You access this extension under each type of alarm extension along with a **FaultAlgorithm** container.

This extension's properties specify which alarm conditions to check for.

alarm-OutOfRangeAlarmExt

This extension implements a standard out-of-range alarming algorithm, and applies to points with a status numeric output. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

Algorithm properties

These properties are unique to the **OutOfRangeAlarmExt**.

Property	Value	Description
Fault Algorithm, High Limit	true, false	Enable and disable high limits.
Low Limit	true, false	Enable and disable low limits.
Deadband		
High Limit Text		
Low Limit Text		
Offnormal Algorithm, High Limit	true, false	Enable and disable high limits.
Low Limit	true, false	Enable and disable low limits.
Deadband		
High Limit Text		
Low Limit Text		

alarm-StatusAlarmExt

This extension provides alarming based upon any combination of status flags, and applies to all points and objects that accept extensions. This alarm extension is available in the **Extensions** folder of the **alarm** palette.

alarm-StatusAlarmExt

StatusAlarmExt provides alarming based upon any combination of status flags, and applies to all points/objects that accept extensions. This alarm extension is available in the Extensions folder of the alarm palette. See “Types of alarm extensions” and “About alarm extensions and components”.

alarm-StringChangeOfValueAlarmExt

This extension generates an alarm upon inclusion or exclusion of a particular string value, or more accurately, regular expression (regexp) of a point's Out slot (string type). This alarm extension is available in the **Extensions** folder of the **alarm** palette.

By default, matching is case sensitive, but this attribute may be configured using the **Case Sensitive** property in the extension's **Offnormal Algorithm** and **Fault Algorithm** container slots.

In addition to the standard alarm properties, this extension supports these properties.

Property	Value	Description
Expression	a value of: .*	This is the regexp value for any text.
Normal On Match	true, false	
Case Sensitive	true (default), false	Matching defaults to case sensitive.

Thus, by default status remains `ok` until an edit is made to one or both properties above.

Simple string example

A hospital emergency room desires an alarm created whenever the moon enters a "full moon" phase. A **StringWritable** is created and given a **StringChangeOfValueAlarmExt**. In this extension:

- In the **Offnormal Algorithm**'s **Expression** property, the following string is entered: `Full Moon`.
- The **Offnormal Algorithm**'s **Normal On Match** is set to `false`, and **Case Sensitive** is left at `true`.

In the station's **WeatherService**, a **WeatherProvider** has a **MoonPosition** component, which serves as the link source. A link is made from the **MoonPosition**'s **Phase** property to the **In16** slot of the **StringWritable**. On all phases of the moon but one, the **StringWritable** has a normal status. When **MoonPosition**'s phase changes to `Full Moon`, the **StringWritable** alarms, and remains in alarm until the next moon phase (`Waning Gibbous`).

NOTE: If the **Expression** string entry was simply: `Moon`, alarms would occur during both phases that include the string "Moon", namely "Full Moon" and "New Moon".

Regex examples

The **Expression** property in both the **Offnormal Algorithm** and **Fault Algorithm** containers can process a simple string value, as in the example. The **Expression** property also processes a value using regular expression (regexp) syntax. This provides even more flexibility, such as with use of "or" operators, among others.

Regex syntax is beyond the scope of this document, but a few regexp examples are listed below:

- Contains the word "alarm":
`(.*) (alarm) (.*)`
- Contains the word "offnormal" or "fault":
`(.*) (offnormal) | (fault) (.*)`
- Eight "1" or "0" characters, with the fourth and eighth characters being 1:
`(1|0){3}(1)(1|0){3}(1)`

- Empty text:

`^$`

- Any text:

`.*`

This is the default **Expression** property value, that is in an extension copied from the **alarm** palette.

Example

(Simple string) A hospital emergency room desires an alarm created whenever the moon enters a “full moon” phase. A `StringWritable` is created and given a `StringChangeOfValueAlarmExt`. In this extension:

In the `Offnormal Algorithm`’s “Expression” property, the following string is entered: `Full Moon`

The `Offnormal Algorithm`’s “Normal On Match” is set to false, and `Case Sensitive` is left at true.

In the station’s `WeatherService`, a `WeatherProvider` has a `MoonPosition` component, which serves as the link source. A link is made from the `MoonPosition`’s “Phase” property to the “In16 slot” of the `StringWritable`. On all phases of the moon but one, the `StringWritable` has a normal status. When `MoonPosition`’s phase changes to “Full Moon”, the `StringWritable` has an alarm status, and remains so until the next moon phase (“Waning Gibbous”).

Note that in this example, if the Expression string entry was simply: `MOON`, that alarms would occur during both phases that include the string “Moon”, namely “Full Moon” and “New Moon”.

Regexp notes

The “Expression” property in both the `Offnormal Algorithm` and `Fault Algorithm` containers of a `StringChangeOfValueExt` can process a simple string value, as in the previous Example. The Expression property also processes a value using “regular expression” (regexp) syntax. This provides even more flexibility, such as with use of “or” operators, among others.

Regexp syntax is beyond the scope of this document, but a few regexp examples are listed below:

- Contains the word “alarm”:

`(.*) (alarm) (.*)`

- Contains the word “offnormal” or “fault”:

`(.*) (offnormal) | (fault) (.*)`

- Eight “1” or “0” characters, with the fourth and eighth characters being 1:

`(1|0){3}(1)(1|0){3}(1)`

- Empty text:

`^$`

- Any text:

`.*`

This is the default Expression property value, that is in an extension copied from the alarm palette.

Components in alarmRdb module

- `rdbAlarmService`

alarmRdb-RdbAlarmService

RdbAlarmService provides a means for using an RDBMS to store alarm records instead of using the default alarm database (^alarm\alarm.adb). This is different than exporting histories to an RDBMS. When you use the rdbAlarmService, it replaces the use of the station's default alarm database.

- Refer to the *Rdbms Driver Guide* for information about setting up the RdbAlarmService component.

Components in backup module

- BackupService
- FoxBackupJob

backup-BackupService

The BackupService provides for complete configuration backups to a 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 Niagara 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."

See the following for more details:

- About a backup dist
- Restoring a backup
- BackupService configuration

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, 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.

CAUTION: Be careful to keep backup dist files in a secure location. They have always contained sensitive information, for example a station's config.bog file. They also may contain sensitive host platform information. In update releases (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, 2013 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 related details, refer to "Restoring a backup dist" in the *Platform Guide*.

Also see "Restoring a backup dist" in the *Niagara 4 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:

- **Enabled**
Either true (default) or false. Currently, this setting makes no difference (manual backup still possible even if service has disabled status).
- **Exclude Files**
Specifies file types to exclude from the backup dist, either by name or extension (each delimited by ";""). By default, the following files are excluded: `*.hdb;*.adb;*.lock;*backup*;console.*;config.bog.b*;config_backup*`
- **Exclude Directories**
Specifies station subdirectories to exclude from the backup dist, using relative ORD syntax. An ORD chooser control provides a Directory Chooser dialog in which you can select station subfolders. By default, the following subfolders are excluded: `file:^history, file:^alarm`
- **Offline Exclude Files**
Specifies file types to exclude from the backup dist, when the station is stopped on the source host. either by name or extension (each delimited by ";""). By default, the following files are excluded: `*.lock;*backup*;console.*;config.bog.b*;config_backup*`
Note that history (*.hdb) and alarm (*.adb) files are backed up, unlike with a running backup.
- **Offline Exclude Directories**
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:

- Job State
Which of the following states the backup job is in currently: unknown, running, canceling, canceled, success, failed.
- Progress
A percentage (0) of progress toward completing the job.
- Start Time
Displays the time that the job started.
- Heart Beat Time
Displays the time of the last indication that the job is alive.
- End Time
Displays the time that the job ends.

For related information, see:

- "backup-BackupService"
 - "baja-Job".
 - "baja-JobService"
- NOTE:** All jobs in a station are cleared upon a station restart.

Components in baja module

- 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-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. For a general review, see "Security model overview".

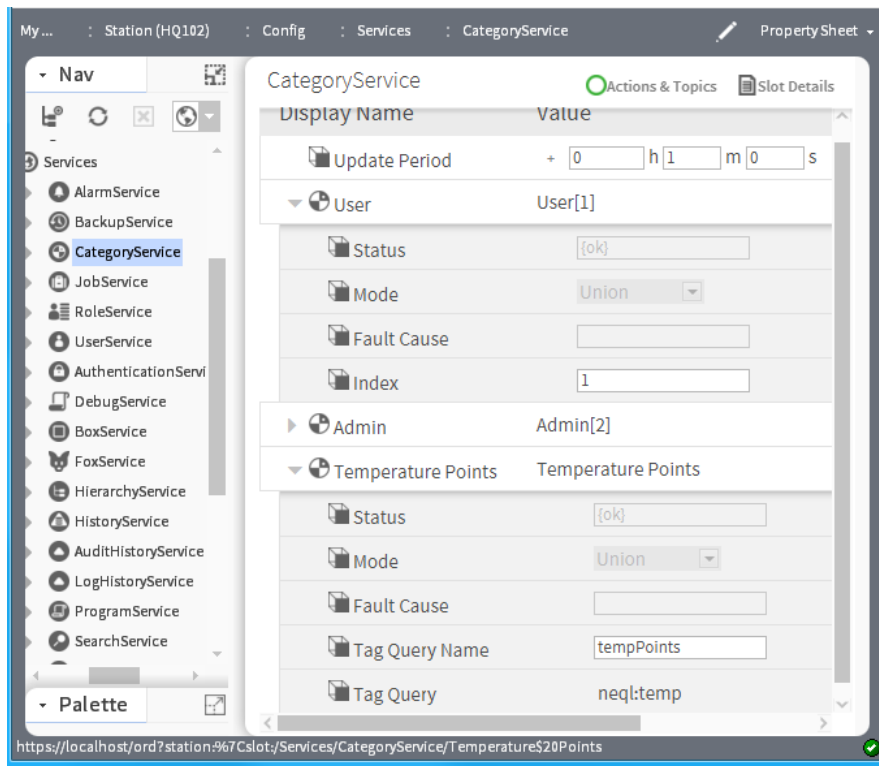
baja-CategoryService

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 126 CategoryService property sheet



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

Property	Value	Description
Update Period	hours minutes seconds	Sets the interval at which the system automatically assigns ancestor permissions. The default value is one (1) minute. If you assign a zero value, the system disables this feature.

User, Admin and additional basic category properties

Property	Value	Description
Status [component]	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> {ok} indicates that the component is polling successfully. {down} indicates that polling is unsuccessful, perhaps because of an incorrect property. {disabled} indicates that the Enable property is set to false. fault indicates another problem.
Mode		

Property	Value	Description
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.

Tagged category properties

Property	Value	Description
Tag Query Name	text	A descriptive name to represent the results of the search.
Tag Query	NEQL	A NEQL query. This property is required when Type is Tagged Category.

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 created 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 call "foo", the order of reflection mapping is:

- special call (see below)
- 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)%"

For related details, see the engineering notes document *BFormat (Baja Format) Property Usage*.

baja-IpHost

IpHost is used to represent a host machine that is identified with an IP address. The hostname of an IpHost is either a a name resolvable via DNS or is a raw IP address.

- A blue square indicates active connection(s) from Workbench, e.g. 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 "PlatformServiceContainer configuration parameters" in the *Niagara 4 Platform Guide*).
- **FoxBackupJob** — From a station backup (dist) to a remote PC, see BackupService.

Many drivers also have various job types too. For example, the NiagaraDriver includes a **StationDiscoveryJob** and **NiagaraScheduleLearnJob**.

Every job finishes displaying one of the following status descriptors:

- **Success** — Job completed successfully.
- **Canceled** — Job canceled by user.
- **Failure** — Job failed to complete.
- **Completed** — Job completed.

Also, if you have the station open in Workbench, you see a momentary "notification popup" 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 JobServiceManager view of a station's JobService. You can also open a Jobs side bar to see all jobs in all opened stations (see "Using the jobs side bar").

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** dialog 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 JobServiceManager.

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").

For related details, see the section "About modules" including subsections "About module characteristics" as well as "Working with modules", including subsections "Creating a new (.jar) module".

baja-ModuleSpace

ModuleSpace is the container for modules which are keyed by their module name. In Workbench this is "My Modules" under "My Host". Starting in AX-3.7, in addition to standard modules (.jar files), "synthetic modules" (.sjar files) can be created in the ModuleSpace. See "About modules".

baja-Permissions

Permissions are a property used to define the permissions given to a user's PermissionsMap.

baja-PermissionsMap

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 more details, see "Permissions and security" and "About permission levels". If a user has been configured as a super user, then all permissions are automatically granted to all categories.

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-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. See "Station" in the *Niagara 4 Developers Guide* for developer information. The Station is available in a fox connection to a Niagara host, along with its file space (**Files**) and histories (**History**).

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 the save job:

- Job State
indicates which of the following states the save job is in currently: unknown, running, canceling, canceled, success, failed.
- Progress
indicates a percentage (0) of progress toward completing the job.
- Start Time
displays the time that the job started.
- Heart Beat Time
displays the time of the last indication that the job is alive.
- End Time
displays the time that the job ends

For related information, refer to:

- "baja-Job".
- "baja-JobService"

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

baja-SyntheticModuleFile

(AX-3.7 and later) 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.

For more details, see the following sections:

- "Users and security"
- "User"(all properties)
- "UserService security notes"

Also, see "Security model overview"

baja-UserPasswordConfiguration

UserPasswordConfiguration (Password Configuration) is a child container under each User component (and UserPrototype component) in an AX-3.7 or later station. 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 "About password expiration and reset" and "Password reset". The station's UserService also has a related Password Configuration child container.

NOTE: As a "mix-in", these components are not available in a new station until it is started and saved.

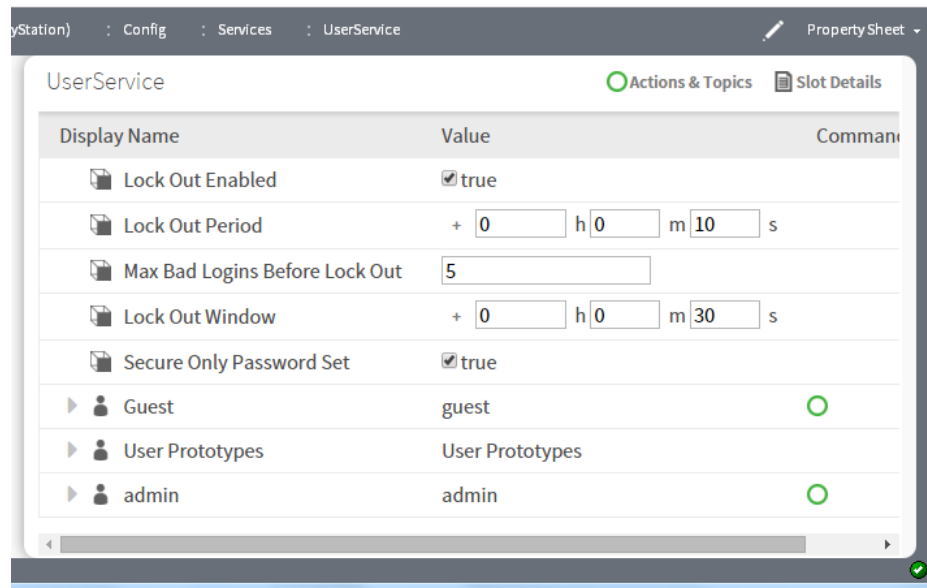
baja-UserPrototypes

UserPrototypes is a frozen container on a station's UserService. It contains a single frozen "Default Prototype" user, as well as any additional users needed for support of "centralized users" in the station's Niagara Network. For more details, see "Network users" and "About User prototypes".

baja-UserService

This service manages all system users: human and machine. It is located in the station's **Services** container. The

The **User Manager** is its primary view of this service. The UserService is available in the baja module. By default, the UserService is included when you create a new station using the **New Station** wizard.

Figure 127 *UserService* property sheet

Property	Value	Description
Lock Out Enabled	true or false	If enabled (<code>true</code>), a number of consecutive authentication failures temporarily disables login access to the user account, for the duration of the lock out period (next property). Using lock out 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 10 seconds)	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 is thwarts such an attack, as the attacker must wait 10 seconds after each five unsuccessful login attempts. If deemed necessary, you can adjust 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 Lock Out Window , 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 30 seconds)	If lock out is enabled, and a user fails to log in successfully before the Max Bad Logins Before Lock Out window (period) expires, the user is locked out for the Lock Out Period duration. The system enforces changes to lock out properties the next time the user logs in. For example, if Max Bad Logins Before Lock Out is set to 5, user ScottF fails to log in four times within the Lock Out Window , and an admin-level user changes Max Bad Logins Before Lock Out to 3, the change does not lock ScottF out. User ScottF still has one more chance to log in before getting locked out.

Property	Value	Description
		If ScottF's fifth attempt to log in fails, the system locks him out the next time he attempts to log in because five failed attempts is greater than or equal to the Max Bad Logins Before Lock Out of 3.
Secure Only Password Set		
User Prototypes	multiple properties	See the next section.

Default Prototype

These are the properties to configure for each user.

Property	Value	Description
Full Name		
Enabled [general]	true or false	Activates and deactivates use of the function.
Expiration	radio button	Allows you to configure a temporary user.
Lock Out	true or false (default)	Allows you to prohibit access for a specific user.
Language		Identifies the language to use for the
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/NavFile.nav	Identifies the file to use for displaying a customized navigation tree.
Prototype Name	text	A name to identify this prototype as belonging to a group. May be left blank.
Network User	true or false (default)	Identifies the user as someone permitted to use the network.
Cell Phone Number	number	The users 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.
Default Web Profile		
Mobile Web Profile		

Authenticator—User Password Configuration

Property	Value	Description
Force Reset at Next Login	true or false (default)	Causes the system to request that the user create a new password the next time they log in.
Expiration	radio button	Allows you to configure a password change for a specific date and time.

baja-UserServicePasswordConfiguration

UserServicePasswordConfiguration (Password Configuration) is a child container under the UserService in an AX-3.7 or later station. 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 “About password expiration and reset” and “Password history (unique passwords)”. Also note each station User has a related Password Configuration child container as well.

NOTE: As a “mix-in”, 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 a general explanation about Baja virtual components, see the *NiagaraAX Drivers Guide* section “Virtual gateway and components”.

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.” Note other object spaces are Files and History. Initial applications of virtual gateways are expected in Niagara drivers. For a general explanation, see the *NiagaraAX Drivers Guide* section “Virtual gateway and components”.

baja-WsTextBlock

WsTextBlock (**Text Block**) 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.

Text Block is available in the baja palette.

baja-ZipFile

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

Components in chart module

- 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

- BooleanPoint
- BooleanWritable
- DiscreteTotalizerExt
- EnumPoint
- EnumWritable
- NullProxyExt
- NumericPoint
- NumericTotalizerExt
- NumericWritable
- StringPoint
- StringWritable
- TimeTrigger

control-BooleanPoint

BooleanPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt. See “About control points” for details. The BooleanPoint 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. See “About control points” and “About writable points” for more details.

The BooleanWritable 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-EnumPoint

EnumPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt. See “About control points” for details. 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. See “About control points” and “About writable points” for more details.

The EnumWritable is available in the Points 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-NumericPoint

NumericPoint is a basic read-only control point, with default slots: Facets, Out, and ProxyExt. See “About control points” for details. The NumericPoint is available in the Points 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-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. See “About control points” and “About writable points” for more details.

The NumericWritable 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. See “About control points” for details. 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. See “About control points” and “About writable points” for more details.

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

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 converters module

- EnumToSimpleMap

converters-EnumToSimpleMap

EnumToSimpleMap maps ordinals to Simple instances of the same type.

Components in crypto module

- CryptoService
- SslProvider

crypto-CryptoService

CryptoService is a station service that can provide SSL features for earlier model QNX-based JACEs (JACE-2/4/5 series, running the IBM J9 JVM) at any Niagara release level. It also applies to any Niagara platform running a pre-AX-3.7 release that require SSL capability.

NOTE: Any AX-3.7 or later platform that runs the Oracle Hotspot JVM (most newer QNX-based JACEs) and all Windows-based hosts) should be configured differently for SSL, and should not use this station service or module. .

crypto-SslProvider

Provides SSL configuration for the CryptoService in a NiagaraAX host that cannot run the newer, platform-based SSL introduced in AX-3.7 (such as JACE-2/4/5 series, running the IBM J9 JVM).

Components in email module

- Email
- EmailAlarmAcknowledger
- EmailRecipient
- EmailService
- IncomingAccount
- OutgoingAccount

email-Email

Email represents an email message.

email-EmailAlarmAcknowledger

The EmailAlarmAcknowledger component (AX-3.5 and later) provides a way for users to acknowledge alarms by sending an email reply to an email alarm notification. This component is available in the email palette and works with the On Call Service or directly with the Email Service. Also see "About the Email Alarm Acknowledger".

Actions

Actions include: Received, which opens the Received email dialog box.

email-EmailRecipient

The top-level email Messaging service. This service manages the routing of all email messages from message sources to message recipients. The EmailRecipient is available in the email palette.

Actions

Actions include RouteAlarm.

RouteAlarm

RouteAlarm routes the alarm.

email-EmailService

EmailService manages the life-cycle of Email. The EmailAccountManager is its primary view. The EmailService is available in the email palette. EmailService may contain an IncomingAccount component and an OutgoingAccount component.

email-IncomingAccount

IncomingAccount is a Email that receives mail. A number of properties specify the SMTP host, port, and user credentials to use, as well as other parameters. The IncomingAccount is available in the email palette.

IncomingAccount properties include the following:

- Hostname

This is the name of the mail server. For example `mail.acme.com` could be a Hostname.

- Port

This is the port number associated with the email account. Typically email incoming account port numbers are port "110", however, if you leave the setting at its default value of "-1" the IncomingAccount will search for and use a valid port.

CAUTION: With the default configuration (see "Delivery Policy" property, below) the incoming email account deletes all emails from the mail server when it checks the account to retrieve new email, even if the emails are already marked as read by another email client. If permanent retention of the emails is required then do one of the following: change the Delivery Policy setting from Delete or configure a second service account which the mail server forwards emails to and configure the station's incoming account to check the second service account.

- Account

This is the name of the distinct account that is authorized for access to the "Hostname" mail server. For example, if you are using an email account named "controls@acme.com" on the host described above, the account name is simply "controls". The Hostname in this case could be "mail.acme.com".

- **Password**
This is the login credential for the Account specified in the previous field.
- **Pollrate**
This field allows you to specify how often the account connects to the mail server and checks for unread mail messages. Increasing the pollrate value increases the time between polls.
- **Enabled**
This field allows you to activate or deactivate the account by choosing true or false, respectively.
- **Status**
This is a read-only display of the condition of service. The status displays {ok} if the account is polling successfully. Other indications include:
 - {down} which means that the polling is unsuccessful, perhaps because of an incorrect Hostname, Password, or Account name.
 - {disabled} because the Enable property is set to false
- **Last Poll Success, Last Poll Failure**
These two properties display the time (in hours and minutes) of the last polling success and failure.
- **Last Poll Failure Cause**
This display-only field provides an error message to indicate a reason for polling failure.
- **Debug**
This Boolean property turns Debug mode on true or off, as desired. When Debug is turned on, you can see detailed debug information in a station's standard output view (WorkbenchAX Platform > Application Director view) when the Station tries to send or receive email. This can be used to troubleshoot the accounts and faults.
- **Use Ssl**
The Ssl (Secure Socket Layer) option list allows you to disable Ssl (false) or enable it (true). Enable Ssl for communication with a host email server that requires it.
- **Store**
This property allows you to choose between two standards of mail retrieval. You need to choose the option that is in use by your host mail server: Imap or Pop3
- **Delivery Policy**
This property provides an option list that allows you to select how the incoming email account handles incoming emails at the mail server.
The following options are available:
 - **Delete**
With this configuration the incoming email account deletes all emails from the mail server when it checks the account to retrieve new email, even if the emails are already marked as read by another email client.
 - **Mark As Read**
This option marks all emails as read on the mail server when it checks the account to retrieve new email.
 - **Mark As Unread**
This option marks all emails as unread on the mail server when it checks the account to retrieve new email.

email-OutgoingAccount

OutgoingAccount is a Email component that sends mail. A number of properties specify the SMTP host, port, and user credentials to use, as well as other parameters. A "Max Sendable Per Day" property can limit the number of station-generated emails, where the default value is 100. The OutgoingAccount is available in the email palette.

OutgoingAccount properties include the following:

- **Hostname**
This is the name of the mail server. For example mail.acme.com could be a Hostname.
- **Port**
This is the port number associated with the email account. Typically email outgoing account port numbers are port "25", however, if you leave the setting at its default value of "-1" the OutgoingAccount will search for and use a valid port.
- **Account**
This is the name of the distinct account that is authorized for access to the "Hostname" mail server. For example, if you are using an email account named "controls@acme.com" on the host described above, the account name is simply "controls". The Hostname in this case could be "mail.acme.com".
- **Password**
This is the login credential for the Account specified in the previous field. Increasing the pollrate value increases the time between polls. During the time between polls, emails may be queued (up to the max queue size) until the next poll time. At the next poll time all queued emails are sent.
- **Pollrate**
This field allows you to specify how often the account executes a send action.
- **Enabled**
This field allows you to activate or deactivate the account by choosing true or false, respectively.
- **Status**
This is a read-only display of the condition of service. The status displays {ok} if the account is polling successfully. Other indications include:
 - {down} which means that the polling is unsuccessful, perhaps because of an incorrect Hostname, Password, or Account name.
 - {disabled} because the Enable property is set to false
- **Last Poll Success, Last Poll Failure**
These two properties display the time (in hours and minutes) of the last polling success and failure.
- **Last Poll Failure Cause**
This display-only field provides an error message to indicate a reason for polling failure.
- **Debug**
This Boolean property turns Debug mode on true or off, as desired. When Debug is turned on, you can see detailed debug information in a station's standard output view (WorkbenchAX Platform > Application Director view) when the Station tries to send or receive email. This can be used to troubleshoot the accounts and faults.
- **Use Ssl**
The Ssl (Secure Socket Layer) option is available on the OutgoingAccount. The option list allows you to disable Ssl (false) or enable it (true). Enable Ssl for communication with a host email server that requires it.

- **Transport**
This field allows you to select from available options for email communication. The default setting and most common is SMTP.
- **Connection Timeout**
This configurable field setting controls how long the station waits for a response from the mail server before generating an exception and setting the fault cause. It waits for the next scheduled poll and attempts to contact the mail server again at that frequency.
- **Use Authentication**
This property allows you to specify that login credentials are required for sending any email. Sometimes authentication is not required for emails routed to recipients in the same domain. Setting this property to true makes the login credentials mandatory for any email.
- **Reply To**
This field specifies the contents of the "From:" field in the email that is sent.
- **Persistent, Persistent Directory**
Setting the Persistent property to true saves each queued email as an xml file in the designated persistence directory. Once the emails are actually sent, the xml files are deleted from the directory. The purpose of this is to keep a copy of the emails in the queue which would be lost if the station was stopped prior to the emails being sent. When the station restarts, emails are loaded from the "Persistent Directory" back to the queue.
- **Email Queue Tracking Properties**
"Queue" is where emails reside while they are waiting to be sent. Assuming that the Account Status is "ok", typically, the length of time that an email is in the queue depends on the "Pollrate" setting. The following properties relate to the queue and mail management properties
 - **Allow Disabled Queuing**
This property (when set to true) allows emails to reside in the queue even when the Enabled status is set to false.
 - **Queue Size**
This property refers to how many emails are currently in the queue (waiting to be sent). You can clear the queue at any time by right-clicking on the appropriate outgoing account property and selecting Actions > Clear Queue from the popup menu.
 - **Max Queue Size**
This property allows you to specify how many emails are allowed to occupy the queue.
 - **Number Sent**
This property displays the number of emails that have been sent. You can reset this number to zero at any time by right-clicking on the appropriate outgoing account property and selecting Actions > Reset Number Sent from the popup menu.
 - **Max Sendable Per Day**
This property allows you to specify how many emails may be sent in one day.
 - **Number Discarded**
This read-only property displays a value to indicate how many emails did not successfully send.
 - **Last Discard**
This read-only property displays a date and time value to indicate when the last email did failed to send.
 - **Last Discard Cause**

This read-only property displays an error message that indicates the cause of the last email send failure.

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

Palette file is just a Bog file with a different extension and icon. Many modules include a palette. For details on palettes in Workbench, see "About palettes".

file-PdfFile

PdfFile stores a Adobe PDF file.

file-PngFile

PngFile stores a PNG image.

file-PowerPointFile

PowerPointFile stores a Microsoft PowerPoint file.

file-PrintFile

PrintFile stores XML print markup.

file-PxFile

PxFile is just a Bog File file with a different extension and icon.

file-TextFile

TextFile stores plain text.

file-VideoFile

VideoFile stores an video file.

file-VisioFile

VisioFile stores a Microsoft Visio file.

file-WordFile

WordFile stores a Microsoft Word file.

file-XmlFile

XmlFile stores an xml file.

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 history module

- AuditRecord
- AuditHistoryService

- BooleanCovHistoryExt
- BooleanIntervalHistoryExt
- ConfigRule
- ConfigRules
- CovAlgorithm
- EnumCovHistoryExt
- EnumIntervalHistoryExt
- FoxHistory
- FoxHistorySpace
- HistoryConfig
- HistoryDevice
- HistoryEditorOptions
- HistoryGroup
- HistoryGroupings
- HistoryId
- HistoryPointList
- HistoryPointListItem
- HistoryService
- HistoryShortcut
- IntervalAlgorithm
- LocalDatabaseConfig
- LogHistoryService
- NumericCovAlgorithm
- NumericCovHistoryExt
- NumericIntervalHistoryExt
- StringCovHistoryExt
- StringIntervalHistoryExt

history-AuditRecord

The AuditRecord keeps a History of changes made by users. If enabled, it registers itself as the Auditor for the system when the service is started. The AuditRecord is available in the History Module under the HistoryService.

history-AuditHistoryService

The AuditHistoryService keeps a history of changes made by users. If enabled, it registers itself as the auditor for the station when the service is started. The AuditHistoryService is available in the history palette.

One of the important aspects of security is the ability to analyze what has happened after the fact. The Niagara component model is designed to audit all property modifications and all action invocations. Auditable actions include:

- Property changed
- Property added

- Property removed
- Property renamed
- Properties reordered
- Action invoked

Component modifications are only audited when the modification method is passed a non-null context with a non-null user. The history module includes a standard implementation of an audit trail stored to an AuditHistory history.

For more details, see “About user audits”.

history-BooleanCovHistoryExt

BooleanCovHistoryExt is a history extension that is used for recording on change of value for boolean point data. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” For more information about this extension.

history-BooleanIntervalHistoryExt

BooleanIntervalHistoryExt is a history extension that is used for recording on intervals for boolean point data. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” for more information about this extension.

history-ConfigRule

ConfigRule is used to determine the overrides for an existing history configuration. The ConfigRule is available in the history Module.

history-ConfigRules

ConfigRules is the container for rules that determine the configuration of histories that are pushed to the local device. Configuration rules are applied when a history is created. Changing a rule has no effect on existing histories. The ConfigRules is available in the history Module.

history-CovAlgorithm

CovAlgorithm determines when to log a point's value according to change of value. The CovAlgorithm is available in the history Module Extensions Directory under some Change Of Value extensions as Collector.

history-EnumCovHistoryExt

EnumCovHistoryExt is a history extension used to record on a change of value for Enum point data. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” for more information about this extension.

history-EnumIntervalHistoryExt

EnumIntervalHistoryExt is a history extension used to record on intervals for Enum point data. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” for more information about this extension.

history-FoxHistory

FoxHistory is the implementation of BIHistory that works with the FoxHistorySpace.

history-FoxHistorySpace

FoxHistorySpace provides access to a History Database using the fox protocol.

history-HistoryConfig

HistoryConfig is the configuration for a History in the History Database.

history-HistoryDevice

HistoryDevice represents a source device for histories.

history-HistoryEditorOptions

The HistoryEditorOptions stores the options used to configure history options.

These are stored under `/user/{user}/textEditor.options`.

history-HistoryGroup

The HistoryGroup component provides a way to organize alternate navigation presentations for a station's history space. Use the properties in this component to specify metadata properties for grouping histories. Add the HistoryGroup component to the HistoryGroupings container-component by dragging and dropping it from the history palette or by clicking the **New** button in the History Group Manager view.

Also see the following topics:

- "About history grouping"
- "About the History Group property sheet view"
- "About the History Group Manager view"

history-HistoryGroupings

The HistoryGroupings component is available in AX-3.5 as a property of (frozen slot on) the HistoryService component. This component serves as a container for HistoryGroup components. The default view of the HistoryGroup component is the HistoryGroupManager view. You can add HistoryGroup components under the HistoryGroupings component by dragging and dropping a HistoryGroup from the History palette or by You can add HistoryGroup components to this component by clicking the **New** button in the History Group Manager view.

Also see the following topics:

- "About history grouping"
- "About the History Group property sheet view"
- "About the History Group Manager view"

history-HistoryId

The HistoryId component is a container for History id.

history-HistoryPointList

This component is available in the history module. It is a container that holds the "history-HistoryPointListItem" components that you add to it. The default view of this component is the Live History Chart view. The property sheet view is where you configure the HistoryPointList properties.

The HistoryPointList component properties include the following;

- Max Samples
Defines the maximum number of samples to store for each history extension displayed.
- Time Window

Defines the time range to display on the Time Axis while viewing the live chart. If the chart is "zoomed" you will be able to pan and view all the collected data defined by Max Samples.

- Background

Specifies the background color of the chart. The default setting is "null".

- Show Horizontal Grid Lines

Shows (true) or hides (false) horizontal grid lines on the chart.

- Show Vertical Grid Lines

Shows (true) or hides (false) vertical grid lines on the chart.

NOTE: If the Live History Chart view is invoked from a history extension, the following default property values are used to display the view:

- Max Samples value = 5000
- Time Window value = 10 minutes
- Start Time = 0 hours 0 minutes
- Sample Rate = Auto

In addition to the HistoryPointList properties, the property sheet view displays all associated HistoryPointListItem components below the properties. If the view is invoked from the History Point List component the items in the list can be selected/unselected and the view will update as needed.

Refer to "history-HistoryPointListItem" and "About the Live History Chart View" for more information.

history-HistoryPointListItem

This component is available in the history module. Each HistoryPointListItem is linked to a single history extension and is configured in the Add Item dialog box or in the HistoryPointListItem property sheet. HistoryPointListItem configurable properties include the following.

- History Extension

The Ord to the History Extension of the Control Point you wish to view. This **MUST** be the History Extension and **NOT** the Control Point or its corresponding History Component.

- Display On Startup

Indicates whether or not to automatically display the chart for this item when the view first comes up.

- Start Time

Indicates how much of the collected history data you wish to view. The time that is entered in this property will be subtracted from the current time to get the initial history data.

NOTE: The following conditions apply:

If no data is found within the given Start Time the **LAST** collected value is retrieved for the item.

If the Start Time is 0 hours 0 minutes **ALL** collected history values is retrieved for the item.

- Sample Rate

The rate at which to sample the "live" data.

- Auto

This value follows the interval defined in the history extension configuration

- Fixed

This value follows a defined interval that is specified by the user in terms of hours, minutes, and seconds.

- Cov:

This value specifies that updates are plotted whenever there is a change of value.

- **Min Value Range**

The minimum value to display on the value axis.

- Auto

This value uses the minimum value from the data collected.

- Fixed

This value uses the value specified by the user. This only applies to numeric points (Booleans and Enums do not use this property).

- **Max Value Range**

This value is the maximum value to display on the value axis.

- Auto

This value uses the maximum value from the data collected.

- Fixed

This value uses the value specified by the user. This only applies to numeric points (Booleans and Enums do not use this property).

NOTE: If Min Value Range and Max Value Range are Fixed AND have the same values charts may share the value axis. If these two conditions do not exist, additional value axes may be added to the chart, as needed.

- **Line Color**

This value specifies the desired color for the plot line.

- **Pen**

This value specifies the line-weight and type to use for the plot line.

- **Item Name**

This value specifies the name that is displayed on the view. This name must be unique within the list of items.

Refer to “[history-HistoryPointList](#)” and “[About the Live History Chart View](#)” for more information.

history-HistoryService

The History service provides http access to all histories in the Station. This service manages the History life-cycle, including creation and deletion. Additionally, HistoryService maintains the namespace for all Histories. As histories are created, they are added as slots to this service, and removed upon deletion. When the service is started, all existing histories are added to the service. Additionally, this service maintains global configuration for the histories. Each Station contains a single HistoryService. The HistoryService is available in the history module.

See the “[About the history service](#)” for more information.

CloseUnusedHistories

Close all Unused Histories in this service.

FlushHistories

Flush all histories managed by this service.

history-HistoryShortcut

The HistoryShortcut (History Nav Shortcut) component provides a way to include convenient access to histories from various locations in a station nav tree hierarchy.

The HistoryShortcuts component is available on the History palette. For details, see “About history nav shortcuts”.

history-IntervalAlgorithm

IntervalAlgorithm logs a value periodically at a fixed interval. The IntervalAlgorithm is available in the history Module Extensions Directory under some Interval extensions as Collector.

history-LocalDatabaseConfig

LocalDatabaseConfig is the configuration for a LocalHistoryDatabase.

history-LogHistoryService

The LogHistoryService keeps a History of Niagara log records. If enabled, it registers itself as the LogHandler for the system when the service is started. The LogHistoryService is available in the History Module.

history-NumericCovAlgorithm

NumericCovAlgorithm determines when to log a numeric point's value according to change of value. The NumericCovAlgorithm is available in the history Module Extensions Directory under NumericChangeOfValue as Collector.

history-NumericCovHistoryExt

NumericCovHistoryExt is a history extension used to record on change of value for numeric point data. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” for more information about this extension.

history-NumericIntervalHistoryExt

NumericIntervalHistoryExt is a history extension used to record on interval for numeric point data. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” for more information about this extension.

history-StringCovHistoryExt

StringCovHistoryExt is a history extension used for collecting a string control value on change of value. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” for more information about this extension.

history-StringIntervalHistoryExt

StringIntervalHistoryExt is a history extension for collecting a string control value at intervals. This extension is available in the history Module Extensions Directory. Refer to “Types of history extensions” for more information about this extension.

Components in net module

The net module contains the components listed below.

- InetAddress
- 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:
 - Status

This display-only property displays the status of the Http Proxy Service.
 - Fault Cause

This display-only property provides an error message that indicates the reason for a fault.
 - Enabled

This property has a true (enabled) and false (disabled) option.
 - Server

This property provides a text field for entering the address of the proxy server you are connecting to.
 - Port

This property is for specifying the port number for communicating with the proxy server.
 - Exclusions

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

This property provides the following two options:

 - None

no authentication is required at the proxy server when this option is set.
 - Basic

basic authentication is required at the proxy server when this option is set.
 - User

This is the login name to be used when authentication is set to Basic.
 - Password

This is the password text that is to be used when authentication is set to Basic.

Components in onCall module

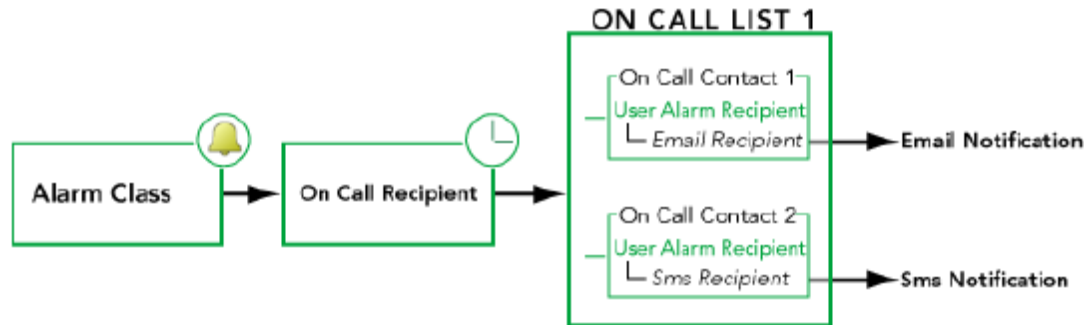
- onCallService
- onCallRecipient

- OnCallSchedule

onCall-OnCallService

This component is a customization of the AlarmService. It expands the features of standard alarm escalation to notify users based on a priority list (contact list). The `OnCallService` initiates an email (or text message) notification for designated alarms, sending the notification sequentially, as alarms escalate, to users based on their assigned priority. The `OnCallService` allows you to set up a flexible user contact list and schedule that list using the standard scheduler interface.

Figure 128 On call processing



The general process for the `OnCallService` is as follows:

1. A designated alarm class receives an alarm notification and sends the alarm notification to an `OnCallRecipient`.
2. The `OnCallRecipient` sends the alarm notification to the active `OnCallList`.

The active `OnCallList` is designated by the **On Call List Schedule**. The `OnCallList` specifies which users (`OnCallContacts`) are notified. The `OnCallContact` properties include a reference to a **User Alarm Recipient** which specifies how the alarm notification is sent. For example, **User Alarm Recipient** types include:

- **EmailRecipient** (sends alarm notification by email)
 - **SmsRecipient** (sends alarm notification by text message)
3. After a specified time, if no recipient acknowledges the alarm, the system escalates the alarm and sends an email or text message to the next specified contact. The system continues to escalate each alarm until a recipient acknowledges it or until it reaches the final escalation level. At any time, any user on the contact list may acknowledge the alarm and halt the escalation process.

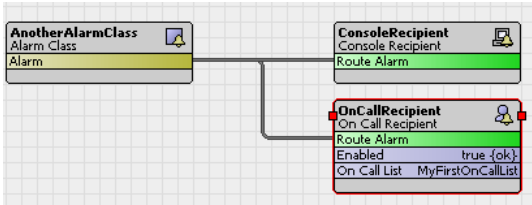
To add the `OnCallService` to a station, drag a copy of the `OnCallService` from the `onCall` palette to the `Config→Services` node in the Nav tree.

onCall-OnCallRecipient

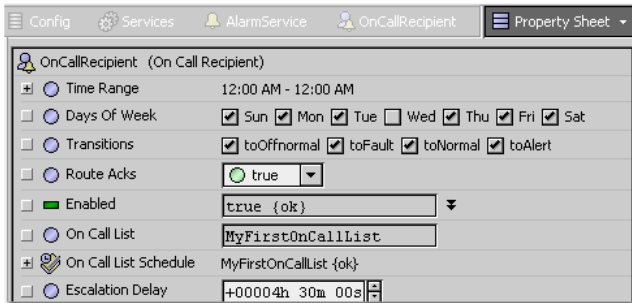
This component manages the transfer of alarms between the `AlarmService` and on call contacts. For example, a station sends alarm notifications by email or text message to one or more contacts that are in the on call contact list. You access this component's property sheet by double-clicking the `OnCallRecipient` component.

NOTE:

Stations can send email by broadband (Niagara outgoing account configuration required) and Sms messages using a GPRS modem.



The **OnCallRecipient** component is linked to an alarm class component that provides a place to specify the scheduling details and other routing options. A special on call scheduling component is contained in the **OnCallRecipient** component. It provides a standard scheduling view that is similar to other schedule views.



Property	Value	Description
Time Range	Start Time and End Time	Start Time sets the time of day to begin the function (for example, trigger schedule, alarm event)
Days of the Week or Days of Week		
Route Acks	true or false	Enables and disables the routing of alarm acknowledgements to the recipient.
Enabled [general]	true or false	Activates and deactivates use of the function.
On Call List	text	Specifies the current active OnCallList and its status. This is the list that the OnCallContact is a member of. A contact may be a member of more than one contact list.
Escalation Delay	hours:minutes:seconds	Sets the time to wait for an alarm to be acknowledged by an OnCallContact before escalating the alarm to the next contact in the OnCallList .

onCall-OnCallSchedule

This component sets times for making On Call Lists active. It is available in the **onCall** palette.

Actions

Actions include **Cleanup**.

onCall-OnCallList

This component contains a set of one or more people (**OnCallContacts**) to contact when an alarm event occurs. You create, edit, or delete unique on call lists using the **On Call List Manager** view.

When you create a list, it appears under the **OnCallService** node in the Nave tree and is also available as an option on the Event Output property of the on-call list schedule's **Scheduler** view from where you may select it as a scheduled event.

Like other scheduled outputs, you may assign more than one on-call list to a single day. The on call list is active only during the scheduled day and time. The on-call list active status is displayed in the on call **List** property sheet view as well as in the **On Call List Manager** view.

NOTE: When assigning `OnCallList` events in the **Scheduler** view, make sure **OnCallList** event times are contiguous.

Property	Value	Description
Active	Active or Inactive	Displays the current state of the list as defined by the On Call List Schedule. Only one On Call List is active at a time.
Last Fault Cause		

Components in program module

- Program
- ProgramModule
- ProgramService

program-Program

Program uses an instance based classfile to implement user defined component logic. The Program can be viewed and edited using the ProgramEditor. Program is available in the program palette.

Actions

Actions include **Execute**.

program-ProgramModule

ProgramModule provides a **Program Module Builder** view used to build standard Niagara modules from Program components. This provides a mechanism to version and provision Program modules just like any other Niagara modules.

Starting in AX-3.5, the ProgramModule is available in the program palette.

program-ProgramService

The **ProgramService** provides a (default) **Batch Editor** view to launch a batch operation on multiple selected component slots. For more details, refer to the *Batch Editor - Engineering Notes* document.

The ProgramService also provides a secondary RobotEditor 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

Actions include **Run Robot**.

Components in the Sms module

The Sms module contains components that allow you to set up a service for sending text messages. The following components are included in the Sms module:

- SmsService
- SmsRecipient

- Sms
- SmsAlarmAcknowledger

sms-SmsService

The Sms Service is the main component used for sending text messages. The service consists of a Transport object. The Transport object is used to send an Sms text message. The Transport object can be changed via altering the 'Transport Type' Property. All text messages are queued under the Sms Service. If there are any text messages queued when the station shuts down, they will be sent when the station next starts up. See also *SmsService*, in the *Sms User Guide*.

Actions on this component include the following:

- Ping
- Send
- Clear Queue
- Process Queue
- Reset Number Sent Today
- Read Meessages

sms-SmsRecipient

This component acts as an alarm recipient for transmitting alarms via a text message. Its use is similar to the EmailRecipient component. The user is able to format the text for the SMS message within the body slot. Note the body of an SMS text message is limited to 140 characters.

Actions on this component include: Route Alarm

sms-Sms

This component is a general use Sms component to send text messages. The phone number for the recipient must be in International format and is configured via the component's property sheet.

Actions available on this component include:

- Send Default Message
- Send Message

sms-SmsAlarmAcknowledger

The SmsAlarmAcknowledger component (AX-3.5 and later) provides a way for users to acknowledge alarms by sending a reply to a text message alarm notification. This component is available in the Sms palette and works with the with the On Call Service or with the Sms Service.

Actions available on this component include: Received

Components in schedule module

Schedules allow you to program commands to occur at specific times. Each schedule component is designed to provide an output based on the current time. Components can receive this command by linking to this output. See the "Weekly Scheduler view" for more information on using schedules.

- BooleanSchedule
- CalendarSchedule
- EnumSchedule
- NumericSchedule

- StringSchedule
- TriggerSchedule
- BooleanScheduleSelector
- NumericScheduleSelector
- StringScheduleSelector
- EnumScheduleSelector

schedule-BooleanSchedule

A deployable weekly schedule that provides a continuous StatusBoolean output. Other weekly schedule types include EnumSchedule, NumericSchedule, and StringSchedule. See "About weekly schedules" for more details.

Input - If the "in" property is linked and its value is non-null, then this value overrides the scheduled output. The BooleanSchedule is available in the schedule palette.

schedule-CalendarSchedule

The CalendarSchedule provides a calendar for scheduling holidays or other schedule overrides. The CalendarSchedule is available in the schedule palette.

schedule-EnumSchedule

A deployable weekly schedule that provides a continuous StatusEnum output. Other weekly schedule types include BooleanSchedule, NumericSchedule, and StringSchedule. See "About weekly schedules" for more details.

Input - If the "in" property is linked and its value is non-null, then this value overrides the scheduled output. The EnumSchedule is available in the schedule palette.

schedule-NumericSchedule

A deployable weekly schedule that provides a continuous StatusNumeric output. Other weekly schedule types include BooleanSchedule, EnumSchedule, and StringSchedule. See "About weekly schedules" for more details.

Input - If the "in" property is linked and its value is non-null, then this value overrides the scheduled output. The NumericSchedule is available in the schedule palette.

schedule-StringSchedule

A deployable weekly schedule that provides a continuous StatusString output. Other weekly schedule types include BooleanSchedule, EnumSchedule, and NumericSchedule. See "About weekly schedules" for more details.

Input - If the "in" property is linked and its value is non-null, then this value overrides the scheduled output. The StringSchedule is available in the schedule palette.

schedule-TriggerSchedule

A schedule that fires topics - there is no continuous output. The TriggerSchedule is available in the schedule palette.

schedule-BooleanScheduleSelector

ScheduleSelector components provide an easy way to select a preconfigured schedule of the proper data type for controlling a particular component. BooleanScheduleSelector components only link BooleanSchedule components to a control component of the Boolean data type.

See also,

- About ScheduleSelector components
- Types of ScheduleSelector components
- Types of ScheduleSelector component properties
- Example ScheduleSelector configurations

schedule-NumericScheduleSelector

ScheduleSelector components provide an easy way to select a preconfigured schedule of the proper data type for controlling a particular component. NumericScheduleSelector components only link NumericSchedule components to a control component of the Numeric data type.

See also,

- About ScheduleSelector components
- Types of ScheduleSelector components
- Types of ScheduleSelector component properties
- Example ScheduleSelector configurations

schedule-StringScheduleSelector

ScheduleSelector components provide an easy way to select a preconfigured schedule of the proper data type for controlling a particular component. StringScheduleSelector components only link StringSchedule components to a control component of the String data type.

See also,

- About ScheduleSelector components
- Types of ScheduleSelector components
- Types of ScheduleSelector component properties
- Example ScheduleSelector configurations

schedule-EnumScheduleSelector

ScheduleSelector components provide an easy way for to select a preconfigured schedule of the proper data type for controlling a particular component. EnumScheduleSelector components only link EnumSchedule components to a control component of the Enum data type.

See also,

- About ScheduleSelector components
- Types of ScheduleSelector components
- Types of ScheduleSelector component properties
- Example ScheduleSelector configurations

Components in timesync module

- TimeSyncClient

- TimeSyncService

timesync-TimeSyncClient

Slot of a TimeSyncService used to poll a time protocol server. The TimeSyncClient is available in the timesync palette.

NOTE: The **NtpPlatformService** has largely replaced this older RFC 868-based timesync module and components, since AX-3.4. For details, refer to “About the NtpPlatformService” in the *Platform Guide*.

timesync-TimeSyncService

NOTE: The **NtpPlatformService** has largely replaced this older RFC 868-based timesync module and components, since AX-3.4. For details, refer to “About the NtpPlatformService” in the *Platform Guide*.

A time protocol (RFC 868) client and server. To synchronize with other servers, TimeSyncClients must be added to the service. As a convenience, a disabled default client has already been added (it will need configuring). The service uses the clients to poll their configured servers, and if the service detects a drift greater than the tolerance property, the service will reset the system time. The module palette contains clients pre-configured for well known time protocol servers. The primary view of this Component is TimeSyncManager.

From the PropertySheet, you can set Enabled and Server Enabled.Enabled (to false) to disable the time sync client. Server Enabled enables the time sync server. In order to sync between stations, enable master TimeSync service to be a server, then add a TimeSyncClient in the station, configured for the server Station. The TimeSyncService is available in the timesync palette.

ForceSync

Force Time Synchronization.

Sync

Time Synchronization.

Components in web module

- WebService

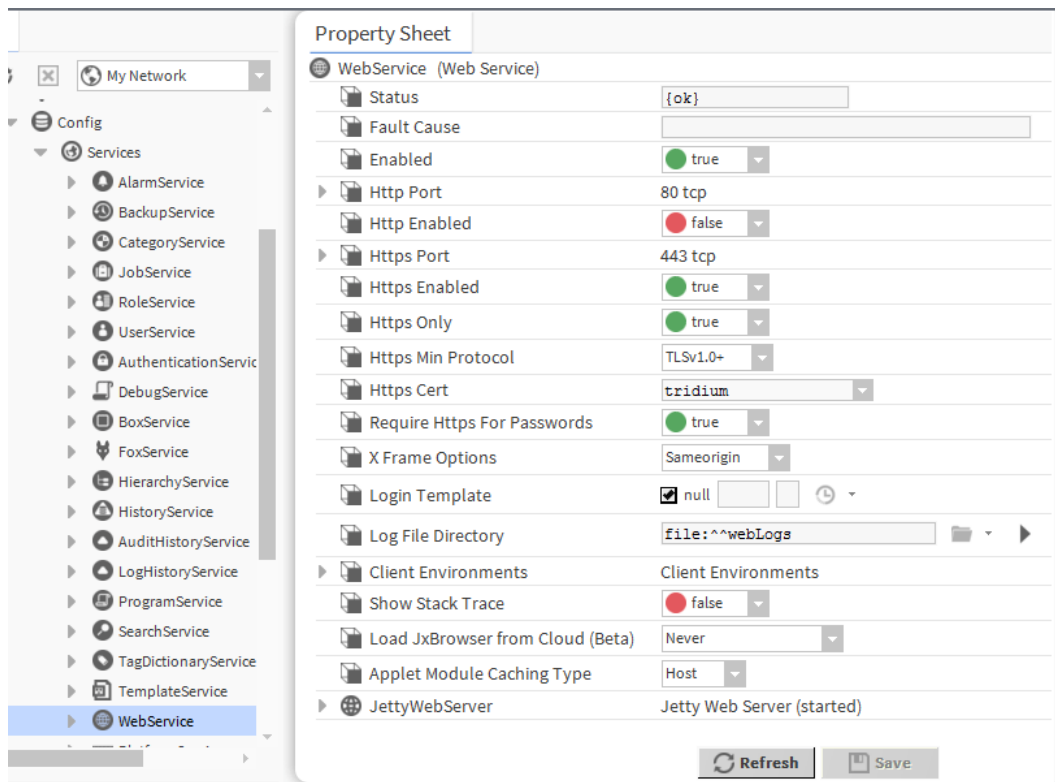
web-MobileClientEnvironment

MobileClientEnvironment (mobile) is a child of the ClientEnvironments container of a station’s WebService, and present only if the host is licensed with the `mobile` feature. It is used in the automatic selection of the appropriate webProfile type for a user, based on the detection of the incoming browser client type (e.g. desktop or mobile). For details, refer to “About the Mobile Client Environment (mobile) property” in the *NiagaraAX Mobile Guide*.

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 129 Example of Webservice properties



Name	Value	Description
Status [component]	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> • {ok} indicates that the component is polling successfully. • {down} indicates that polling is unsuccessful, perhaps because of an incorrect property. • {disabled} indicates that the Enable property is set to false. • fault indicates another problem.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Enabled [general]	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, Http Enabled in the FoxService must also be set to true (for wapplet 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 Https Only is enabled, this setting (false for Http Enabled) is irrelevant.

Name	Value	Description
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, Foxxs Enabled in the FoxxService must also be set to true (for webapplet 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	
X Frame Options	Sameorigin (default) Deny or Any (least secure)	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 <frame> or <iframe>. You can use this to avoid clickjacking attacks, by ensuring that content is not embedded into other sites. 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. If you specify the Deny , attempts to load the page in a frame will always fail. NOTE: The Deny option inhibits display of some typical Shell Hx Profile views.

Name	Value	Description
		CAUTION: If you specify the <code>Any</code> option then Cross-Frame Scripting (SFS) and Cross-Site Scriptin (XSS) are allowed.
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		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		This property is a container for Mobile Client Enviroment (mobile) entries, available if the station's host is licensed with the mobile feature. It is used in detection of a user's browser type (e.g. desktop or mobile) and the selection of the appropriate webProfile for that user. For details, refer to <i>About the Mobile Client Environment (mobile) property</i> in the <i>NiagaraAX Mobile Guide</i> .
Show Stack Trace	true or false (default)	
Load JxBrowser from Cloud	Never	
Applet Module Caching Type	Host (default) or User	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 a JACE.
JettyWebServer	read-only	Jetty Web Server Started

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

Name	Value	Description
Enabled [general]	true or false	Activates and deactivates use of the function.
Status [component]	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> • <code>{ok}</code> indicates that the component is polling successfully.

Name	Value	Description
		<ul style="list-style-type: none"> <code>{down}</code> indicates that polling is unsuccessful, perhaps because of an incorrect property. <code>{disabled}</code> indicates that the <code>Enable</code> property is set to false. <code>fault</code> indicates another problem.
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.

JettyWebServer

Name	Value	Description
Server State		
Min Threads	3 (default)	
Max Threads	30 (default)	
N C S A Log	NCSA Request Log	See N C S A Log, page 197

N C S A Log

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

Name	Value	Description
Enabled [general]	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	
Log Cookies	true or false (default)	
Log Time Zone	list of time zones	Identifies the time zone to use for time stamps.

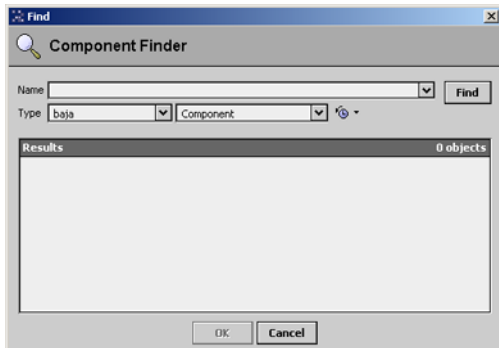
Components in workbench module

- ComponentFinder
- KioskService
- WbFieldEditorBinding
- WbViewBinding
- WsOptions


workbench-ComponentFinder

The ComponentFinder dialog provides a means of finding Components.

Figure 130 Sample Window



workbench-KioskService

 workbench-KioskService is used to enable the Kiosk mode. Kiosk mode provides a way to run a Niagara 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" folder (Config > Services) 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



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

- Show Thumbnail
- Show Grid
- Show Status Colors

These are stored under `/user/{user}/wiresheet.options`.

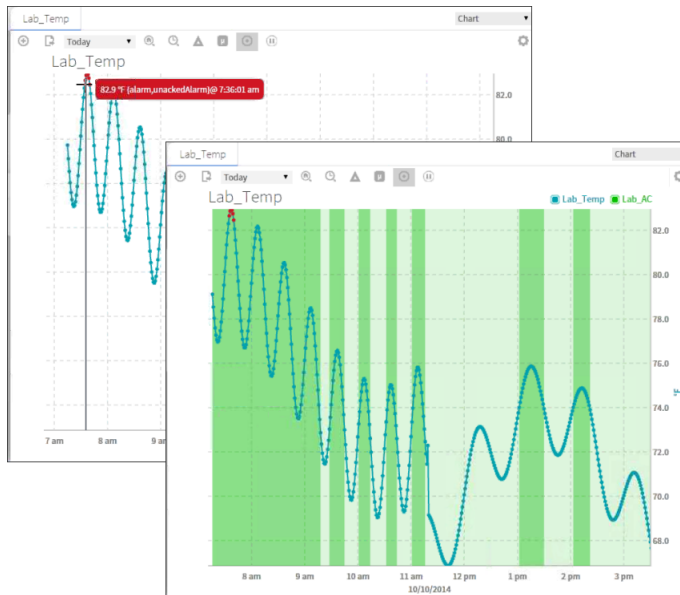
workbench-WebWidget

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:

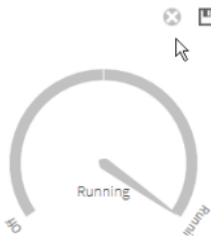
- The `WebChart` displays the **Chart** view which can display historical data and update with live data. Also, in the view you can easily add data and invoke numerous commands and settings to modify data presentation.

Figure 131 Chart WebWidget



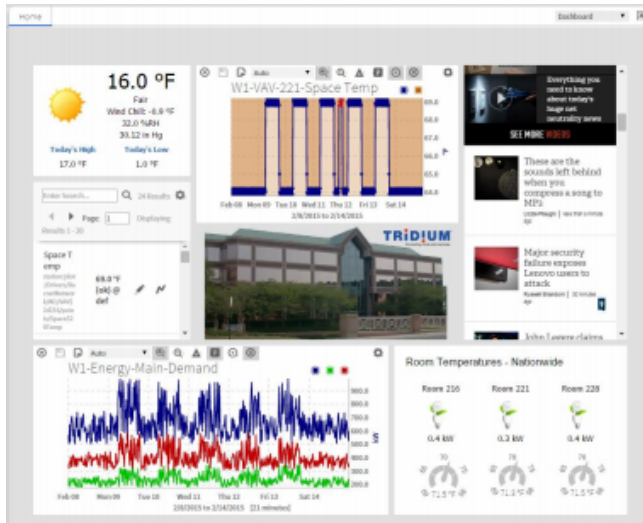
- The `CircularGauge` displays the graphical gauge view which updates with live data and provides contextual information for the current value. At any time you can dynamically switch the display to another component simply by dragging and dropping a different component onto this widget.

Figure 132 CircularGauge WebWidget



- The `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.

Figure 133 Dashboard WebWidget



Chapter 6 Plugin Guides

Topics covered in this chapter

- ◆ 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 onCall module
- ◆ Plugins in program module
- ◆ Plugins in raster module
- ◆ Plugins in schedule module
- ◆ Plugins in timesync module
- ◆ Plugins in wiresheet module
- ◆ 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 timesync module
- Plugins in wiresheet module
- Plugins in wbutil module
- Plugins in workbench module

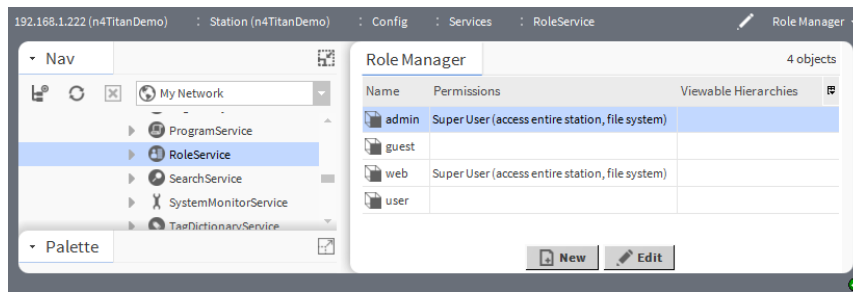
Plugins in alarm module

- alarm-AlarmConsole
- alarm-AlarmDbMaintenance
- alarm-AlarmDbView
- alarm-AlarmClassSummary
- alarm-AlarmExtManager
- alarm-AlarmInstructionsManager
- alarm-AlarmPortal

wbutil-RoleManager

This manager allows you to create, edit and delete roles. It is the default view of the RoleService and is located in the station's **Services** container.

Figure 134 Role Manager view

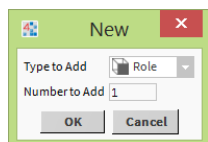


The system creates the `admin` role by default and grants it super user permissions. The `admin` role does not appear in the **Role Manager** view and you may not delete it.

Column or field	Value	Description
Name	text	Identifies the role to be assigned to one or more users. Role names are case sensitive.
Permissions	text	Associates a name with a specific set of permissions.
Viewable Hierarchies	text	Identifies the hierarchies this user may view.
Type	Role (default)	Identifies the type of entity being created.
Number to add	number	Allows you to create many rows at once in the Role Manager view's table.

New role window

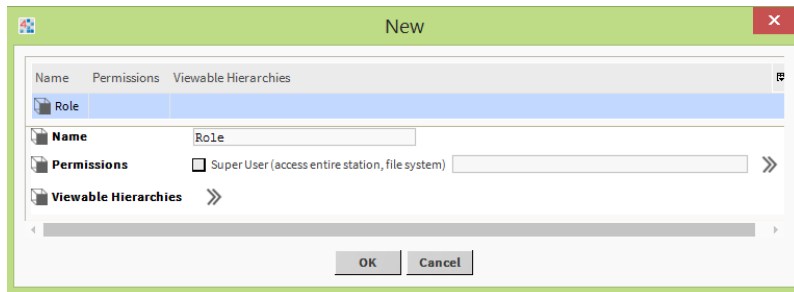
Figure 135 New Role window



Column or field	Value	Description
Type	Role (default)	Identifies the type of entity being created.
Number to add	number	Allows you to create many rows at once in the Role Manager view's table.

New role properties

Figure 136 New role properties



Property	Value	Description
Name		
Permissions		
Viewable Hierarchies		

alarm-AlarmDbMaintenance



AlarmDbMaintenance is a view of the alarm service. It allows you view and manage alarm records.

For more information refer to:

- "About the alarm database maintenance view"
- "About the alarm service"

alarm-AlarmDbView



AlarmDb is a view of the alarm service. It allows you view and sort alarm records.

For more information refer to:

- "About the alarm service"

alarm-AlarmClassSummary



AlarmClassSummary is a view of the alarm service. It presents a table of information about all alarm classes that are assigned to the alarm service.

For more information refer to:

- "About the alarm class summary view"
- "About the alarm service"

alarm-AlarmExtManager

AlarmExtManager is a view on the AlarmService. It provides a convenient way to manage Alarm extensions.

The Alarm Ext Manager view provides a table view of Alarm Source Extensions. The table includes the following information, as available, about each alarm extension:

- Point
- Extension
- Alarm State
- toOffnormal Enabled
- toFault Enabled
- Alarm Class
- Instructions

alarm-AlarmInstructionsManager



AlarmInstructionsManager view displays a standard table-type report that provides a way to view, assign, and edit alarm instructions.. For details, refer to “About the Instructions Manager view”.

alarm-AlarmPortal



The Alarm Portal allows you to view and acknowledge Alarms from multiple Stations. It can be started from the main Menu by selecting **Tools→AlarmPortal**.

From the Alarm Portal, you can view the 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:

- Acknowledge
- Silence
- Hyperlink
- Notes
- Filter

Plugins in backup module

- backup-BackupManager

backup-BackupManager



The BackupManager 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** dialog appears to select the destination directory on your PC (see File Ord Chooser) and the file name for the backup dist file.

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

- Workbench (Fox) — !backups

A subdirectory “backups” under your Niagara release directory. If you have not previously made station backups, this directory is automatically created.

- Browser access (Web Web Profile) — !backups

A subdirectory “backups” under the Niagara subfolder of your installed Java 2 runtime environment (Java plugin).

For example: C:\Program Files\Java\j2rel1.4.2_05\niagara\backups.

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_J403IP98c_050618_1429.dist” for a backup made of station “J403IP98c” on June 18, 2005 at 2:29 pm.

In the BackupManager, a progress bar and Job Log (>> control) is provided for an initiated backup.

The BackupManager displays a table of the 10 most recent backups, with the following data columns:

- Timestamp
Station date and time when the backup was initiated.
- Host
IP address of the requesting (remote) PC for the backup.
- Path
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
The station user that initiated the backup.

Plugins in chart module

- chart-ResourceManager

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.

Plugins in email module

- email-EmailAccountManager

email-EmailAccountManager

✉ Email Account Manager allows you to view email accounts. It is the default view of the EmailService.

Plugins in help module

The following plugin is in the help module

- help-BajadocViewer

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. It shows documentation for the following:

- Subclasses
- Properties
- Actions
- Topics
- Constructors
- Methods
- Fields

In order to view Bajadoc, either right-click on a Bajadoc file and select **Views**→**Bajadoc Viewer** or select **Bajadoc On Target** from the main **Help** Menu or popup Menu.

See Object Model in the Developer Guide for more information.

BajadocViewer Menus

The **BajadocViewer** menu functions include:

- Show Baja Only
- Flatten Inheritance

Show Baja Only

This option allows you to view only 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

This option allows you to flatten the inheritance hierarchy into a single set of documentation. When it is false only the Java members and Baja slots declared in the specified class are displayed. When it is set to true all the Java members and Baja slots inherited from super classes are also shown.

Subclasses

Subclasses provides a subclass tree of all that are subclassed from this item.

Properties

Properties represent a storage location of another Niagara object. Flags are boolean values which are stored as a bitmask on each slot in a Baja Object. Some flags apply to all slot types, while some only have meaning for certain slot types.

Flags

Flag	Char	Applies	Description
readonly	r	P	The <code>readonly</code> flag is used to indicate slots which are not accessible to users.
transient	t	P	Transient properties do not get persisted when saving a object graph to the file system. Transient properties are usually also readonly, unless they are designed to be a linkable input slot.

Flag	Char	Applies	Description
hidden	h	P,A,T	Hidden slots are designed to be invisible to the user, and exist only for Java developers. User interfaces should rarely display hidden slots.
summary	s	P	Summary properties are the focal points of any given BComponent. This flag is used by user interface tools to indicate primary properties for display. This might be as a columns in a Table, or as a glyph in a graphical programming tool.
async	a	A	By default Action are invoked synchronously on the callers thread. By using the <code>async</code> flag on an Action, invocations are coalesced and executed asynchronously at some point in the near future on the engine's thread.
noExecute	x	P	No execute properties prevents start/stop from recursing on properties with this flag set.
defaultOnClone	d	P	Specifies that when an object is cloned via the <code>newCopy()</code> method these properties retain their default value, not the clone source's value.
confirmRequired	c	A	When the Action is invoked by a user, a confirmation dialog must be acknowledged before proceeding.
operator	c	P,A,T	This makes a slot as operator security level. By default when this flag is clear, the slot is an admin security level.
userDefined1	1	P,A,T	User defined.
userDefined2	2	P,A,T	User defined.
userDefined3	3	P,A,T	User defined.
userDefined4	4	P,A,T	User defined.

See Config Flags for information on configuring flags on properties.

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 history module

- history-DatabaseMaintenance
- history-DeviceHistoriesView
- history-GroupManager
- history-HistoryChart
- history-HistoryChartBuilder
- history-HistoryEditor
- history-HistoryExtManager
- history-HistorySummaryView
- history-HistoryTable
- history-LiveHistoryChart

history-DatabaseMaintenance



The DatabaseMaintenance allows you to maintain the histories in a station from the Workbench. You can Clear Old Records, Clear All Records or Delete Histories.

history-DeviceHistoriesView



The Device Histories View is available on any device that supports import and export of histories (for example, Niagara devices, BACnet devices, and others). The view shows a filtered list of history shortcuts for that particular device. This view displays all the related shortcuts in a table. You can double-click on any single entry in the table to open that history in the History Chart view.

history-GroupManager



The HistoryGroupManager provides a tabular view of all history groups. It is the default view of the History-Groupings component and provides information about the groups contained in the HistoryGroup component.

Refer to the following sections for related information:

- “About history grouping”
- “About the History Group property sheet view”
- “About the History Group Manager view”

history-HistoryChart





The HistoryChart is a view that allows you to display and configure charts of history records. Any point History can be charted by double-clicking directly on it under the History node in the nav tree.

The HistoryChart properties are configurable when editing the HistorChart view in a Px page.

Figure 137 History Chart display properties are configurable in the Px page view

History Chart		
Choose how to display these chart features on initial view	defaultDelta	false
	defaultLiveUpdates	false
	defaultTimeRange	timeRange:startTir
	enabled	true
	layout	0,0,450,350
Choose to show or hide these chart features individually	showDeltaEditor	true
	showLiveUpdates	true
	showTimeRangeEditor	true
	visible	true

- **defaultDelta**
When set to true, this property causes the history chart view to plot delta values on initial display and any time the view is refreshed.
- **defaultLiveUpdates**
When set to true this property sets the chart view to Live Update mode on initial display and whenever the view is refreshed.
- **defaultTimeRange**
This property allows you to choose a specific time range option that appears on initial display and whenever the history chart view is refreshed.
- **enabled**
This property allows you to disable (false) or enable (true) the history chart view.
- **layout**
This property provides four values that define the size of the history chart view display area.
- **showDeltaEditor**
This property allows you to hide (false) or show (true) the delta mode button  on the history chart view.
- **showLiveUpdates**
This property allows you to hide (false) or show (true) the live update button  on the history chart view.
- **showTimeRangeEditor**
This property allows you to hide (false) or show (true) the Time Range Editor option list on the history chart view.
- **visibility**
This property allows you to hide (false) or show (true) the history chart view.

history-HistoryChartBuilder



The HistoryChartBuilder is a view that allows you to view and configure charts of one or more histories. Add a history by double-clicking or dragging into the configuration area. For any chart, you can specify a Time Range, Title, and whether to show grid lines. Each history has a selectable Chart Type of either Line or Discrete Line. After you Build a chart, you can Save it.

history-HistoryEditor



The HistoryEditor allows you to view and query a History in a Station.

The HistoryEditor allows you to  Hide or  Unhide selected records.

history-HistoryExtManager



The HistoryExtManager provides a tabular view of all history extensions. It is the default view of the History-Service and provides information about the control point, extension type, name and status of each entry. Refer to the following references for information about the history extension manager, its associated menus, toolbar icons, and other details.

Refer to the following sections for related information:

- “About the History Ext Manager menu”
- “About the history extension manager view”
- “About the history extension manager popup menu items”

history-HistorySummaryView



The HistorySummaryView allows you to view the summary information about a History from the Workbench.

history-HistoryTable



The HistoryTable is a view that allows you to view tables of histories. The selected entries are shown as timestamp, trendFlags, status and value in each row of the Table.

You can resize columns in the HistoryTable. If you move the cursor over the line between columns, it will change to a resize cursor. You can then drag the line to the desired size.

If want to reorder the columns in the HistoryTable, drag the column header to a new location.

Query



Query lets you select the period for the query. Selections include:

- Time Range
- Today
- Yesterday
- Last Week
- Last 7 Days
- Last Month
- Last Year
- Month-to-date
- Year-to-date

Filter



Filter lets you use the filter you have configured for the data in the Table.

Filter Config



Filter lets you configure the filter for the data in the Table.

history-LiveHistoryChart



Background and live history charting is available for history data. This feature includes the ability to display historical data (trend data) in a Live History Chart view that plots a range of data from a configurable start time to the current time and continues plotting as new data is generated by the source. Refer to “About the Live History Chart View” for more details.

Plugins in html module

- html-WbHtmlView
- html-SpyViewer

html-WbHtmlView

- Introduction
- HTML Support
- Stylesheet Support

Introduction



The WbHtmlView allows you to view HTML files.

WbHtmlView Menus

The Workbench main menu functions are available.

WbHtmlView Toolbar

In the WbHtmlView, the toolbar contains navigation and editing buttons as described in “About the toolbar”. In addition, Find, FindNext and FindPrev toolbar buttons are available.

HTML Support

HTML Tags

The **WbHtmlView** attempts to ignore mismatched or missing Tags. It can parse any HTML, no matter how badly messed up the Tags are, although it might do a pretty bad job of rendering the results. It does simplistically attempt to repair missing <p>, but other problems are solved by ignoring Tags. Warnings will appear on the command line showing its best guess as to what the problem was. Valid tags include:

- a - anchor Ex:Title<a> and Title<a> (Attributes href and name)
- applet - not supported
- b - bold text style Ex: bold text
- 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: emphasis text
- 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 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://bajoui/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

HTML Attributes

HTML Elements have associated properties, called attributes, which may have values. Attribute/value pairs appear before the final ">" of an element's start tag. Valid attributes include:

- `align` - vertical or horizontal alignment **Deprecated**; elements: `img`, `object?`; values: `bottom`, `left`, `middle`, `right` or `top`.
- `align` - table position relative to window **Deprecated**; elements: `table`; values: `center`, `left` or `right`
- `align` - align, text alignment **Deprecated**; elements: `div?`, `h1?`, `h2?`, `h3?`, `h4?`, `h5?`, `h6?`, `p`; values: `center`, `justify`, `left` or `right`
- `align` - alignment; elements: `col?`, `colgroup?`, `tbody?`, `td`, `tfoot?`, `th`, `thead?`, `tr`; values: `center`, `char`, `justify`, `left` or `right`
- `bgcolor` - background Color **Deprecated**; elements: `h1`, `h2`, `h3`, `h4`, `h5`, `h6?`, `p`, `td`, `th`, `tr`
- `border` - controls frame width around table; elements: `table`
- `cellpadding` - spacing within cells; elements: `table`
- `cellspacing` - spacing between cells; elements: `table`
- `class` - class name or set of class names for stylesheets; elements: most elements
- `color` - text Color **Deprecated**; elements: `font`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6?`, `p`, `pre`, `td?`, `th?`, `tr?`
- `colspan` - number of cols spanned by cell; elements: `td`, `th`
- `content` - associated information; elements: `meta?`
- `href` - URI for linked resource; elements: `a`, `link`
- `id` - name to an element; elements: most elements
- `lang` - Language Code; elements: most elements not supported
- `name` - named link end; elements: `a`
- `name` - meta information name; elements: `meta?`
- `rel` - forward link types; elements: `link`; values: `stylesheet`
- `rowspan` - number of rows spanned by cell; elements: `td`, `th`
- `size` - size of font; elements: `font`; value: fixed 1-7 or relative -7 to +7
- `summary` - purpose/structure for speech output; elements: `table`
- `type` - advisory content type; elements: `link`; values: `text/css`
- `width` - table width; elements: `table`

Character Entity References

Character Entity References are supported including the following:

- `&` - ampersand `&`
- `'` - apostrophe `'`
- `©` - copyright `©`
- `>` - greater than `>`
- `“` - double quotation, left `"`

- `‘`; - single quotation, left `'`
- `<`; - less than `<`
- `—`; - em dash `—`
- ` `; - non breaking space `" "`
- `–`; - en dash `–`
- `”`; - double quotation, right `"`
- `’`; - single quotation, right `'`
- `"`; - quotation mark `"`
- `®`; - registered trademark `®`
- `™`; - trademark `™`

Stylesheet Support

Introduction

Stylesheets are supported using a link in the header as follows:

```
<head>
<title>Sample</title>
<link rel='StyleSheet' href='module://bajai/doc/style.css' type='text/css' />
</head>
```

See the default stylesheet used for Niagara developer documentation: `module://bajai/doc/style.css` or the CSS stylesheet used for this document at `docbook.css`.

Pseudo-classes and Pseudo-elements

Anchor pseudo-classes include:

- `A:link` unvisited links unsupported
- `A:visited` visited links unsupported
- `A:active` active links unsupported

CSS1 Properties

Stylesheet elements supported include:

- `background` - The 'background' property is a shorthand property for setting the individual background properties (i.e., 'background-color', 'background-image', 'background-repeat', 'background-attachment' and 'background-position') at the same place in the style sheet.
- `background-attachment` unsupported
- `background-color` - This property sets the background Color of an element.
- `background-image` unsupported
- `background-position` unsupported
- `background-repeat` unsupported
- `border` unsupported
- `border-bottom` unsupported
- `border-bottom-width` unsupported
- `border-color` unsupported

- border-left unsupported
- border-left-width unsupported
- border-right unsupported
- border-right-width unsupported
- border-style unsupported
- border-top unsupported
- border-top-width unsupported
- border-width unsupported
- clear unsupported
- color - This property describes the text Color of an element (often referred to as the foreground Color).
- display unsupported
- float unsupported
- font unsupported
- font-family unsupported
- font-size unsupported
- font-style unsupported
- font-variant unsupported
- font-weight unsupported
- height unsupported
- letter-spacing unsupported
- line-height unsupported
- list-style-image unsupported
- list-style-position unsupported
- list-style-type unsupported
- margin unsupported
- margin-bottom unsupported
- margin-left unsupported
- margin-right unsupported
- margin-top unsupported
- padding unsupported
- padding-bottom unsupported
- padding-left unsupported
- padding-right unsupported
- padding-top unsupported
- text-align
- text-decoration unsupported
- text-indent unsupported
- text-transform unsupported

- white-space unsupported
- width unsupported
- word-spacing unsupported

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 onCall module

- onCall-OnCallListManager
- onCall-OnCallContactManager
- onCall-OnCallUserReportView

onCall-OnCallListManager



The On Call List Manager view (AX-3.5 and later) is the default view of the On Call Service. This view displays a table listing of all existing On Call lists and allows you to add, delete, and edit On Call Contacts. See also, "About the On Call List Manager view".

onCall-OnCallContactManager



The On Call Contact Manager view (AX-3.5 and later) is the default view of the On Call List component. This view displays a table listing of all contacts in the selected On Call list and allows you to add, delete, and edit On Call Contacts. See also, “About the On Call Contact Manager view”.

onCall-OnCallUserReportView



The On Call user Report view (AX-3.5 and later) is the default view of the On Call Contact component. This view provides a tabular presentation of the details associated with any On Call List that the selected user is assigned to. The purpose of this view is to provide each On Call Contact with a configurable table of information that shows scheduled times and priorities associated with all of On Call Contact Lists that they are assigned to. See also, “About the On Call User Report view”.

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 using drag and drop from the Nav tree, or copy and paste 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 *Batch Editor - Engineering Notes* document.

program-ProgramEditor



The ProgramEditor Plugin provides the ability to view and edit Program Components. To view, Right-click a Program and select **ProgramEditor**. It shows ProgramEditor Edit, ProgramEditor Slots, ProgramEditor Imports and ProgramEditor Source tabs.

ProgramEditor Edit



Edit allows you to edit the **onExecute**, **onStart**, **onStop** and **freeForm** methods. An example from the demo Database follows:

```
BStatusNumeric inA = getInA();
BStatusNumeric inB = getInB();
BStatusNumeric out = getOut();

out.setValue( inA.getValue() + inB.getValue() );
```

ProgramEditor Slots






Slots allows you to view and change the slots of the program component. It includes **Slot, #, Name, Display Name, Definition, Flags** and **Type** for each slot. See SlotSheet for more information on slots.

ProgramEditor Imports



Imports allows you to view the Modules that have been imported.

It also allows the following:

-  ProgramEditor ImportType
-  ProgramEditor ImportPackage
-  ProgramEditor Remove

ProgramEditor ImportType



ImportType allows you to import a new type.

ProgramEditor ImportPackage



ImportPackage allows you to import a new type.

ProgramEditor Remove



Remove allows you to import a new type.

ProgramEditor Source



Source allows you to view and edit the source of the program component. The editor supports special Color coding for Java Files. An example from the demo Database follows:

```
/* 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
////////////////////////////////////
```

```

public BStatusNumeric getOut() { return (BStatusNumeric)get("out"); }
public BStatusNumeric getInA() { return (BStatusNumeric)get("inA"); }
public BStatusNumeric getInB() { return (BStatusNumeric)get("inB"); }

////////////////////////////////////
// Setters
////////////////////////////////////

public void setOut(javax.baja.status.BStatusNumeric v) { set("out", v); }
public void setInA(javax.baja.status.BStatusNumeric v) { set("inA", v); }
public void setInB(javax.baja.status.BStatusNumeric v) { set("inB", v); }

////////////////////////////////////
// onExecute
////////////////////////////////////










public void onExecute()
    throws Throwable
    {
        BStatusNumeric inA = getInA();
        BStatusNumeric inB = getInB();
        BStatusNumeric out = getOut();

        out.setValue( inA.getValue() + inB.getValue() );
    }
}

```







ProgramEditor Menus

The Workbench main menu functions are available. When the ProgramEditor is visible, the following ProgramEditor menu function is also available:

-  ProgramEditor ImportType
-  ProgramEditor ImportPackage
-  ProgramEditor Remove
-  Add Slot Ctrl + A
-  Delete
-  Rename Slot Ctrl + R
-  Reorder
-  ProgramEditor Compile & Save F9
-  ProgramEditor Compile Ctrl + F9

ProgramEditor Toolbar

The Workbench toolbar contains navigation and editing buttons as described in "About the toolbar". When the ProgramEditor is visible, additional toolbar buttons include:

-  Find F5
-  Replace F6
-  ProgramEditor Compile & Save F9
-  ProgramEditor Compile Ctrl + F9
-  Console Prev
-  Console Next

ProgramEditor Compile & Save



CompileSave allows you to compile and save the source of the program component. The shortcut is F9.

ProgramEditor Compile



Compile allows you to compile the source of the program component. The shortcut is Ctrl + F9.

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 RobotEditor 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
- ▶ RobotEditor Compile & Run F9

RobotEditor Toolbar

The Workbench toolbar contains navigation and editing buttons as described in "About the toolbar". When the RobotEditor is visible, additional toolbar buttons include:

- Find F5
- Replace F6
- RobotEditor Compile Ctrl + F9
- ▶ RobotEditor Compile & Run F9
- Console Prev (?need to find SearchConsolePrev.html source?)
- Console Next (?need to find SearchConsoleNext.html source?)

RobotEditor Compile



Compile allows you to compile the source of the Robot component. The shortcut is Ctrl + F9.

RobotEditor Compile & Run



Compile& Run allows you to compile and run the source of the Robot component. The shortcut is F9.

Plugins in raster module

- raster-RasterViewer

raster-RasterViewer



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

Plugins in schedule module

- schedule-CalendarScheduler
- schedule-CurrentDaySummary
- schedule-Scheduler
- schedule-TriggerScheduler

schedule-CalendarScheduler



The CalendarSchedule view is used to define calendar days in a CalendarSchedule. Typically, calendar days represent holidays.

Use the **Add** button to add a calendar day (or range of days). Each entry requires a unique name, a date type, and other specific calendar data criteria. Other controls in the CalendarSchedule allow you to edit priority, rename, or delete calendar date entries.

For more details, see “Calendar Scheduler view”.

schedule-CurrentDaySummary



The CurrentDaySummary view is available on BooleanSchedule, EnumSchedule, NumericSchedule, and StringSchedule components. It provides a simple linear listing of all schedule events for the current day, moving left-to-right from 0-to-24 hours.

For more details, see “Summary”.

schedule-Scheduler



The Scheduler allows you to view and edit schedule components, namely BooleanSchedule, EnumSchedule, NumericSchedule, and StringSchedule. Each of these components has a Scheduler view. The only difference between these different schedules is the output type.

NOTE: For more details, see “Weekly Scheduler view”.

There are four major elements in the Scheduler interface, separated by tabs in the view:

- Weekly Schedule
- Special Events
- Properties
- Summary

Weekly Schedule

Defines the regular (repeating) day-of-week schedule events for each day of the week.

Creating A Time Range Right-click and drag down in the slider column for desired the day.

Deleting A Time Range Click on a time range and press the delete or backspace key.

Editing A Time Range Time can be adjusted four ways: Dragging the top, bottom or center of a time range, or using the fine grained start and finish editors. To fine-tune a time range click on a time range then adjust the times in the editors to the left.

NOTE: IMPORTANT: The finish time is exclusive; it is the first non-effective time after the effective period. The start time is inclusive.

Output is assigned to each time range and can be edited just below the time editors on the left.

Special Events

Special events override (yet intermingle with) events in the normal weekly schedule.

Adding, Prioritizing, Renaming, and Deleting. Use the controls at the bottom for this purpose.

Editing At least one time range must be present for an exception to work. Output is assigned to the time ranges. The slider operates exactly like the normal weekly sliders.

Properties

Define the schedule's effective period, default output, facets, and whether automatic special-event cleanup occurs.

Default Output

This is the value of the schedule's output when no normal or exceptional schedule is effective.

Cleanup Special Events

If set to true (default), special events that have expired are automatically removed from the view. This occurs on the day following the scheduled special event.

Summary

Shows months with selectable days, including all schedule event activity for any day selected.

schedule-TriggerScheduler



The TriggerSchedule view is used to define schedule events in a TriggerSchedule. Trigger schedules are defined by combination of calendar day (or days) and trigger event time(s), including time ranges, each with a repeating interval.

Use the bottom-left **Add** button to add a scheduled day (or range of days). Each entry requires a unique name, a date type, and other specific calendar data criteria. Other controls in the TriggerSchedule allow you to edit priority, rename, or delete calendar day entries.

Use the bottom-right **Add** button to add trigger events to any selected scheduled day(s). Use the time selector to specify the individual trigger event. Use the Range controls to define a time range with repeating interval for trigger events to occur.

For more details, see "About trigger schedules".

Plugins in timesync module

- timesync-TimeSyncManager

timesync-TimeSyncManager



The TimeSyncManager is used to display the status of TimeSyncClients, as well as add new or edit existing TimeSyncClients. By default, it displays Server Name, Status, Poll Delta, Poll Local Time, and Poll Server Time for each TimeSyncClient.

NOTE: The **NtpPlatformService** has largely replaced this older RFC 868-based timesync module and components, since AX-3.4. For details, refer to "About the NtpPlatformService" in the *Platform Guide*.

Plugins in wiresheet module

- wiresheet-WireSheet

wiresheet-WireSheet



The WireSheet 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 popup menu.




WireSheet Menus

The WireSheet includes the following menus:

- WireSheet Main Menu
- WireSheet Component Menu
- WireSheet Background Menu
- WireSheet Link Menu


WireSheet Main Menu









The WireSheet main menu functions are available. When the *wiresheet; is visible, the following **WireSheet** menu functions are also available:

-  Delete Links
-  Arrange
-  Select All
- Show Thumbnail
- Show Grid
- Show Status Colors

WireSheet Component Menu









If you Right-click any Component in the WireSheet, 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.
-  Cut Ctrl + X

-  Copy Ctrl + C
-  Paste Ctrl + V
-  Duplicate Ctrl + D
-  Delete Delete
-  CompositeEditor
-  Rename
-  Reorder
-  Pin Slots









WireSheet Background Menu

If you Right-click the Background of the WireSheet, you can choose from the following:

-  Cut Ctrl + X
-  Copy Ctrl + C
-  Paste Ctrl + V
-  Duplicate Ctrl + D
-  Delete Delete
-  Delete Links
-  Arrange
- Select All
-  workbench-CompositeEditor

WireSheet Link Menu

If you Right-click any link in the WireSheet, you can choose from the following:

-  Cut Ctrl + X
-  Copy Ctrl + C
-  Paste Ctrl + V
-  Duplicate Ctrl + D
-  Delete Delete
-  Delete Links
-  Arrange
- Select All
-  CompositeEditor

WireSheet Toolbar

The Workbench toolbar contains navigation and editing buttons as described in “About the toolbar”. When the WireSheet is visible, additional toolbar buttons include:

-  Delete Links

Delete Links



Delete Links may be accessed from the Menu under **Edit** or from the toolbar. It allows you to eliminate the selected link(s). You can **Delete Links** only in the WireSheet.

Pin Slots



Pin Slots provides the capability to make Properties of a Component visible in the WireSheet view of the Component. Right-click on the Component and select **Pin Slots**. This will bring up the Pin Slots Dialog.

The selected Properties are now visible in the WireSheet.

Arrange



Arrange allows you to arrange the items in the WireSheet. You can Arrange All or Arrange Selection.

Select All

Select All allows you to select all items in the WireSheet.

Show Thumbnail

Show the thumbnail view in the upper right corner of the WireSheet. This allows you to toggle whether this function is enabled. This function may be accessed from the Menu under **WireSheet**. It allows you to display a thumbnail of the WireSheet view in the corner of the WireSheet. You can use the thumbnail to move around the WireSheet by Dragging the shaded area in the thumbnail. You can also hold down the Ctrl key and move the thumbnail around the WireSheet to keep it out of your way.

Show Grid

When enabled, the Grid is displayed in the background of the WireSheet. This helps you to align Components when you move them. You may control whether the Grid layer is enabled or visible from the WireSheet Menu or the **Tools→Options** Menu. This allows you to toggle whether this function is enabled.

Show Status Colors

Show the Status Colors. This allows you to toggle whether this function is enabled. This function may be accessed from the Menu under WireSheet. It allows you to display StatusColors in the WireSheet.

workbench-PrintDialog



Print provides the capability to print the WireSheet. The WireSheet will remain fixed at logical size of 100 x 100 "blocks". Printing will scale the WireSheet drawing to fit the page size (minus header and footer). Thus if you build your WireSheet logic in the top left corner, you will get a large scale for your print out. If you use the entire WireSheet down to the bottom right corner, you will get a much smaller scale for your print out. Margins are 1" for left and 1/2" for top, bottom, and right. A standard header is included which displays container's reference and current date. A future version will create a footer listing external links and knobs.

workbench-CompositeEditor



Composite Editor provides the capability to expose child slots of a Component as slots on the parent. Right-click on the Component and select **Composite Editor**. This will bring up the Composite Editor Dialog. The selected Properties are now visible in the WireSheet.

workbench-ExportDialog



Export provides the capability to Export views. You can Export any Table found in the workbench (anything built with `bajoui:Table`). The Table options menu (that little button to the right of the header), includes a Print and Export command. These commands allow you to Export the Table to PDF, HTML, or CSV (ExcelFile). The Export uses the current sort order and visible columns you have displayed.

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.

Figure 138 Category Browser

Category	Inherit	User	Admin	Operator	Viewer	Signal	Temp
Custom Config	n/a		•				
Services	✓		•				
Drivers	✓		•				
Apps		•	•				
category			•				
Ramp	✓		•			•	
Noise	✓		•				
SineWave	✓		•			•	
Threshold			•	•			
AddRamp	✓		•				
AddSine	✓		•				
GreaterThan	✓		•				
Temp1	✓		•				•
Temp2	✓		•				•
Alarm	✓		•				
AverageRoomTemp	✓		•				

There is no need to define component-to-category associations for Tagged Categories, so each Tagged Category column is grayed out and cannot be edited in the **Category Browser**. Any component with a tag that satisfies the NEQL query is visible in the **Category Browser**.

Columns

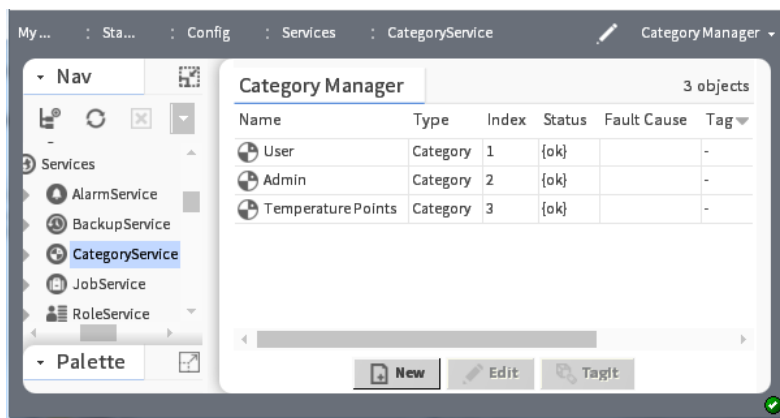
Column	Value	Description
Inherit	check mark or blank	A check mark indicates that the object inherits the category from its parent in the table.
User	Category 1	All system objects except for those listed as assigned to <code>Admin</code> are assigned to this category.
Admin	Category 2	These objects default to the <code>Admin</code> category: <ul style="list-style-type: none"> The configuration services: <code>UserService</code>, <code>CategoryService</code>, and <code>ProgramService</code> All files (the entire file space)
Categories 3–10	bold bullet, grayed out bullet, or blank	Objects with a bold bullet have tags that satisfy a tagged category. 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 139 Category Manager with tagged category, `Temperature Points` as `Category 3`



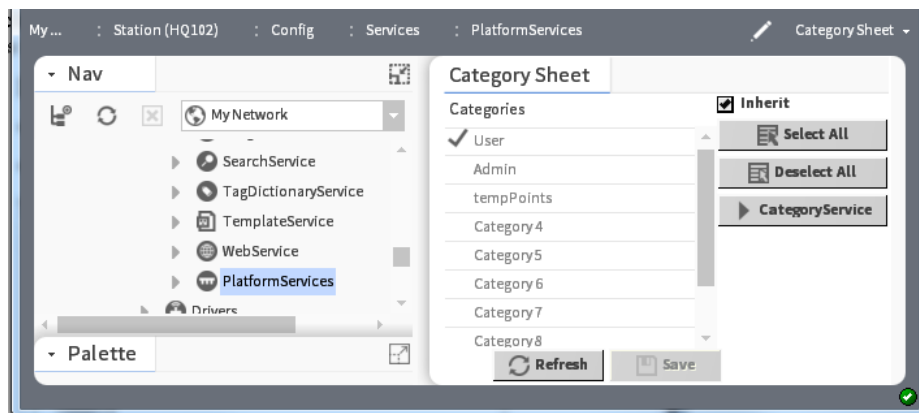
Column	Value	Description
Name	text string	Descriptive text that reflects the purpose of the entity or logical grouping.
Type	Category or Tagged Category	A basic Category is a name that is associated with individual objects. A Tagged Category includes a NEQL query that returns objects with tags that satisfy the query..

Column	Value	Description
Index	number	A unique number for the category, as it is known to the station.
Status [component]	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> • {ok} indicates that the component is polling successfully. • {down} indicates that polling is unsuccessful, perhaps because of an incorrect property. • {disabled} indicates that the Enable property is set to false. • fault indicates another problem.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Tag Query Name	text	A descriptive name to represent the results of the search.
Tag Query	NEQL	A NEQL query. This property is required when Type is Tagged Category.

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.

Figure 140 Category Sheet



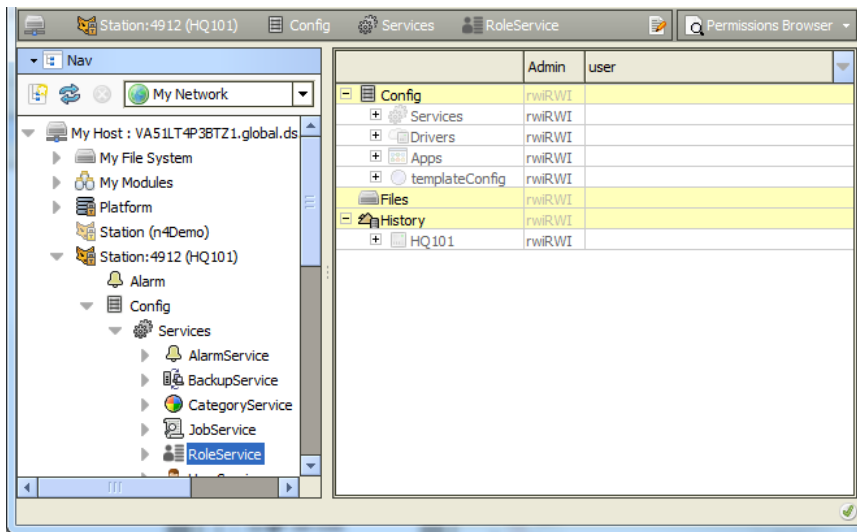
Option/button	Value	Description
Categories	text	Provides one table row for each category name.
Inherit	toggle	A check mark indicates that the component belongs to the same categories as its parent component. No check mark allows you to make explicit category assignments for this component.
Select All	button	Effective if Inherit is cleared, clicking this button assigns this component to all categories in this station.
Deselect All	button	Effective if Inherit is cleared, clicking this button removes this component from all categories.

Option/button	Value	Description
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**.

Figure 141 Permissions Browser view



Columns represent roles, 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 dialog for that role. This allows you to globally change a user's permission levels for any category in the station.

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 = writel = invoke actionadmin 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 actionoperator 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 UserManager is the primary view of the UserService. You use it to add, edit, and delete users for accessing the station.

Plugins in workbench module

- workbench-CollectionTable
- workbench-DirectoryList
- workbench-HexFileEditor
- workbench-JobServiceManager
- workbench-LinkSheet
- workbench-ModuleSpaceView
- workbench-NavContainerView
- workbench-NavFileEditor
- workbench-PropertySheet
- workbench-SlotSheet
- workbench-StationSummary

- workbench-TextFileEditor
- workbench-ServiceManager
- workbench-WbPxView
- workbench-WbServiceManagerView

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 co
```

The Table options menu (see “Table controls and options” for more details about the table options), allows you to Reset Column Widths, Print and Export.

workbench-DirectoryList










The **DirectoryList** Plugin provides a listing of the subdirectories and files found in a given Directory. If you Double-click an item, you will go to its default view.

Files will be displayed with an icon based on file type.

DirectoryList Menus

The Workbench main menu functions are available. When the DirectoryList is visible, the following popup menu functions are also available:

- Views - show the view menus for the Component.
- Actions - Perform any of the Actions on the Component.
- New - Create a new file of standard types (if a Directory).
-  Cut Ctrl + X
-  Copy Ctrl + C
-  Paste Ctrl + V
-  Duplicate Ctrl + D
-  Delete Delete
-  Rename
-  Reorder

DirectoryList 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.

New

New allows you to make a new Folder or File in the selected Folder of these types:

-  New Folder allows you to make a new Folder in the selected Folder.
-  BogFile.bog

-  HtmlFile.html
-  JavaFile.java
-  NavFile.nav
-  PrintFile.print
-  TextFile.txt

workbench-HexFileEditor



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

workbench-JobServiceManager



The **JobServiceManager** 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 popup **Job Log** dialog displays all the interim steps about 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. All jobs in a station are cleared upon a station restart.



workbench-LinkSheet



The **LinkSheet** allows you to view and delete links. It provides a view of the links on a **Component**.

LinkSheet Main Menu

The Workbench main menu functions are available. When the **LinkSheet** is visible, the following **LinkSheet** menu functions are also available:

-  Delete Links
-  GoTo

LinkSheet Toolbar

The Workbench toolbar contains navigation and editing buttons as described in "About the toolbar". When the **LinkSheet** is visible, additional toolbar buttons include:

-  Delete Links

GoTo



Go To goes to the selected item.

workbench-ModuleSpaceView



The **ModuleSpaceView** allows you to view **Modules**.

OptionsButton

The Options Button provides the ability to toggle columns on and off like Mozilla does.

workbench-NavContainerView



NavContainerView is a default listing of nav children.

workbench-NavFileEditor



The NavFileEditor 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.

In order to use the Nav File, you must place the 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. In order to see properties of components in a **Property-Sheet**, 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.

Display Name	Value	Commands
▶ MainKw	873.9 kW {ok} @ 10	[Edit] [Action]
▶ HistoryStartDate	Program	[Action]
▶ MainKwh	162.5 kW-hr {ok} @ 10	[Edit] [Action]
▼ Energy	Px View	
Icon		
Required Permissions	r	
Media	workbench	
Px File	file:^px/Energy.px	
b:energy	M	
b:equip	M	
icon		
LastWeekMainKw	Transform Graph	

- **AX Property Sheet view**

The default property sheet view for the NiagaraAX and 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 press  **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	Invokes the Edit Tags dialog, 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 dialog box.
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 dialog box. 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).

Item	Description
Reorder	Displays the Reorder Points dialog box, which provides the following commands for reordering points within the selected parent component. <ul style="list-style-type: none"> • Move Up • Move Down • Sort by Name • Sort by Type • Reset
Composite	This menu item displays the Composite Editor dialog box.
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 dialog.

PropertySheet Toolbar

The Workbench toolbar contains navigation and editing buttons as described in "About the toolbar".

Config Flags



Configure Flags on a component. This is available on Properties. Right-click a Property to view the Config Flags Dialog.

Links



Links provide the mechanism to dynamically attach Components. They allow you to attach an output Property of one Component to an input Property of another Component. The links are visible in the Property Sheet of the Component. Links show whether they are direct or indirect and their source Property.

In order to view the Properties of the link, Left-click the + plus sign on the Component to expand it.

Each link shows the following:

- Link Type and source
- Source Ref - This is the linked Component.
- Source Slot Name - This is the linked Component's property.
- Target Slot Name - This is the current Component's property.
- Enabled - This shows whether the link is currently enabled.

See for more information on creating links.

OrdChooser

In order to select an Ord instead of typing it, you can use one of the Ord editors. They include:

- Bql Builder
- Directory Ord Chooser
- File Ord Chooser

- History Ord Chooser
- Ref Chooser

Once you have selected an editor, you can use the » button to the right of the selection box. It will present the selected editor.

workbench:FileOrdChooser

In order to select a file Ord instead of typing it, you can use the FileOrdChooser.

You can select FileSpaces, Bookmarks and directories and files. In addition toolbar functions are provided including:

Back



Back moves back to the previous view.

Uplevel



Uplevel moves up one level in tree.

Home



Home moves up one level in tree.

New Folder



New Folder creates a new Directory.

ListView

ListView allows you to use the list view.

DetailsView

DetailsView allows you to use the details view.

Bookmarks

You can press the  **Bookmarks** button to add the selection to Bookmarks.

Show/Hide Preview



Show/Hide Preview toggles whether the preview pane is visible.

workbench:DirectoryOrdChooser

In order to select a Directory Ord instead of typing it, you can use the DirectoryOrdChooser.

You can select FileSpaces, Bookmarks and directories. In addition toolbar functions are provided including:

New Folder



New Folder creates a new Directory.

Bookmarks

You can press the **Bookmarks** button to add the selection to Bookmarks.

workbench:HistoryOrdChooser

In order to select a reference Ord instead of typing it, you can use the HistoryOrdChooser.

You can select items from the tree by type slot or Handle.

workbench:RefChooser

In order to select a reference Ord instead of typing it, you can use the RefChooser.

You can select items from the tree by type slot or Handle.

workbench:BqlQueryBuilder

In order to select a BQL query instead of typing it, you can use the BqlQueryBuilder.

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

In order to build the projection for a BQL request instead of typing it, you can use the ProjectionBuilder in Bql Builder. You can select an item from the center window to use with the select statement. Press the \Rightarrow right arrow to add each one to the projection.

Bql Expressions "

An expression can be one of the following:

- field like displayName
- x.y.z
- any method that returns non-void and takes zero parameters

After you build the BQL statement, you can remove the "Select" to provide a BQL expression instead of a Select statement.

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

In order to build the Qualifier for a BQL request, you can type it in the QualifierBuilder in Bql Builder. As you type, it will immediately appear in the BQL statement at the top.

Edit Facets

Edit Facets allows viewing and editing of facets. In order to change facets, you use the » button to the right of the facets. It will present the Edit Facets dialog box.

From here you can Add, Remove or select the Enum Range Dialog.

Add

Add allows you to add a Key and Type.

Remove

Remove allows you to remove facets.

Enum Range Dialog

In order to view or set the range of values for a Enum, you can use the ». It will present the Enum Range Dialog:

You can enter one of the standard Enumerations in the field under the **Use Enum Type in Range (module: name)**.

Press **Use Enum Type in Range (module:name)** to use the Enumeration that you entered. In order 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 press **Modify**. Press **OK** to complete the dialog.

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 SlotSheet is a Table that shows the following for each slot:

- Slot






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

The SlotSheet also optionally shows any Name Maps. The SlotSheet includes the following menus:

- SlotSheet Main Menu
- SlotSheet Component Menu







SlotSheet Main Menu

The Workbench main menu functions are available. When the SlotSheet is visible, the following **SlotSheet** menu functions are also available:

-  Add Slot Ctrl + A
-  Rename Slot Ctrl + R
-  Config Flags
-  Config Facets
-  Reorder
- Add Name Map





SlotSheet Component Menu

If you Right-click any Component in the SlotSheet 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 as described in "About the toolbar". When the SlotSheet 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 Dialog.

workbench-StationSummary




StationSummary is the default view on a Station. It holds primary components (e.g. 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



(AX-3.7 and later) The Synthetic Module File View is the default view on a  synthetic module. Refer to the *Engineering Notes* for details.

workbench-TextFileEditor



The TextFileEditor Plugin provides a powerful Color coded text editor. It supports Color coding of C, java and xml file types. See File Types for more information on Color coding of specific file types. See Text Editor Options to change editor options including Color coding.

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, refer to the *NiagaraAX Graphics Guide* sections "About Px views" and "About the Px Editor menu". Other related Px information is available in that document as follows:

- "About Px files"
- "About Px viewer"
- "About Px Editor"

WbPxView Menus

The Workbench main menu functions are available. When the **Px Viewer** is visible, the following menu functions are also available:

- PxViewer View Source Xml

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.

A References

Topics covered in this appendix

- ◆ 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 appendix contains reference topics about keyboard shortcuts, menu bar items, popup 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. Refer to “About the menu bar” for an overview of the menu bar.
- Types of popup menu items
Describes the popup menus that appear in different views and contexts.
- Types of toolbar icons
Describes the types of icons that are in the toolbar. For an overview of the toolbar, refer to “About 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:

- F1 - Help On View
- F3 - Console
- F4 - Hide Console
- F5 - Find
- F6 - searchReplace;
- F7 - Goto File
- F8 - searchConsoleNext
- F9 - Save amp; Compile
- Alt + Left - Back
- Alt + Right - Forward
- Alt + Up - Up Level
- Alt + Home - Home
- Alt + Space - Recent Ords

- Ctrl + A - Add Slot
- Ctrl + C - Copy
- Ctrl + D - Duplicate
- Ctrl + F - Find Next
- Ctrl + G - Goto Line
- Ctrl + L - Open Ord
- Ctrl + N - New Window
- Ctrl + O - Open File...
- Ctrl + P - Print
- Ctrl + R - Rename Slot
- Ctrl + S - Save
- Ctrl + T - New Tab
- Ctrl + V - Paste
- Ctrl + W - Word Wrap
- Ctrl + X - Cut;
- Ctrl + Z - Undo
- Ctrl + F1 - Find Bajadoc
- Ctrl + F4 - Active Plugin
- Ctrl + F5 - Find Files
- Ctrl + F6 - Replace Files
- Ctrl + F9 - Compile
- Ctrl + PageUp - Next Tab
- Ctrl + PageDown - Previous Tab
- Shift + F8 - searchConsolePrev
- Ctrl + Shift + F - Find Prev
- Ctrl + Alt + Z - Redo

Types of menu bar items

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


- File menu
See "About the File menu"
- Edit menu
See "About the Edit menu"
- Search menu
See "About the Search menu"
- Bookmarks menu
See "About the Bookmarks menu"
- Tools menu

- See "About the Tools menu"
- Window menu
 - See "About the Window menu"
- Px Editor menu
 - See "About the Px Editor menu"
- History Ext Manager menu
 - See "About the History Ext Manager menu"
- Help menu
 - See "About the Help menu"


About the File menu


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


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

-  Open Ord

You may open a ORD by selecting **File > OpenOpen Ord** from the main menu or from the toolbar open button.

The Ord Chooser also allows you to directly type the Ord of a location in order 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.
-  Open File

You may open a file by selecting **> File→Open→Open File...** from the main menu.
-  Open Directory

You may open a Directory by selecting **> File→Open→ Open Directory...** from the main menu.
-  Open Station (Fox)

You may open a Station by selecting **> File→Open→ Open Station (Fox)...** from the main menu. This will cause the following popup window to be presented so that you can complete each field.


Address - This is the address of the computer running the Station that you wish to access.

User name - This is the user name given to you by your system administrator.

Password - This is the password given to you by your system administrator.

Remember these credentials - This will save this connection in your connection list.

Once you successfully connect to the Station, it will appear in the tree and your home page will be displayed.

If you will be away from the system, you should close the Station to prevent unauthorized access.
-  Open Platform

You may open a Platform by selecting **File→Open→ Open Platform...** from the main menu. This will cause the following popup window to be presented so that you can complete each field.


Host - This is the address of the computer running the Platform that you wish to access.

Port - This is the port used.


Password - This is the password given to you by your system administrator.

Remember these credentials - This will save this connection in your connection list.

Once you successfully connect to the Platform, it will appear in the tree and the available tools will be displayed.

-  Find Stations

You may find Stations by selecting **File→Open→ Find Stations...** from the main menu. This command finds all the stations running on the network. It will search for 5 seconds, then display a Table of all the stations found. You may display additional columns about each station using the options button (on the right of the header). Double-click a Station to open a Fox connection to it.

-  Back

Back allows you to go to the previous view. The shortcut is Alt + Left.

-  Forward

Forward allows you to go to the next view. You must have used the Back command previously. The shortcut is Alt + Right.

-  Up Level

Up Level allows you to go to the next level up. The shortcut is Alt + Up.

-  Save

Saves the value of the Component.

-  Save Bog

Save the Component changes to the Bog File file.

-  Save All

Save all open views. This saves all Tabs when browsing with multiple Tabs.

-  Recent Ords

Recent Ords allows you to see recent Ords. The shortcut is Alt + Space.

-  Refresh

Refresh allows you to refresh the current view.

-  Home

Home allows you to go to the home view. The shortcut is Alt + Home.

-  Printing

Printing provides the capability to print the current view. When it appears dimmed, printing is not available.

-  Export

Export provides the capability to Export the current view. When it appears dimmed, Export is not available.

- Logging off the System

You may exit the system by a number of means. You may **Close** the session, **Close All** sessions, **Exit**, or **Close** the current window. Each of these causes different actions and should be understood.

- Close Session

You may logoff the Station at any time by selecting **File→Close** from the main menu.

Once you successfully close a session, it will be removed from the tree. This could leave a user interface running without an open session.

- Disconnect


You may logoff the system at any time by Right-clicking the connected system in the tree and select **Disconnect**. This allows you to close a session with a Station without removing it from the tree.


- **Dismiss**


You may logoff a connected system at any time by Right-clicking the connected system in the tree and select **Dismiss**. This allows you to close a session with a Station and remove it from the tree.
- **Exiting the System**

You may 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.
- **Close the Current Window**

You may close the current window at any time, if it is not the primary window, by any of the following methods:

 - selecting **File→Close** from the main menu
 - pressing the box in the top left corner and selecting **Close** from the menu
 - pressing the  close icon in the top right corner of the window
- **New Window**

New Window requests that a new window be created identical to the current one. This allows you to view multiple views concurrently.
-  **New Tab**

New Tab requests that a new tab be created identical to the current one. This allows you to view multiple views in the same Window. You can hold down the Ctrl key during a Double-click to hyperlink into a new tab. You can also Right-click to get a popup menu on tabs to close the tab, or close all other tabs.
-  **Close Tab**


Close Tab requests that a tab be closed.
- **Close Other Tabs**

Close Other Tabs requests that all other tabs be closed.
- **Next Tab**


Next Tab selects the next tab. The shortcut is Ctrl + PageUp.
- **Previous Tab**


Previous Tab selects the previous tab. The shortcut is Ctrl + PageDown.

About the Edit menu


-  **Cut**

Cut copies the current Selection to the clipboard and changes its Color to gray until a Paste is performed to delete it. If the Selection is a string, Cut makes a Copy of the current text selection and places it in the clipboard and removes it from the current string. It may be accessed by selecting an item and:

 - Right-clicking the item(s) and choosing Cut.
 - Right-click in the tree and choosing Cut.
 - In the Wire Sheet press the  Cut toolbar icon.
 - In the Wire Sheet select **Edit→Cut** from the main menu.
 - pressing the shortcut key Ctrl + X (hold down Ctrl and press X) when in a window other than the main browser window.

You can use Drag to Cut and Paste in one operation.
-  **Copy**

Copy copies the current contents of the clipboard to the destination component as a set of new dynamic properties. If the Selection is a string, Copy makes a copy of the current text Selection and places it in the clipboard. To Copy a Component, do any of the following:

- Right-click a Component in a view or tree and choose Copy
- Right-click the background in the Wire Sheet choosing Copy
- In the Wire Sheet press the  Copy toolbar button.
- In the Wire Sheet select **Edit**→**Copy** from the main menu
- pressing the shortcut key Ctrl + C (hold down Ctrl and press C) when in a window other than the main browser window.

You can copy a selected group of Components in the Wire Sheet.


You can use Drag to Copy also.

In order to Copy a Component, you must Copy it from the Palette or an existing Database. You can follow one the following steps:

- In the Palette Sidebar, expand a module (like control) and sub-directory (like Points), Right-click on a Component (like BooleanWritable) and select Copy.
- In the namespaces sidebar (tree), expand System, Modules, a module (like control), Files, module.pallete, and sub-directory (like Points), Right-click on a Component (like BooleanWritable) and select Copy.
- OR Right-click on a Component that you want to Copy from a Wire Sheet, Property Sheet or the tree and select Copy.

-  Paste

Paste copies the contents of the clipboard to the destination Component as a set of new dynamic properties. If the target is a string, Paste places a reference to the source that was cut or copied into clipboard. It may be accessed by cutting or copying item(s) and:

- Right-clicking the item and choosing Paste
- Right-click in the tree and choosing Paste
- In the Wire Sheet press the  Paste toolbar icon.
- In the Wire Sheet select **Edit**→**Paste** from the main menu
- pressing the shortcut key Ctrl + V (hold down Ctrl and press V) when in a window other than the main browser window.

You can use Drag to Cut and Paste in one operation.






You can add a Component to a running Station or an offline Bog File file. You can do this using any of the following:

- In the Wire Sheet, Right-click on the background and select Paste to add the new Component. The new Component is created and is selected.
- In the tree, Right-click on a Container and select Paste to add the new Component inside the Container.

See the Bajadoc Index for reference information. 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 `module://control/doc/javax/baja/control/BBooleanWritable.bajadoc`.







-  Paste Special

Paste Special copies the contents of the clipboard to the destination Component when it is a Special Transferable.




-  Duplicate
Duplicate copies the current Selection and places a duplicate in the same Container. This function may be accessed from the menu under **Edit** (shortcut Ctrl + D) or from the toolbar.
-  Delete
Delete removes the current Selection from its parent Container. It may be accessed by selecting an item and:
 - Right-clicking the item(s) and choosing **Delete**
 - Right-click in the tree and choosing **Delete**
 - In the Wire Sheet press the  elete toolbar button.
 - In the Wire Sheet select **Edit**→**Delete** from the main menu
-  Undo
This reverses the last Action as if it had not been performed. It is only available for certain actions such as:
 - Paste Component
 - Cut Component
 - link
 - Delete Links
-  Redo
This 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:

-  Find
Find allows you to search in the file for the selected string. You can **Match Case** or **Match Word**. The shortcut is F5.
-  Find Next
Find Next allows you to find the next occurrence of the selected string. The shortcut is Ctrl + F (hold down Ctrl and press F).
-  Find Prev
Find Prev allows you to find the previous occurrence of the selected string. The shortcut is Ctrl + Shift + F (hold down Ctrl and Shift and press F).
-  Replace
Replace allows you to replace the next occurrence in the file. The shortcut is Ctrl + R (hold down Ctrl and press R).
- Goto Line
Goto Line allows you to go to a line number in the file. The shortcut is Ctrl + G (hold down Ctrl and press G).
-  Goto File
Goto File allows you to go to a file. The shortcut is Ctrl + F3 (hold down Ctrl and press F3).
-  Find Files

Find In Files allows you to find the all occurrences of a string in files. You can **Match Case** or **Match Word**. You can choose **Files to Find**. You can select a **Folder**. You can choose whether to **Search subfolders**.

-  Replace Files
Replace Files allows you to replace the all occurrences in the files.
-  Console Prev
Console Prev allows you to go to the previous console error. The shortcut is F7.
-  Console Next
Console Next allows you to go to the next console error. The shortcut is F8.



About the Bookmarks menu

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

- AddTo 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:

- Viewing and Changing the Options
The **Options** allow you to customize the framework for the way you use it. It can be selected from the main Menu by selecting **Tools→ Options**. It includes the following:
 - General
 - Lexicon
 - Text Editor
 - Wire sheet
- ColorChooser
The ColorChooser allows you to choose Colors.
- New Module Wizard
 The **New Module** Wizard allows you to build a new Module. It can be selected from the main Menu by selecting **Tools→ New Module**.
- New Station Wizard
- 
The **New Station** Wizard allows you to build a new station Database. It can be selected from the main Menu by selecting **Tools→ New Station**.
You must enter a **Station Name**. You can also choose whether this is a **JACE Station** or a **Supervisor Station**. Press **Next** to proceed. Next you should enter an **Admin Password**. You can also change the

Fox Port and **HTTP Port**. Press ✓ **Finish** to proceed. You are then presented a view of your newly created Station at local:file:!stations/stationname/config.boglbog:slot:/.

- Manage Credentials



Manage Credentials is available in the main **Tools** menu by selecting **Manage Credentials....** You can **Reset** or **Remove** selected Credentials or **Remove All** Credentials. You can also **Open** selected Credentials.

- Request License

Request License is available in the main **Tools** menu by selecting **Request License....** You can request a license by submitting the form.

About the Window menu

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

- 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.

For more information about the side bar refer to “About the side bar panes”.

- Bookmarks

You can choose to show the Bookmarks Side Bar by selecting **Window→ Side Bars→Bookmarks** from the main menu.

For more information about the Bookmarks side bar refer to “About the bookmark side bar”.

- ? Help Sidebar

The typical configuration provides a Help Sidebar frame in the left side of the main window. If it is not open, you can choose to display the Help Side Bar by selecting **Window→ Side Bars→ Help** from the main menu.

Initially, the Help Sidebar is empty until you press **Load Help**.



Load Help will reappear any time any Module's timestamp is changed or a new Module is added or removed. **Load Help** searches through all the available Modules to create the help Directory. The Help Sidebar provides a view of the available Help.

For more information about the Help Sidebar refer to “About the help side bar”.

-  Jobs Sidebar

The Jobs SideBar shows all the current jobs in all the stations with which you have a connection. Controls are provided to allow you to view a job log or dismiss a completed job.

For more information about the Jobs side bar refer to “About the jobs side bar”.

- Nav

You can choose to show the Nav side bar by selecting **Window→ Side Bars→ Nav** from the main menu.

For more information about the Nav side bar refer to “About the nav side bar”.

- Palette

You can choose to show the Palette Side Bar by selecting **Window→ Side Bars→ Palette** from the main menu.

For more information about the Palette side bar refer to “About the palette side bar”.

- PathBar Uses NavFile

You can toggle this option ON or OFF by selecting **Window**→ **Side Bars**→ **PathBar Uses NavFile**. When ON, the PathBar (located at the top of the Workbench main window) displays your current path, as defined by the NavFile (logical path). When OFF (not selected) the PathBar displays your absolute path regardless of whether it is mapped to the NavFile or not. You must refresh the view after changing this setting in order 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.

Refer to “Types of console commands”⁶ for information about console commands.

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.
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.

Item	Description
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 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"> • Workbench: WbPx Media • hx: HxPxMedia • report: ReportPxMedia • mobile: MobilePx Media <p>See “About presentation media” for more details about media types.</p>

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:

- ▶ 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 dialog box.
- 🏷 Edit System Tags
Select this menu item 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 Help menu

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

- ? Help Contents Index
The help contents provides a common point of access to all system documentation. It is accessed by selecting **Help→ Contents Index** from the menu or pressing the ? Help button on the toolbar to see the Help Index.
- ◆ Help On View
This provides help for the current Plugin. It is accessed by selecting **Help→ On View** from the menu with the Plugin in use.
- ● Help Guide On Target
This provides context sensitive help for Components. It is accessed by selecting **Help→ Guide On Target** from the menu when the current view is a view of a Component. It is also available by Right-clicking a Component and choosing **Views→ Guide Help**.
- ● Help Bajadoc On Target

This provides context sensitive Bajadoc help for Components. It is accessed by selecting **Help→Bajadoc On Target** from the menu when the current view is a view of a Component. It is also available by Right-clicking a Component and choosing **Views→ Bajadoc Help**.

-  Help Find Bajadoc

This is accessed by selecting **Help→ Find Bajadoc** from the menu. It searches for the requested bajadoc.

-  Help About

This is accessed by selecting **Help→ About** from the menu. It provides the software release and license information.

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

For descriptions of the popup menu items that are available in the nav side bar refer to “About the nav side bar popup menu items”.

- Wire sheet

For descriptions of the wire sheet popup menu items, refer to “About the wire sheet popup menu items”.

- Property sheet

For descriptions of the property sheet popup menu items, refer to “About the property sheet popup menu items”..

- Px Editor

For descriptions of the Px Editor popup menu items, refer to “About the Px Editor popup menu items”..

- History extension manager

For descriptions of the History extension manager popup menu items, refer to “About the history extension manager popup menu items”..

- Todo list

For descriptions of the Todo list popup menu items, refer to “About the Todo list popup menu items” ..

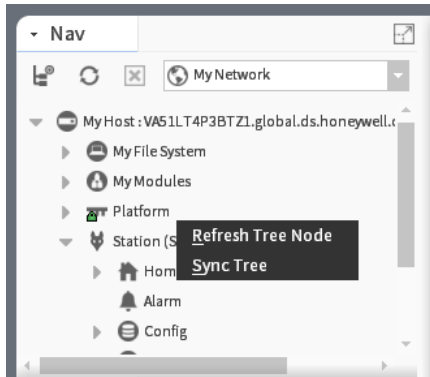
- Point Manager

For descriptions of the Point Manager popup menu items, refer to “About the point manager popup menu items”.

About the Nav side bar popup menu items

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

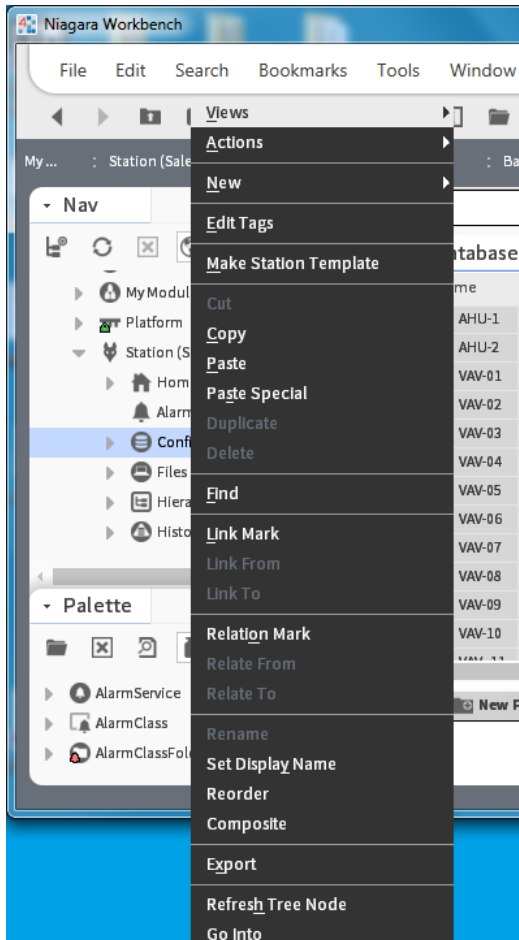
Figure 142 Nav side bar popup menu with nothing selected



This popup menu has these options:

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

Figure 143 Nav side bar popup menu with component selected



The popup menu has the following options:

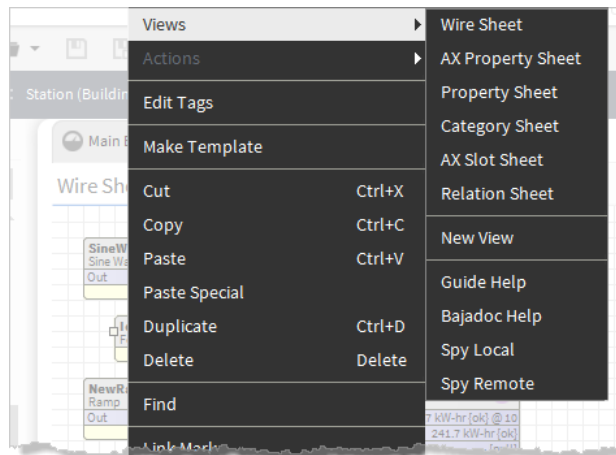
Item	Description
Views	Goes to any of the views of the selected component.
Actions	Perform any available actions on the component.
New	Provides additional options for creating new objects.
Edit Tags	Invokes the Edit Tags dialog, permitting you to add or remove tags on a component.
Make Template	Creates 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	
Duplicate	Makes a copy in the same location as the original item
Delete	Removes the item
Find	
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 dialog box. 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	Displays the Reorder Points dialog box, which provides the following commands for reordering points within the selected parent component. <ul style="list-style-type: none"> • Move Up • Move Down • Sort by Name • Sort by Type

Item	Description
	<ul style="list-style-type: none"> Reset
Composite	This menu item displays the Composite Editor dialog box.
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	Invokes the Pin Slots dialog. 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 popup menu items

Most of the Wire Sheet menu commands are described in "About the nav side bar popup menu items".

Figure 144 Wire sheet popup menu



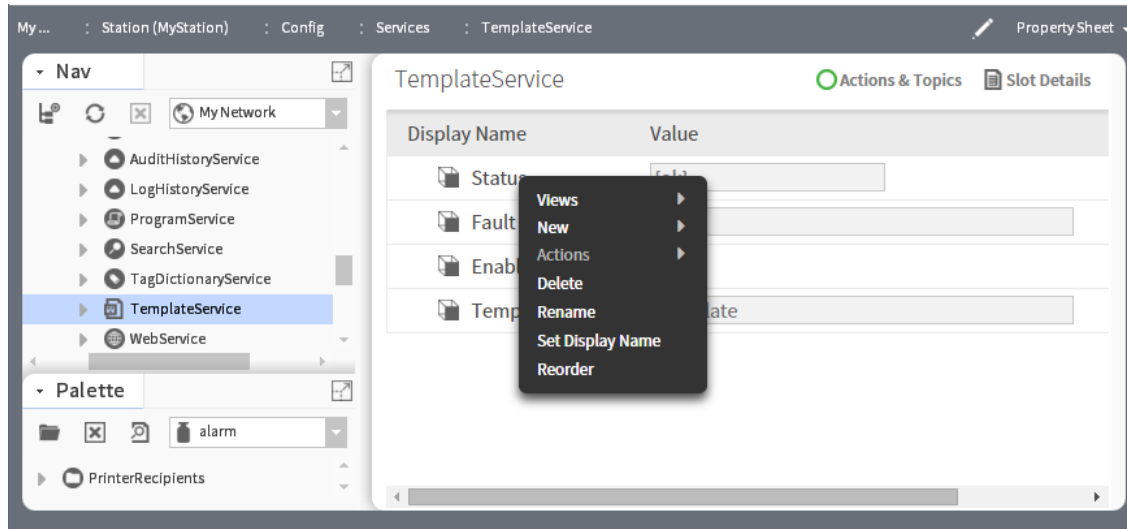
Following are wire sheet specific popup menu options:

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

About the property sheet popup menu items

Most of the property sheet menu commands are described in “About the nav side bar popup menu items”.

Figure 145 Property sheet popup menu



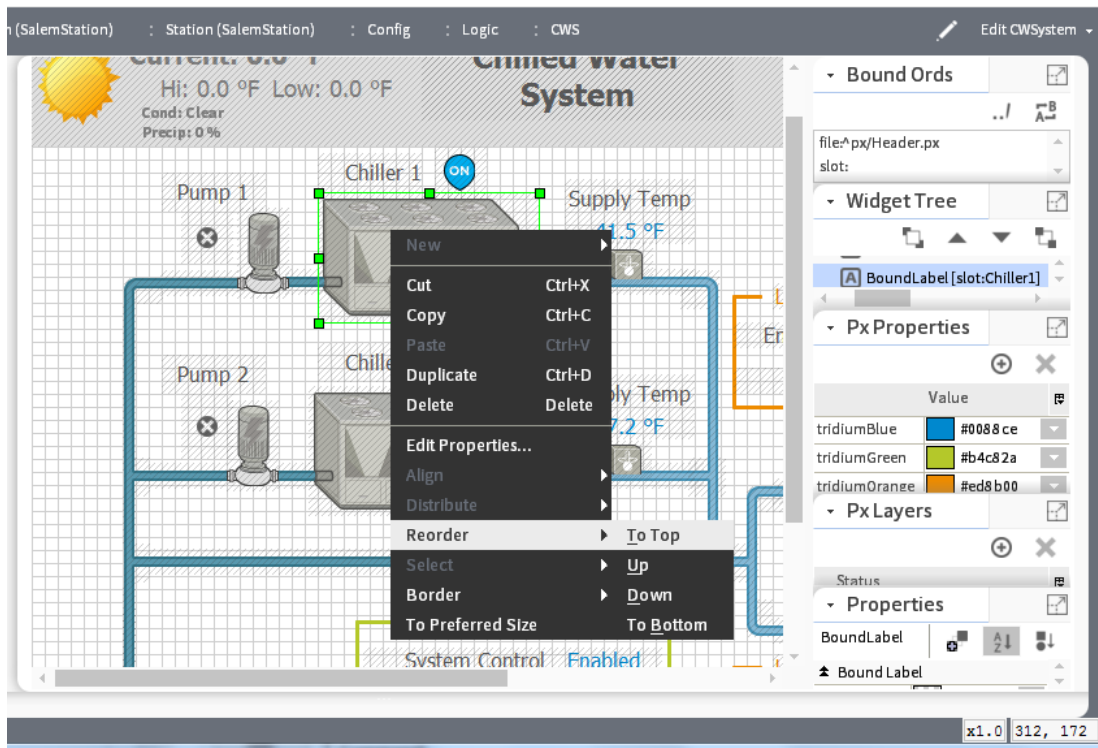
Item	Description
Views	
New	
Actions	
Delete	
Rename	
Set Display Name	
Reorder	

- Config flags
Opens the Config dialog box 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 146 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 "About the properties side bar")
Align	<ul style="list-style-type: none"> Left This command is available when you select two or more objects in the Px Editor. Choose the Left command to align the left edges of two or more objects along a vertical line. Right This command is available when you select two or more objects in the Px Editor. Choose the Right command to align the right edges of two or more objects along a vertical line. Top


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

- **Views**
This menu item 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. See “About history names” for more information about history naming.

- **Go To Point**
This menu item displays the property sheet view of the point associated with the selected entry.
- **Go To History**
This menu item 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 dialog box. 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** 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.

Refer to “About the History Ext Manager menu” for information about the History Ext Manager menu. Refer to “About the history extension manager view” for general information about the history extension manager view.

About the Todo list popup menu items

The Todo list popup menu has the following items:

- ** Add**
This menu item opens the **Add** dialog box, 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** 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.

-  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.

Refer to “Todo List” for general information about the Todo list view.

About the point manager popup menu items

The point manager popup menu has the following items:

- 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 (see “About the view selector”) 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 points manager. Valid options are presented in the submenu.

- Cut

See “Types of edit commands”.

- Copy

See “Types of edit commands”.

- Paste

See “Types of edit commands”.

- Duplicate

See “Types of edit commands”.

- Delete

See “Types of edit commands”.

- Find

This menu item displays the **Component Finder** dialog box.

- Link Mark

This menu item sets a selected point to your popup 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 dialog box. You can only rename one point at a time.

- **Reorder**

This menu item displays the **Reorder Points** dialog box, which provides the following commands for re-ordering points within the selected parent component.

- Move Up
- Move Down
- Sort by Name
- Sort by Type
- Reset

- **Composite**

This menu item displays the **Composite Editor** dialog box.

- **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.

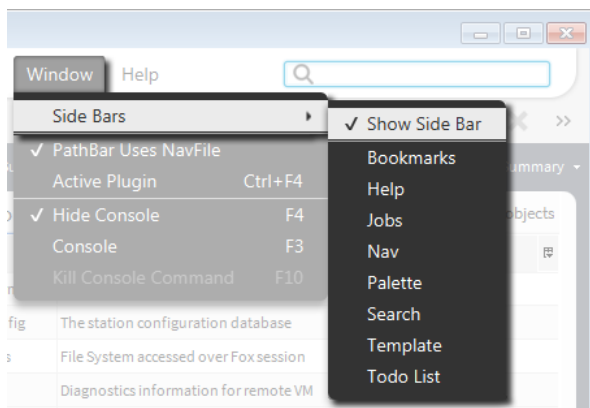
- **Edit**

This menu item displays the **Point Editor** dialog box.

Types of side bars

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

Figure 147 Default Side Bars menu



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

- **Bookmarks** side bar

Displays a list of your bookmarks. For details, see "About the bookmark side bar".

- **Help** side bar

Provides a tree view of available help documentation. For details, see “About the help side bar”.

- **Jobs** side bar

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. For details, see “About Jobs side bar”.

- **Nav** side bar

Provides a tree view of the system. For details, see “About the Nav side bar”.

- **Palette** side bar

Provides a tree view of components that are available in specific palettes. For details, see “About the palette side bar” and “About the palette side bar toolbar”.

- **Search** side bar

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** side bar

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. For details, see “About the Template side bar”.

- **Todo List** side bar

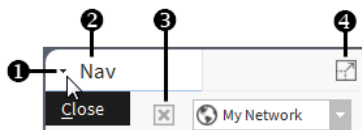
Provides a customizable list of tasks or notes. For details, see “About the Todo list sidebar”.

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 148 Side bar title bar



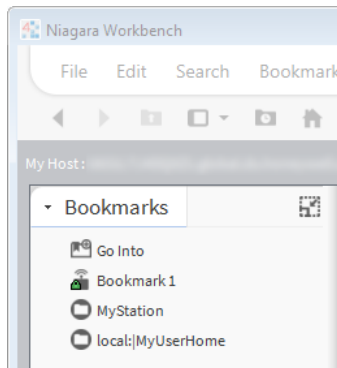
- 1 Close dropdown — drop down control used to close the side bar
- 2 Pane title— text that identifies the side bar type
- 3 Close — alternative to the Close dropdown, closes the side bar
- 4 Expand/Restore — clicking on the expand/restore icon causes the side bar to expand, filling the side bar pane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

To find out more about specific side bar panes and their controls see “Types of side bars”.

About the Bookmarks side bar

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

Figure 149 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.

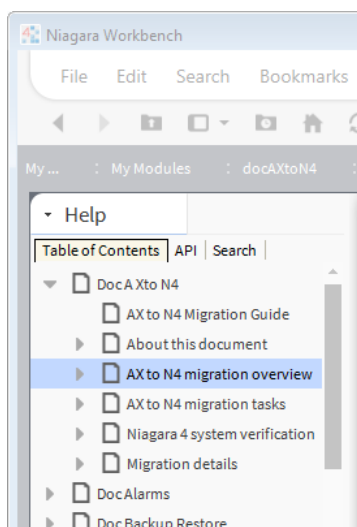
For more details about the bookmark side bar, refer to the following:

- To find out how to add or remove bookmarks in the bookmark side bar, see “To add a bookmark”.
- For details about managing bookmarks, see “To manage bookmarks”
- For details about editing bookmarks, see “To edit a bookmark”

About the Help side bar

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

Figure 150 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:

- **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.

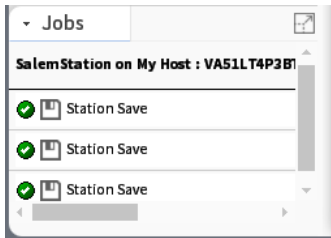
- **Search tab**

Contains a **Find**: text entry field and **Search** button. For details on using the Search, see “Using the Help side bar”.





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 151 Jobs side bar

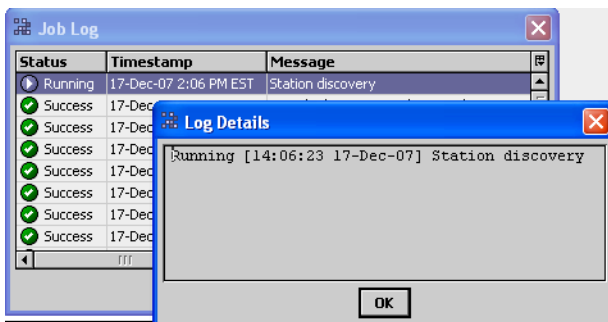


The following icons on the Jobs side bar indicate job status:

-  **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** dialog box. The **Job Log** dialog box 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** dialog boxes shown below.

Figure 152 Log Details dialog box

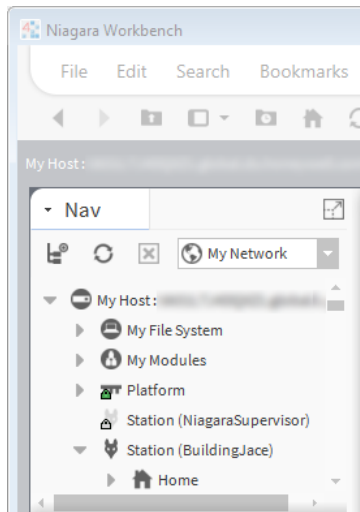


About the Nav side bar

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 153 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. Refer to “Types of nodes in the Nav side bar tree” for more details about file types. Refer to “About the Nav side bar popup menu items” for more details about the Nav side bar popup menu.

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 inWorkbench. 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.

- **My Host** node
Represents a physical computer (hardware) that the rest of the nodes (subnodes) reside on. For more details about how the host fits into the Niagara architecture, see “About Niagara software architecture”.
- **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. For more details on modules, see “About modules”.
- **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 “About a platform connection” in the *Platform Guide* for more details.

- **Station node**

Represents a station (connected or disconnected). When expanded, the station node displays the station contents in a hierarchical tree. You can double-click on the station node and sub-nodes, or use a right-click shortcut menu to perform all operations that are available on or under this node (connecting, disconnecting, selecting views, and more).

- **Home**
- **Alarm**
- **Config node**

When expanded, displays a tree view of the station contents or “configuration”. The config node usually contains one or more of the following types of nodes:

- ◆ **Services**

Component for storing services, such as alarm service, history service, tagdictionary service, and more.

- ◆ **Drivers**

Provides a place to store driver modules (such as the NiagaraNetwork, BACnet drivers, Modbus, and more). When expanded, displays a tree view of loaded driver modules. For more details, refer to the “About Network architecture” section in the *Drivers Guide*.

- ◆ **Apps**

- ◆ **Schedules**

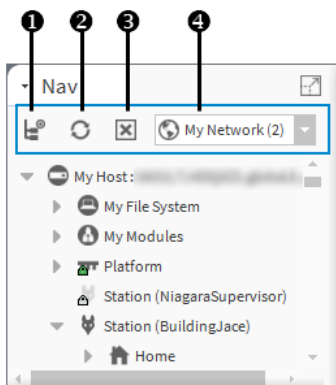
- ◆ **Control or Logic**

Control points may be displayed directly in the root of the Config node.

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 154 Nav side bar tool bar



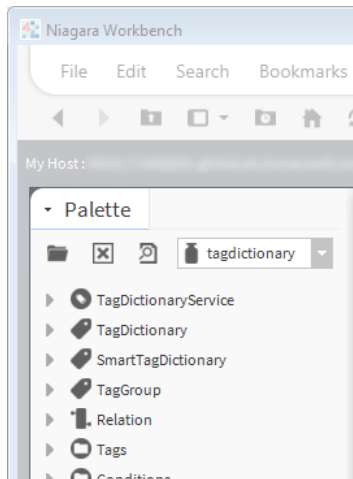
- 1 **New tree** button — creates a new tree in the nav side bar. When you have more than one tree node, you can select one to activate from the dropdown tree selector.
- 2 **Sync tree** button — synchronizes the tree node display with the currently selected view.

- 3 **Close tree** button — closes the currently displayed side bar.
- 4 **Drop-down tree** selector — when multiple Nav side bars are open, this selector allows you to choose which one to display.

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.

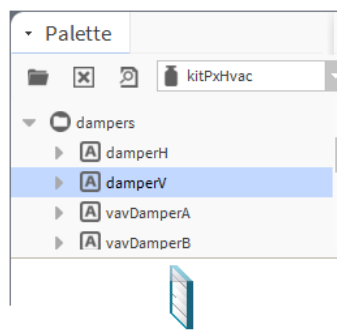
Figure 155 Palette side bar with preview pane



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 156 Palette preview pane



Palette previews display in the palette when components have images configured either as the default image property or as the image assigned to the `comPreviewWidget` property. If no preview is associated with a component, you can add a `comPreviewWidget` property to a widget in order to display an image in the

preview pane of the palette side bar. See “To add a side bar preview using the compPreviewWidget property”.

For more details about the Palette side bar, refer to the following:

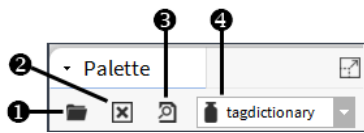
- To find out how to use the Palette side bar, see “Using the Palette side bar”.

For details about adding palettes to the Palette side bar, see “To open a palette”

About the Palette side bar toolbar

In addition to the standard side bar title bar (see “About the side bar title bar” for more details) the **Palette** side bar has a toolbar, located just below the title bar.

Figure 157 Palette side bar toolbar



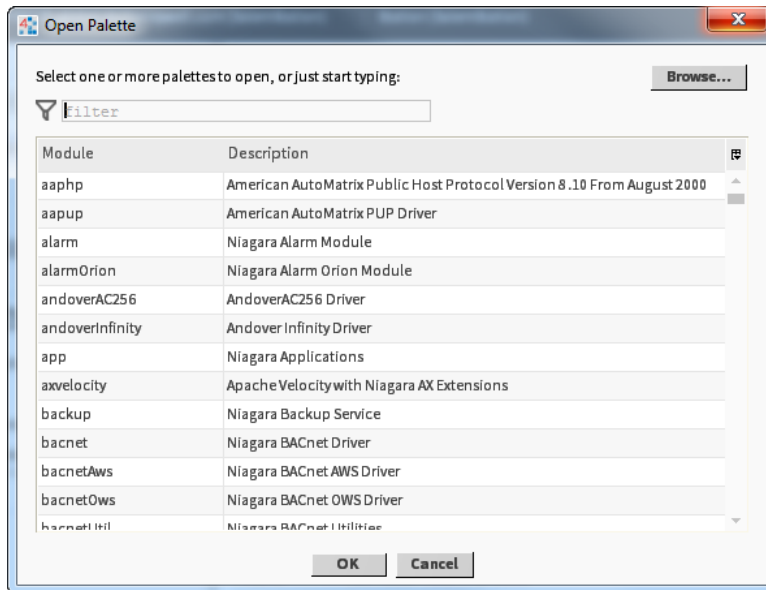
The palette toolbar includes the following:

1	Open palette button — opens the Open Palette dialog box. Refer to “To open a palette” for information about opening palettes.
2	Close palette button — closes the currently displayed palette.
3	Preview button — toggles the preview pane on and off. Previews are available on some components.
4	Dropdown palette selector — when palettes are open in the palette 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 158 Open Palette window



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

- **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

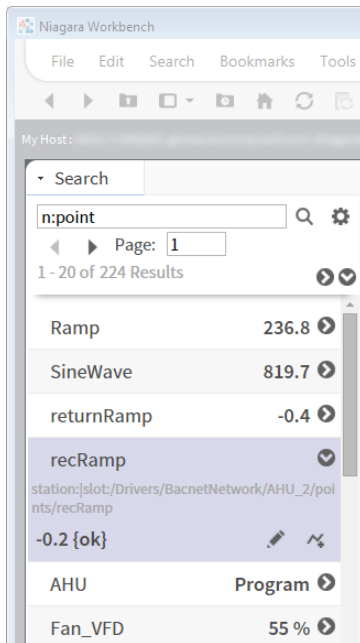
- **Description**

This is a short title or name of the palette's parent module

About the Search side bar

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

Figure 159 Search side bar



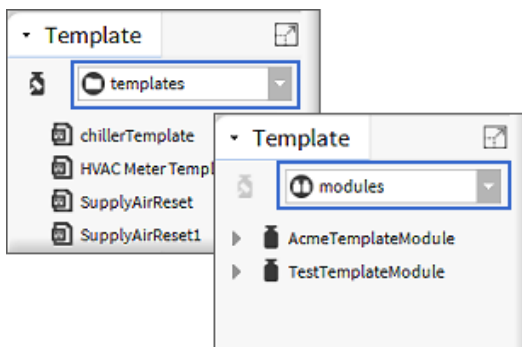
The **Search** side bar provides a query field for entering your search criteria. For example, you might enter "n:point" (as shown) to query for all points in the station that have the Niagara tagdictionary point tag.

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.

Figure 160 Template side bar



The pull-down list in the side bar switches the view between the `\templates` and `\modules` folders. When the `\modules` folder is selected, expand any module to see the template files contained within.

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.

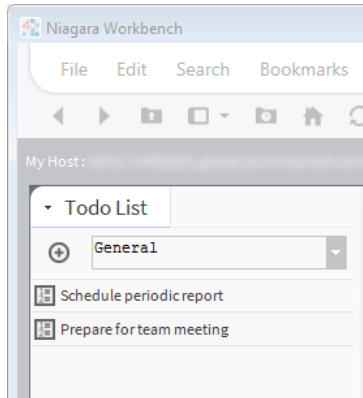
NOTE:

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. In order to make changes you must first click **Save As** and save the template with a different filename in the Workbench User Home /templates folder.

About the Todo list sidebar

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

Figure 161 Todo bar

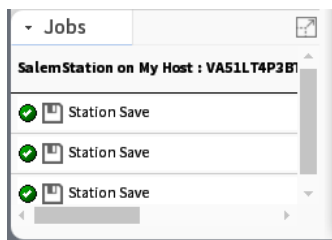


Click the **+** button on the toolbar to open the **Add** window. This window provides a text fields for adding and categorizing Todo list items. For more details about the Todo List see "Todo List".

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 162 Jobs side bar



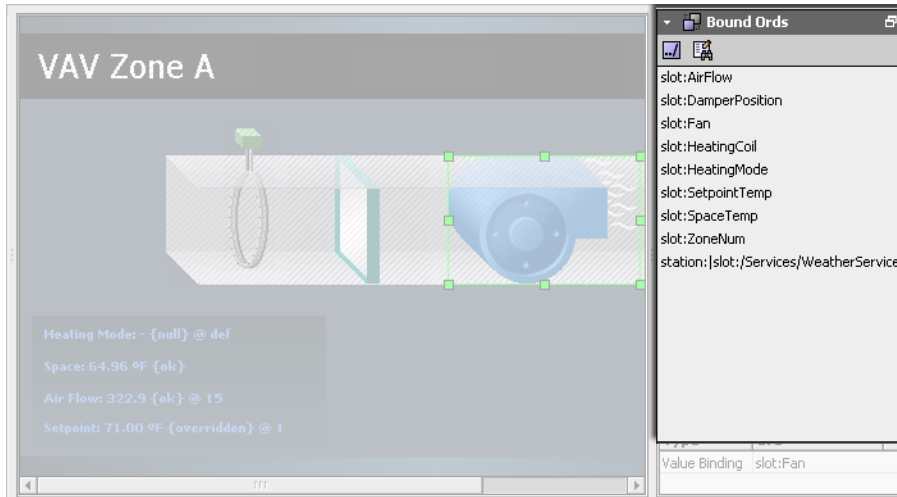
The current status of each job is shown as: running, canceling, canceled, success, or failed. If the job is running, a progress bar displays estimated progress.

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. For details about using the job side bar, refer to "Using the jobs side bar".

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 163 Bound Ords side bar

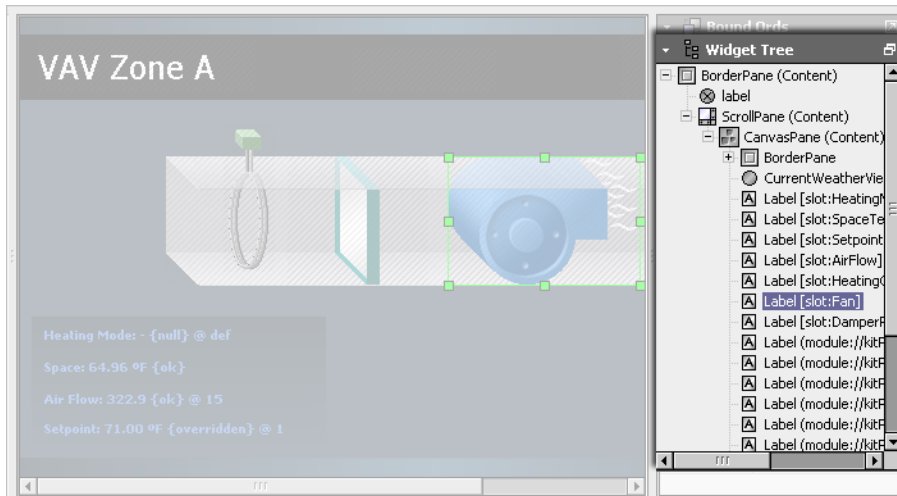


Double click on any ORD in the list to display the ORD in the ORD editor window.

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 164 Widget tree side bar

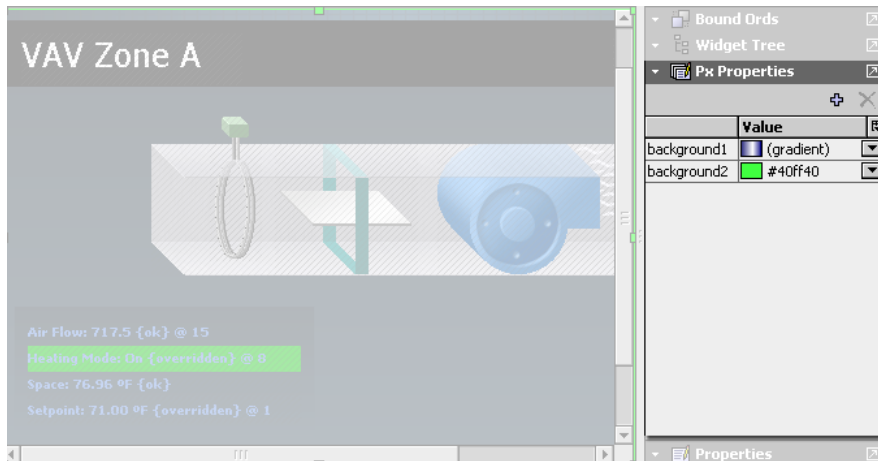


It is often easier to use the Widget Tree to select objects when you have a lot of objects on a view—especially when there are several layers of objects. When you select an object in the tree view it is selected in the Px view as well and displays the selection borders and handles.

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 165 Px properties side bar

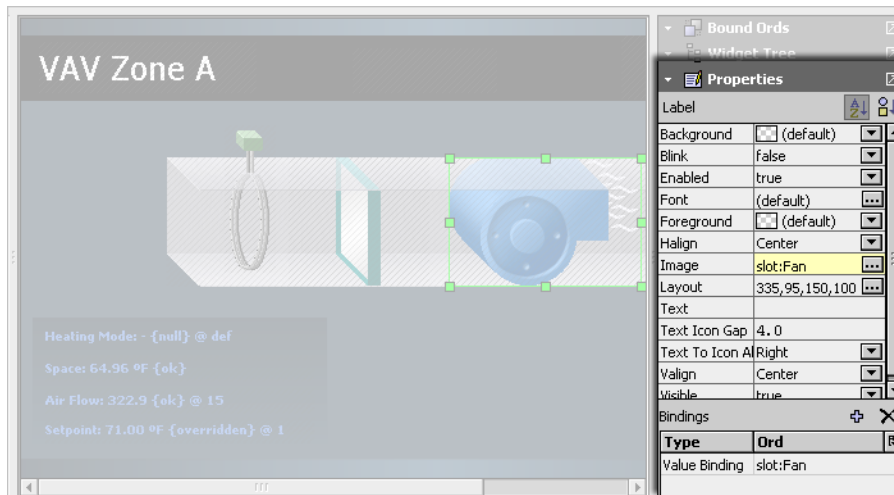


Use the menu bar icons to add, define, assign, and delete Px properties. For more information about Px Properties, see the *Graphics Guide* section “About Px Properties”.

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 166 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.

- 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 command 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

For an overview of the Workbench toolbar, refer to “About the toolbar”.



















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:

- **Standard toolbar icons**
These icons may be dimmed (or unavailable in certain views) but they are always present on the toolbar. Refer to “Standard toolbar icons” for the listing of Standard toolbar icons.
- **Slot sheet toolbar icons**
These icons appear only when the Slot Sheet is active. Refer to “Slot Sheet toolbar icons” for the listing of Slot Sheet toolbar icons.
- **Px Editor toolbar icons**
These icons appear only when the Px Editor or Px Viewer is active. Refer to “Px Editor toolbar icons” for the listing of Px Editor toolbar icons.
- **History extension manager toolbar icons**
These icons appear when the History Editor view is active. Refer to “About the history extension manager toolbar icons” for the listing of History Editor toolbar icons.
- **History Editor toolbar icons**
These icons appear when the History Editor view is active. Refer to “About the history editor toolbar icons” for the listing of History Editor toolbar icons.
- **Todo list toolbar icons**

These icons appear when the Todo list view is active. Refer to “About the Todo list toolbar icons” for the listing of Todo list toolbar icons.





Standard toolbar icons

For an overview description of the toolbar, refer to “About the toolbar”. Following are the icons that appear in all views:

-  Back. See “Back” for details about the Back function.
-  Forward. See “Forward” for details about the Forward function.
-  Up Level. See “Up Level” for details about the Up Level function.
-  Recent Ords. See “Recent Ords” for details about the Recent Ords function.
-  Home. See “Home” for details about the Home function.
-  Refresh. See “Refresh” for details about the Refresh function.
-  Left Side Bar. See “Side Bars” for details about the Side Bars function.
-  Open. See “About the File menu” for details about all the Open functions.
-  Save Ctrl + S. See “Save” for details on the Save function.
-  Save Bog. See “Save Bog” for details on the Save Bog function.
-  Printing Ctrl + P. See “Printing” for details on the Printing function.
-  Cut Ctrl + X. See “Cut” for details on the Cut function.
-  Copy Ctrl + C. See “Copy” for details on the Copy function.
-  Paste Ctrl + V. See “Paste” for details on the Paste function.
-  Duplicate Ctrl + D. See “Duplicate” for details on the Duplicate function.
-  Delete Delete. See “Delete” for details on the Delete function.
-  Undo Ctrl + Z. See “Undo” for details on the Undo function.
-  Redo Ctrl + Alt + Z. See “Redo” for details on the Printing function.











Slot Sheet toolbar icons

For an overview description of the toolbar, refer to “About the toolbar”. Following are the icons that appear when the Slot Sheet view is active:

-  Add Slot
Creates a new slot (appearing as a row) on the slot sheet.
-  Rename Slot
Displays the **Rename Slot** dialog box, when clicked. This icon is dimmed when no slot, or more than one slot is selected in the slot sheet editor view.
-  Config Flags
Displays the **Config Flags** dialog box, when clicked. This icon is dimmed unless one or more slots are selected in the slot sheet editor view.
-  Reorder
Displays the **Reorder** dialog box, when clicked.

Px Editor toolbar icons





For an overview description of the toolbar, refer to “About the toolbar”. Following are the icons that appear when the Px Editor is the active view:

-  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. See the *NiagaraAX Graphics Guide* section “About Px views” for more details about the Px Editor and Px Viewer.
-  Right (Px Editor) side bar menu
Displays a dropdown menu of side bar options for the Px Editor. The following options are available:
 - Bound Ords
Shows or hides the Bound Ords side bar (refer to “About the bound ords side bar”).
 - Widget Tree
Shows or hides the Widget Tree side bar (refer to “About the widget tree side bar”).
 - Properties
Shows or hides the Properties side bar (refer to “About the properties side bar”).
-  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





The Workbench toolbar contains navigation and editing buttons as described in [About the toolbar](#).

For an overview description of the history editor view, refer to ["About the history extension manager view"](#). Following, are the icons that appear when the history extension manager view is active:

-  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

For an overview description of the history editor view, refer to ["About the history editor view"](#). Following are the icons that appear when the history editor view is active:






-  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
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).
-  Filter
Displays the **Configure Flags** dialog box, when clicked. This icon is available even if no records are selected.
-  Configure Outliers
Displays the **Configure Outliers** dialog box, when clicked.

About the Todo list toolbar icons

For an overview description of the Todo list view, refer to ["Todo List"](#). Following are the icons that appear when the Todo list view is active:

-  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.
-  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 following sections provide descriptions of how to use the Workbench console in terms of:

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

- Niagara shell commands
These commands are used at the command line and may be typed in directly. See “Niagara shell commands”:
- nre commands
These commands are used at the command line and may be typed in using the “nre” prefix. See “nre (station) commands”.
- wb commands
These commands are used at the command line and may be typed in using the “wb” prefix. See “wb (Workbench) commands”.
- plat commands
These commands are used at the command line and may be typed in using the “plat” prefix. See “plat (platform) commands”:

Niagara shell commands

- cd
This command displays and changes the current directory. Type `cd <directory name>` to change to a specific directory. Type `cd` to display the current directory.
- debug
This command turns debug tracing on and off. Type `debug on` to turn on debug. Type `debug off` to turn off debug.
- print
This command prints a message to the output stream. Type `print <message>` to print the literal message to the console.
- reset

This command resets the environment to its default state. Type `reset` to set the environment to its default state.

- `set`

This command displays and modifies environment variables.

- Type `set` to display all the environment variables.
- Type `set <prefix>` to display all the environment variables that start with the specified prefix.
- Type `set <name>=` to remove the “named” variable.
- Type `set <name>=<value>` to set the “named” variable to the “value” specified.

- `which`

This command resolves a filename in a path. Type `which <filename>` 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.

- `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 to `main`.

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

- `-version`

This option displays the `nre` version.

- `-modules:<x>`

This option displays the modules that match the pattern defined by “x”.

- `-hosted`

This option displays the id for the system host.

- `-licenses`

This option displays a summary of the license information.

- `-props`

This option displays a list of the system properties.

- `-locale:<x>`

This option allows you to set the default locale. For example, to set the default locale to US English, type:
`-locale:en_US`

- `-@option`

This option allows you to pass the specified option to the Java VM.

- `-testheap`

This option tests and displays the max heap size.

- `-buildreg`

This option causes a rebuild of the registry.

wb (Workbench) 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.

- ORD

This option specifies the ORD of the initial view that you want display when Workbench starts up.

The following options may be used with the wb command.

- -profile

This option specifies the workbench profile to assign when Workbench starts up.

- -file:ord

This option specifies the initial file to display when Workbench starts up.

- -locale<x>

This option sets the locale on startup.

- -@<option>

This options allows you to pass the specified option to the Java VM.

plat (platform) commands

Use the following syntax with the plat command:

```
plat <command> <flags> <command-flags>
```

The following commands may be used with plat.

- details

This command displays a configuration summary for a remote host.

- fget

This command gets one or more files from a remote host.

- flist

This command provides file details for a single file, or for all files in a directory.

- ipconfig

This command displays the TCP/IP configuration for a remote host.

- jacejar

This command creates Niagara module files that can be run on embedded hosts.

- liststations

This command lists stations that are managed by the Niagara platform daemon.

- moduleinstall

This command installs Niagara modules to a remote host.

- reboothost

This command requests that a remote Niagara platform daemon reboot its host.

- script

This command runs one or more platform commands in a script.

- `startstation`
This command requests that a Niagara platform daemon start a station.
- `stopstation`
This command requests that a Niagara platform daemon stop a station.
- `tellstation`
This command sends text to the console of a running Niagara station.
- `watchstation`
This command monitors the output of a niagara station.
- `installdaemon`
This command installs the Niagara platform service (Win32 only).
- `uninstalldaemon`
This command removes the Niagara platform service (Win32 only).
- `installdialup`
This command installs the Niagara dialup service (Win32 only).
- `uninstalldialup`
This command removes the Niagara dialup service (Win32 only).

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

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