

Technical Document

NiagaraAX User Guide

AX-3.8 and AX-3.7u1

November 4, 2013



Niagara^{AX} User Guide

Copyright © 2013 Tridium, Inc.

All rights reserved.

3951 Westerre Pkwy., Suite 350

Richmond

Virginia

23233

U.S.A.

Confidentiality Notice

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 Notices

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, Niagara^{AX} Framework, and Sedona Framework are registered trademarks, and Workbench, WorkPlace^{AX}, and ^{AX}Supervisor, 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.

© Tridium, Inc. 2013. All rights reserved. The product(s) described herein may be covered by one or more U.S or foreign patents of Tridium.

CONTENTS

Preface	xiii
Document Change Log	xiii
About NiagaraAX Framework	1-1
Well designed	1-1
The Niagara Framework solution	1-2
About control systems integration	1-2
About Java	1-3
About Virtual Machines	1-3
About common networking and Internet protocols	1-3
About programming for non-programmers	1-3
About embedded systems capabilities	1-4
About distributed systems capabilities	1-4
About component software design	1-4
About Niagara software architecture	1-4
About Baja	1-5
About APIs	1-6
<i>javax.baja</i>	1-6
<i>com.tridium</i>	1-6
About Niagara building blocks	1-6
About modules	1-6
About module characteristics	1-7
About jar files	1-7
About module versions	1-7
About module directory structure	1-8
About benefits of modular software development	1-8
About components	1-9
About slots	1-9
Slot type	1-10
Slot name	1-10
Slot definition	1-10
Flags	1-10
Facets	1-10
About master/slave components	1-11
About point components	1-11
About component naming	1-12
Escaped names	1-12
About palettes	1-12
About presentation	1-13
About presentation modules	1-13
About presentation xml (Px)	1-13
About presentation design philosophy	1-14
About presentation media	1-14
About stations	1-15
About BOG files	1-16
About ORDs	1-16

Examples of ORDs:	1-17
About schemes	1-17
Types of schemes	1-17
Types of space	1-18
Types of files	1-18
About file naming	1-19
About views	1-19
About lexicons	1-20

About Workbench.....2-1

Tour of the Workbench GUI	2-1
About Workbench window controls	2-1
About the side bar panes	2-2
About side bars	2-3
About the side bar title bar	2-3
Types of side bars	2-3
About the bookmark side bar	2-4
About the help side bar	2-5
About Jobs side bar	2-5
About the nav side bar	2-6
Types of nodes in the nav side bar tree	2-6
About the nav side bar toolbar	2-7
About the palette side bar	2-8
About the palette side bar toolbar	2-8
About the Open Palette dialog box	2-9
About the Todo list sidebar	2-9
About the jobs side bar	2-9
About the bound ords side bar	2-10
About the widget tree side bar	2-10
About the Px properties side bar	2-11
About the properties side bar	2-11
About the view pane	2-12
About the console	2-13
About the menu bar	2-13
About the toolbar	2-13
About the locator bar	2-14
About the view selector	2-15
Types of popup menus	2-15
About the nav side bar popup menus	2-15
About the wire sheet popup menu	2-16
About the property sheet popup menu	2-16
Px Editor popup menu	2-17
Types of data-presentation controls and options	2-17
Table controls and options	2-18
Batch editing (or batch processing)	2-19
Chart controls and options	2-19
Customizing the Workbench environment	2-20
Creating Additional Windows	2-20
Editing popup menu "new" item	2-21
Creating tabs in the view pane	2-21
Working with the Wire Sheet view	2-22
Types of Workbench options	2-25
General options	2-26
Alarm Console options	2-27
Alarm Portal options	2-28
Text Editor options	2-28
Lexicon Editor options	2-29
Bajadoc options	2-29
Station Copier options	2-29
R2 Platform Tool options	2-29
Px Editor options	2-30

Wire Sheet options	2-31
About Workbench themes	2-32
Types of themes	2-32
Define a default custom theme for your brand	2-33
<i>Designate a default theme</i>	2-33
About theme modules	2-33
<i>Create a new theme module</i>	2-34
<i>Copy resources from a built-in theme module</i>	2-35
<i>Modify design elements in the NSS file</i>	2-36
<i>Build a .jar file for the theme module</i>	2-36
<i>Activate the new theme module</i>	2-37

Data and Control Model 3-1

About control points	3-1
Types of control points	3-1
Other control components	3-2
About data value categories	3-2
About point types	3-2
About point Out	3-2
About point inputs	3-3
About point actions	3-3
<i>About override actions</i>	3-4
<i>About set (Fallback) action</i>	3-4
<i>Modifying default actions</i>	3-5
About point facets	3-7
<i>Accessing and editing facets</i>	3-7
<i>Facets importance for Enum points</i>	3-8
<i>Effect of facets on point actions</i>	3-9
<i>Maximum override duration facet</i>	3-10
About point extensions	3-12
Extension process overview	3-12
Types of point extensions	3-12
About the proxy extension	3-13
About control extensions	3-13
<i>Types of control extensions</i>	3-14
About alarm extensions	3-14
<i>Types of alarm extensions</i>	3-15
About history extensions	3-16
<i>Types of history extensions</i>	3-16
About control triggers	3-17
How triggers are used	3-17
About related objects	3-17
About point status	3-18
Types of status flags	3-18
Priority of status indication	3-19
How status flags are set	3-19
<i>Simple control point status</i>	3-19
<i>Propagate Flags status option (linked Math and Logic objects)</i>	3-20
About "isValid" status check	3-21
About writable points	3-21
About the priority scheme	3-21
<i>Priority input scan</i>	3-21
<i>Priority linking rules</i>	3-22
<i>Priority level conventions</i>	3-22
About minimum On and Off times	3-22
About composites	3-23
Some composite examples	3-23
<i>Point-level composite</i>	3-23
<i>Folder-level composite</i>	3-25
Composite issues	3-26

About Histories 4-1

Types of History Services	4-2
About the history service	4-2
About the history extension manager view	4-3
About history service property sheet	4-3
History service actions	4-3
About the audit history service	4-4
About the log history service	4-5
Types of history space views	4-5
About the history chart builder view	4-6
About Time Zones and the Chart Builder view	4-8
About the database maintenance view	4-8
About the nav container view	4-9
Types of history views	4-10
Types of history data fields	4-11
About the history chart view	4-11
Charting in the browser	4-11
About the Live History Chart View	4-12
Selecting the Live History Chart View	4-13
Using relative history extension ords with HistoryPointList	4-13
About the history table view	4-13
About the collection table view	4-14
About the history summary view	4-15
About the history editor view	4-15
About the history slot sheet view	4-16
About the history property sheet view	4-17
Example 1: History imported from remote controller to Supervisor	4-18
Example 2: History exported to Supervisor by remote controller	4-18
About the metadata browser view	4-18
About On Demand history views	4-20
About the history process	4-22
About delta logging	4-23
Add history extensions to a component	4-24
Configure history extensions	4-25
About history names	4-26
Archiving	4-27
About editing history data	4-28
About System Tags	4-29
About history grouping	4-30
About the History Group property sheet view	4-31
About the History Group Manager view	4-31
About history nav shortcuts	4-32
About nesting history shortcuts	4-33

About Alarms 5-1

Alarm examples	5-1
Overview of the alarming process	5-2
About alarm sources	5-3
About alarm extensions and components	5-3
About alarm extension properties	5-3
About alarm instructions	5-7
About notes	5-7
About the alarm service	5-7
About memory alarm service	5-8
About the On Call Service	5-8

About On Call Service alarm data in the alarm console	5-9
About the On Call List	5-10
About the On Call List Manager view	5-10
About the On Call Contact	5-11
About the On Call Contact Manager view	5-12
About the On Call User Report view	5-13
Exporting the On Call User Report view	5-14
About the Email Alarm Acknowledger	5-15
About the EmailService	5-16
About the OutgoingAccount	5-17
About the IncomingAccount	5-19
About the Email Recipient (report-EmailRecipient)	5-21
About the SmsService	5-22
About the Sms Alarm Acknowledger	5-22
Using Spy pages with the On Call Service	5-24
Common alarm controls and indicators	5-25
About the alarm database	5-25
Types of alarm service views	5-26
About alarm extension manager	5-26
<i>Types of extension manager display features</i>	<i>5-27</i>
About the Instructions Manager view	5-27
About the alarm class summary view	5-29
About the alarm database maintenance view	5-30
<i>About the Alarm details dialog box</i>	<i>5-31</i>
<i>Types of alarm database maintenance</i>	<i>5-31</i>
About the alarm database view	5-31
About alarm service property sheet	5-32
About alarm class	5-32
About alarm escalation	5-33
About alarm class properties	5-34
About alarm class mapping	5-34
Types of alarm recipients	5-36
About the console recipient	5-36
About the alarm console	5-37
<i>Hide, show, add, and remove alarm data columns in the alarm console</i>	<i>5-37</i>
<i>Types of Alarm Details View dialog boxes</i>	<i>5-38</i>
About the station recipient	5-41
About the On Call Recipient	5-42
About the Email Recipient (email-EmailRecipient)	5-43
About the Sms recipient	5-44
About the lineprinter recipient	5-45
About the printer recipient	5-46

About Security 6-1

Security changes and notes	6-2
Security model overview	6-2
Categories and security	6-3
Users and security	6-4
<i>About built-in users</i>	<i>6-6</i>
<i>About authentication</i>	<i>6-6</i>
Network users	6-7
<i>Challenge</i>	<i>6-7</i>
<i>User Service changes</i>	<i>6-8</i>
<i>NiagaraNetwork changes</i>	<i>6-8</i>

Permissions and security	6-8
About permission levels	6-10
Component permission levels	6-10
File permissions	6-11
History permissions	6-11
UserService security notes	6-11
About user audits	6-12
AuditHistoryService configuration	6-12
AuditHistory content	6-12
AuditHistory station login activities	6-13
Multi-station security notes	6-15
Domain-wide cookies authentication	6-15
To configure domain-wide cookies authentication	6-15
Workbench single sign-on (SSO) realm	6-17
To configure station's realms for Workbench single sign-on	6-17
Alarm ack permission notes	6-18
About password expiration and reset	6-18
Password expiration	6-18
Password reset	6-21
Password history (unique passwords)	6-23
UserService	6-24
UserService properties	6-24
Strong password notes	6-25
Password Strength (AX-3.8)	6-25
Require Strong Passwords (AX-3.7)	6-27
Stronger passwords	6-28
User Manager	6-28
Lockout notes	6-29
Network user notes	6-29
User	6-30
Network user related properties	6-34
Permissions Browser	6-34
Show Configured	6-34
Permissions abbreviations	6-35
Access permissions	6-35
About User prototypes	6-36
Properties of User Prototypes	6-36
Default Prototype	6-37
Default Prototype importance in Network users scenario	6-37
Additional (non-default) User Prototypes	6-37
Specifying additional "local override" properties	6-38
Naming User Prototypes	6-38
CategoryService	6-38
Category Manager	6-39
Category properties	6-40
Category caveats	6-40
Category Browser	6-40
Show Configured	6-41
Category Sheet	6-41

About Scheduling 7-1

About the scheduling model	7-1
Schedule component categories	7-1
Schedule component views	7-2
Refresh	7-2
Save	7-2
Schedule component links	7-2
Weekly schedule links	7-3
Trigger schedule links	7-3
Schedule special events	7-3
Special events permission change	7-3
Schedule exports and imports (master/slave)	7-4

About weekly schedules	7-5
Types of weekly schedule components	7-5
About the BooleanSchedule	7-6
About the EnumSchedule	7-6
About the NumericSchedule	7-6
About the StringSchedule	7-6
Weekly schedule output processing	7-6
Out slot	7-6
Out Source slot	7-6
Next Time and Next Value slots	7-6
Weekly Scheduler view	7-7
Weekly Schedule	7-7
Special Events	7-8
Properties	7-11
Summary	7-13
Event colors in weekly schedules (AX-3.8)	7-14
Default event colors for EnumSchedules and BooleanSchedules	7-14
Customizing schedule event colors globally (station-wide)	7-15
Customizing schedule event colors at the component level	7-16
Customize schedule event colors using facets	7-16
About calendar schedules (holidays)	7-18
CalendarSchedule Usage	7-18
Calendar Scheduler view	7-19
Adding calendar events	7-20
Right-click menus and other controls	7-20
Calendar day selections	7-21
Date selection notes	7-21
Date range selection notes	7-21
Week and day selection notes	7-21
Custom selection notes	7-22
CalendarSchedule slots and other notes	7-23
About trigger schedules	7-23
Trigger Scheduler view	7-23
Adding trigger events	7-24
Adding trigger event times	7-25
Right-click menus and other controls	7-25
TriggerSchedule slots and other notes	7-26
Trigger Missed slot	7-26
About ScheduleSelector components	7-27
Types of ScheduleSelector components	7-27
Types of ScheduleSelector component properties	7-27
Example ScheduleSelector configurations	7-28
Example 1: Using a NumericScheduleSelector component	7-28
Example 2: Using a BooleanScheduleSelector component	7-28
Using schedules	7-29
Adding a schedule or calendar	7-29
To add a component from the schedule palette	7-29
To copy an existing, saved (configured) schedule component	7-29
Configuring schedules and calendars	7-30
Configuring weekly schedules	7-30
To configure a weekly schedule	7-30
To configure a weekly schedule's properties	7-30
To configure the weekly (normal) schedule	7-30
To add and configure special events	7-30
To review a weekly schedule's configuration	7-31
Configuring calendar schedules	7-31
To configure a calendar schedule	7-31
Configuring trigger schedules	7-31
To configure a trigger schedule	7-31
Linking schedules	7-32
Linking weekly schedules	7-32
To link a weekly schedule using the wire sheet	7-32
To link a weekly schedule using the Nav tree	7-32
Linking trigger schedules	7-33

<i>To link a trigger schedule using the wire sheet</i>	7–33
<i>To link a trigger schedule using the Nav tree</i>	7–33
Importing schedules	7–33
Importing NiagaraAX schedules or Bacnet Schedules	7–34
<i>To import a NiagaraAX schedule or BACnet Schedule or Calendar</i>	7–34

About Workbench tools 8–1

Types of Workbench Tools	8–1
Alarm portal	8–3
Bacnet Service	8–4
Lexicon Tool	8–5
Lexicon Report view	8–5
Lexicon Editor view	8–5
Lexicon Module Builder view	8–6
Local License Database	8–6
Lon Xml Tool	8–7
Lonworks Service	8–8
Credentials manager	8–8
New Driver wizard	8–9
New Module wizard	8–10
New Station wizard	8–10
Request License	8–12
Resource Estimator	8–12
Time Zone Database Tool	8–13
Manifest Info	8–13
Time Zone Database	8–14
Daylight Savings Time Calculator	8–14
Time Zone Calculator	8–14
Todo List	8–15
Workbench Job Service	8–15
Workbench Service Manager	8–15

Performing Workbench tasks 9–1

Starting and exiting Workbench	9–1
<i>To start Workbench</i>	9–1
<i>To close a Workbench window</i>	9–2
<i>To exit Workbench</i>	9–2
Getting started with stations	9–2
<i>To open a platform using Workbench GUI</i>	9–2
<i>To connect to a platform using Workbench GUI</i>	9–3
<i>To close a platform using Workbench</i>	9–4
<i>To disconnect from a platform using Workbench</i>	9–4
<i>To start a station using Workbench</i>	9–4
<i>To stop a station using Workbench</i>	9–4
<i>To open a station</i>	9–5
<i>To close a station using Workbench</i>	9–5
<i>To connect to a station using Workbench</i>	9–5
<i>To disconnect from a station</i>	9–6
Using the side bar pane	9–6
<i>To open the side bar pane</i>	9–6
<i>To open the bookmarks side bar</i>	9–6
<i>To open the help side bar</i>	9–7
<i>To open the nav side bar</i>	9–7

<i>To open the palette side bar</i>	9-7
<i>To close a side bar</i>	9-7
<i>To close the side bar pane</i>	9-7
<i>To vertically resize a side bar</i>	9-7
<i>To restore a side bar to a previous size</i>	9-7
<i>To maximize a side bar</i>	9-7
Using bookmarks	9-8
<i>To add a bookmark</i>	9-8
<i>To manage bookmarks</i>	9-8
<i>To edit a bookmark</i>	9-8
<i>To remove a bookmark</i>	9-8
Using the help side bar	9-8
<i>To search for topics in the help side bar</i>	9-9
Using the nav side bar (tree)	9-9
<i>To refresh a nav side bar tree node</i>	9-9
<i>To Go Into a nav side bar tree node</i>	9-9
Using the jobs side bar	9-10
<i>To view job details</i>	9-10
<i>To cancel a job from the jobs side bar</i>	9-10
<i>To dismiss a job</i>	9-10
Using the palette side bar	9-10
<i>To open a palette</i>	9-10
<i>To close a palette</i>	9-11
<i>To create a palette</i>	9-11
<i>To add a new component to a palette (PaletteFile.palette)</i>	9-11
<i>To add a new component to a palette (module.palette)</i>	9-11
<i>To copy a component to a palette</i>	9-12
<i>To add a side bar preview using the compPreviewWidget property</i>	9-12
Using the view pane	9-13
Editing components	9-13
Selecting a component	9-13
<i>To select a component</i>	9-13
<i>To select multiple components in the nav tree</i>	9-13
<i>To select a multiple components in the wire sheet</i>	9-14
Viewing a component	9-14
<i>To select a component view</i>	9-14
<i>To reorder components (in a component container)</i>	9-14
<i>To create a composite view</i>	9-15
Working with components	9-15
<i>To add a new component (property sheet, nav side bar, or wire sheet view)</i>	9-15
<i>To delete a component (property sheet, nav side bar, or wire sheet view)</i>	9-16
<i>To rename a component (property sheet, nav side bar, or wire sheet view, slot sheet)</i>	9-16
<i>To duplicate a component (property sheet, nav side bar, or wire sheet view)</i>	9-16
<i>To paste a component (property sheet, nav side bar, or wire sheet view)</i>	9-16
<i>To drag a component to the property sheet, nav side bar, or wire sheet (left-click method)</i>	9-17
<i>To drag a component to the property sheet, nav side bar, or wire sheet (right-click method)</i>	9-17
Editing component data	9-17
Using the property sheet	9-17
<i>To add an extension to a component property sheet view</i>	9-17
<i>To add a component facet</i>	9-18
<i>To remove a component facet</i>	9-18
Using the wire sheet	9-18
<i>To pin a slot</i>	9-18
<i>To paste a component onto the wire sheet</i>	9-19
<i>To drag a component to the wire sheet (left-click method)</i>	9-19
<i>To drag a component to the wire sheet (right-click method)</i>	9-19
Using the slot sheet	9-19
<i>To add a new slot</i>	9-19
<i>To delete a slot</i>	9-20
Using the link sheet	9-20
<i>To delete a link using the link sheet view</i>	9-20
<i>To edit a link using the link sheet view</i>	9-20

Using alarms	9-20
Creating alarms	9-20
<i>To set up alarm conditions for a control point</i>	<i>9-21</i>
Routing alarms	9-21
<i>To add an alarm class to the alarm service</i>	<i>9-21</i>
<i>To add a console recipient to the alarm service</i>	<i>9-21</i>
<i>To add a station recipient to the alarm service</i>	<i>9-21</i>
<i>To add an email recipient to the alarm service</i>	<i>9-22</i>
Managing alarms	9-22
<i>To acknowledge alarms from the alarm console view</i>	<i>9-22</i>
<i>To acknowledge alarms from the alarm portal</i>	<i>9-22</i>
<i>To delete alarm records</i>	<i>9-23</i>
Creating a Live History Chart	9-24
<i>Live History Chart view from a History Extension</i>	<i>9-24</i>
<i>Live History Chart view from a HistoryPointsList Component</i>	<i>9-24</i>
Working with modules	9-24
Creating a new (.jar) module	9-24
<i>To create a new (.jar) module</i>	<i>9-24</i>

Component Guides.....10-1

Component Reference Summary	10-1
Components in alarm module	10-1
<i>Example</i>	<i>10-5</i>
Components in alarmRdb module	10-5
Components in backup module	10-5
Components in baja module	10-7
Components in chart module	10-12
Components in control module	10-12
Components in converters module	10-14
Components in crypto module	10-14
Components in email module	10-14
Components in file module	10-17
Components in flr module	10-19
Components in help module	10-19
Components in history module	10-19
Components in net module	10-24
Components in onCall module	10-24
Components in program module	10-25
Components in the Sms module	10-25
Components in schedule module	10-26
Components in timesync module	10-27
Components in web module	10-28
Components in workbench module	10-30

Plugin Guides.....11-1

Types of plugin modules	11-1
Plugins in alarm module	11-1
Plugins in backup module	11-3

Plugins in chart module 11-3

Plugins in email module 11-3

Plugins in help module 11-4

Plugins in history module 11-5

Plugins in html module 11-7

Plugins in onCall module 11-10

Plugins in program module 11-11

Plugins in raster module 11-13

Plugins in schedule module 11-13

Plugins in timesync module 11-14

Plugins in wiresheet module 11-14

Plugins in wbutil module 11-16

Plugins in workbench module 11-17

References.....A-1

In this appendix (introduction) A-1

About keyboard shortcuts A-1

Types of menu bar items A-2

 About the File menuA-2

 About the Edit menuA-4

 About the Search menuA-5

 About the Bookmarks menuA-6

 About the Tools menuA-6

 About the Window menuA-6

 About the Px Editor menuA-7

 About the History Ext Manager menuA-8

 About the Help menuA-8

Types of popup menu items A-8

 About the nav side bar popup menu itemsA-9

 About the wire sheet popup menu items A-10

 About the property sheet popup menu items A-10

 About the Px Editor popup menu items A-10

 About the history extension manager popup menu items A-11

 About the Todo list popup menu items A-11

 About the point manager popup menu items A-12

Types of edit commands A-13

Types of toolbar icons A-13

 Standard toolbar icons A-14

 Slot Sheet toolbar icons A-14

 Px Editor toolbar icons A-14

 About the history extension manager toolbar icons A-15

 About the history editor toolbar icons A-15

 About the Todo list toolbar icons A-15

Types of console commands A-16

 Niagara shell commands A-16

 nre (station) commands A-17

 wb (Workbench) commands A-17

 plat (platform) commands A-17

Glossary.....B-1

PREFACE

Preface

[Document Change Log](#)

Document Change Log

Updates (changes/additions) to this *NiagaraAX User Guide* document are listed below.

- NiagaraAX-3.8 release, November 4, 2013
Document updated to cover both the initial AX-3.8 release as well as AX-3.7 and AX-3.7u1. A significant change was the removal of the prior main section “About Presentation XML (Px)”, which explained graphics usage and related topics. This content was relocated to a new standalone *NiagaraAX Graphics Guide* document, including updates to cover AX-3.8 features and changes. Related to this, these following other sections were removed from this document:
 - In the section “[Performing Workbench tasks](#)”, task subsections “Using the Px Editor” and “Using the Nav File Editor” were removed. Such tasks are now in the *NiagaraAX Graphics Guide*.
 - In the section “[Component Guides](#)”, summary descriptions for components in the following modules were removed: `bajau`, `gx`, `kitPx`, `pxeditor`, and `report`. Summaries for components in these modules are now in the *NiagaraAX Graphics Guide*.
 - In the section “[Plugin Guides](#)”, summary descriptions for plugins (views) in the `pxEditor` and `report` modules were removed. These summaries are now in the *NiagaraAX Graphics Guide*.

Other areas of this *NiagaraAX User Guide* document that are new or changed are described below:

- In the section “[Types of Workbench options](#)” on page 2-25, the subsection “[General options](#)” was updated, and a new subsection “[R2 Platform Tool options](#)” was added. A “[Color Chooser](#)” subsection was removed. The section “[About Workbench themes](#)” on page 2-32 (which explains one selection in the “General options”), was moved to follow the last of the Workbench options.
- In the main “[Data and Control Model](#)” section, control point subsections “[About override actions](#)” and “[About point facets](#)” were edited to reference a new facets subsection “[Maximum override duration facet](#)” on page 3-10 (unrelated to any recent release, i.e. AX-3.8, AX-3.7, etc.).

In the main “[About Security](#)” section, the following subsections had changes:

- The section “[Security changes and notes](#)” on page 6-2 was updated for AX-3.8, summarizing changes in strong passwords, among others.
- In the section “[About built-in users](#)” on page 6-6, an AX-3.8 change for [User guest](#) is noted.
- In the section “[Alarm ack permission notes](#)” on page 6-18, an AX-3.8 change was noted.
- The section “[Strong password notes](#)” on page 6-25 was expanded, including new subsection “[Password Strength \(AX-3.8\)](#)” on page 6-25 and other sections “[Require Strong Passwords \(AX-3.7\)](#)” on page 6-27 and “[Stronger passwords](#)” on page 6-28.

In the main “[About Scheduling](#)” section, most figures had updated screen captures, and most subsections were reviewed and a few property descriptions updated. In the main “[About weekly schedules](#)” subsection, a new section “[Event colors in weekly schedules \(AX-3.8\)](#)” on page 7-14 was added. It includes subsections about default BooleanSchedule and EnumSchedule event colors, and how you can further customize them.

The “[About Workbench tools](#)” section “[Types of Workbench Tools](#)” on page 8-1 was updated to list a “[Driver Upgrade Tool](#)”, and other tool summaries were updated to be more accurate. A few tools subsections were updated, including “[Local License Database](#)” on page 8-6, “[Credentials manager](#)” on page 8-8, and “[New Station wizard](#)” on page 8-10, the last of which has AX-3.8 related changes. In the section “[Performing Workbench tasks](#)”, procedures were modified for “[Starting and exiting Workbench](#)”, and “[Getting started with stations](#)”.

In the “[Component Guides](#)” and “[Plugin Guides](#)” sections, summaries for components and views for the `ldap` module were removed and relocated to a new *NiagaraAX LDAP / Active Directory Config-*

uration Guide. In the “Component Guides” section, the section about the [BackupService](#) was edited for changes made in AX-3.8.

- NiagaraAX-3.7 “Update 1” revision, May 30, 2013
Various document updates concurrent with the “update 1” release for AX-3.7 (in this document denoted as “AX-3.7u1”). Station security-related improvements are a main update focus. A new section “[About file naming](#)” on page 1-19 was added in the “[About NiagaraAX Framework](#)” concepts chapter. It explains the “approved set” of characters for naming files in a station’s file space. In the main “[About Security](#)” section, the following subsections had changes:
 - A new section was added, “[Security changes and notes](#)” on page 6-2.
 - In the section “[About built-in users](#)” on page 6-6, a few edits were made describing the [User admin](#) and [User guest](#), including a *Note* explaining changed password storage, and a *Caution* against enabling the guest user.
 - In the section “[About authentication](#)” on page 6-6, additional details were added describing some security changes.
 - In the section “[Permissions and security](#)” on page 6-8, edits were made about “super users” including a *Note* that recommends limiting the number of super users.
 - The section “[Multi-station security notes](#)” on page 6-15 was edited to reflect security enhancements, and a subsection “[Domain-wide cookies authentication](#)” on page 6-15 with procedure for configuration was edited to include a new `WebService` property “Single Sign On Enabled”.
 - In the section “[UserService properties](#)” on page 6-24, details were added about one property.
 - In the section “[About User prototypes](#)”, more details were added in the subsection “[Default Prototype](#)” on page 6-37, particularly about its “Password Configuration” properties.
- In the “[Component Guides](#)” section, the following changes were made:
 - In the section “[backup-BackupService](#)”, including subsection “[Restoring a backup](#)”, a couple of *Notes* were added explaining that station backups (and backup .dist installs), are impacted by changes in AX-3.7u1.
 - In the section “[web-WebService](#)” on page 10-28, descriptions were added for all properties. Previously, there was only a brief summary description of the `WebService`.
- NiagaraAX-3.7 release revision, August 29, 2012
Document was “version split” from the prior version describing AX-3.5 and AX-3.6, where most mention of NiagaraAX release behavior prior to AX-3.6 was removed.
Concurrent with date of this document, the areas with the most changes include the following:
 - In the “[About NiagaraAX Framework](#)” section, the section “[About modules](#)” on page 1-6 now mentions a new type of “synthetic module”, and refers to a separate *NiagaraAX Synthetic Modules* engineering notes document for more details.
 - In the “[About Workbench](#)” section, a new section “[About Workbench themes](#)” on page 2-32 was added for AX-3.7 and later, with related subsections “[Types of themes](#)” on page 2-32, “[Define a default custom theme for your brand](#)” on page 2-33, and “[About theme modules](#)” on page 2-33.
 - In the “[About Alarms](#)” section, subsection “[About the Sms Alarm Acknowledger](#)” on page 5-22 was updated. Several notes were added about AX-3.7 alarm enhancements, with a few updated screen captures. Sections affected include “[About alarm extension properties](#)” on page 5-3 (alarm text properties are now “multi-line”), “[Types of Alarm Details View dialog boxes](#)” on page 5-38 (previous and next buttons on Alarm Record details dialog, Filters dialog changes), “[About the lineprinter recipient](#)” on page 5-45 and “[About the printer recipient](#)” on page 5-46 (a new `PrinterRecipient` component).
 - In the “[About Security](#)” section, most subsections were updated with new screen captures and new sections were added about functions new in AX-3.7, including “[About password expiration and reset](#)” on page 6-18, “[Password expiration](#)” on page 6-18, “[Configuring expiring passwords](#)” on page 6-18, “[Expiring password operation](#)” on page 6-20, “[Password reset](#)” on page 6-21, and “[Password history \(unique passwords\)](#)” on page 6-23. Other changes were made in sections “[UserService](#)” on page 6-24 and descriptions of properties of a “[User](#)” on page 6-30. Additional new sections now include “[Strong password notes](#)” on page 6-25, and in an expanded section “[About User prototypes](#)” on page 6-36, more details in subsections “[Default Prototype](#)” on page 6-37 and “[Specifying additional “local override” properties](#)” on page 6-38.
 - In the “[About Workbench tools](#)” section, the former Lexicon Editor section changed to the “[Lexicon Tool](#)”, with overview coverage of the “[Lexicon Report view](#)” on page 8-5, “[Lexicon Editor view](#)” on page 8-5, and new tool “[Lexicon Module Builder view](#)” on page 8-6. Find more details about lexicons in the engineering notes document *NiagaraAX Lexicon Guide*.
 - In the “[Component Guides](#)” and “[Plugin Guides](#)” (views) sections that summarize items and provide context-sensitive help from Workbench, some new items were added, and a few icons and descriptions were updated. More work remains in these sections.

CHAPTER 1

About NiagaraAX Framework

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.

NiagaraAX is the third generation of Tridium's Niagara Framework. This Java-based software framework provides an infrastructure to enable 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 standard web browser. The Framework also includes a comprehensive toolset that enables non-programmers to build rich applications in a drag-and-drop environment.

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

Well designed

The Niagara Framework addresses the challenges of automation, control and device to enterprise connectivity of real time systems. NiagaraAX, and its derivative products, offer compelling value to both end users and partners. For OEM and reseller partners, the Niagara Framework solves several key challenges present in all control-related industries:

- High development costs associated with the foundation or infrastructure layer of software that communicates with devices and manipulates the data from them.
- The need to transform information from real-time control processes into a homogeneous structure enabling advanced product and service applications.
- Integration of legacy systems to enable companies to easily migrate their existing customers to new technologies and product offerings.

Partners who adopt the Niagara Framework as their infrastructure eliminate substantial engineering effort and are able to focus on their core competencies of application development and marketing. These companies gain competitive advantage in their markets from lower development costs and quicker time-to-market for their specific products, applications and value-add services.

End users of Niagara benefit from:

- Being able to preserve existing investments in control and monitoring devices as they move to new, standards-based technologies.
- Being able to access and control all of their diverse systems through a standard web browser.
- Combining information from different systems to support better overall management of their enterprise assets.
- Being able to specify interoperable systems and applications from multiple vendors, thereby reduc-

ing the potential for vendor lock-in.

Niagara is designed to address the following specific technical challenges:

- Platform independence - NiagaraAX works with a wide variety of operating systems and hardware platforms.
Unification of diverse systems - NiagaraAX integrates heterogeneous systems, protocols, and field-buses into normalized components.
Distributed architecture - NiagaraAX supports a highly distributed architecture. It successfully combines Internet/IT technology with the services necessary to implement real-time, peer-to-peer automation applications.
- Programming for non-programmers - NiagaraAX provides an environment and comprehensive toolset to enable domain experts to create sophisticated applications without writing code.
- Extensibility - NiagaraAX provides an extensible component model and rich library of open APIs to enable programmers to extend and enhance the Framework and build their own unique applications and products
- Support for embedded systems - NiagaraAX can be embedded in small low cost devices

The Niagara Framework solution

The following section describes how the Niagara framework addresses these challenges:

- [About control systems integration](#)
- [About Java](#)
- [About common networking and Internet protocols](#)
- [About programming for non-programmers](#)
- [About embedded systems capabilities](#)
- [About distributed systems capabilities](#)
- [About component software design](#)

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 Niagara software.

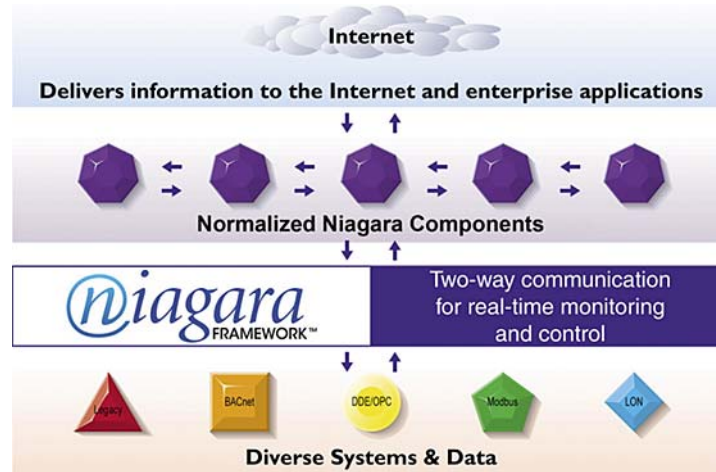
Niagara then models those devices and their data types in software through the common object model. This usually entails simplifying the device's data types to make them easier to manipulate and control through the software.

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 Niagara objects to build control system solutions quickly and efficiently.

The Niagara common object model, shown in [Figure 1-1](#), allows the Niagara Framework to:

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

Figure 1-1 Niagara common object model



About Java

All 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. NiagaraAX 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 for J2ME compliant VMs. The user interface toolkit and graphical programming tools are targeted for J2SE 1.4 VMs.

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. Niagara's design goal is to integrate cleanly with all networks and protocols. The Niagara Framework standardizes what's inside the box, not what the box talks to.

The Niagara software suite implements a highly efficient adaptation of the JavaBean component software model and Internet technologies to provide true interoperability across a wide range of automation products. The Niagara 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 (HTTP), 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 embedded systems capabilities

NiagaraAX 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 NiagaraAX-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

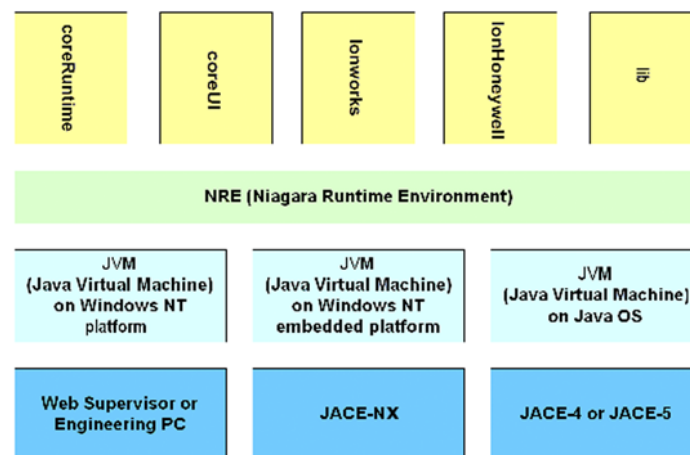
Niagara uses an architecture centered around the concept of “Component Oriented Development.” A *component* is a piece of self-describing software that can be assembled like building blocks to create new applications. A component–centric architecture solves many problems in Niagara:

- **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 Niagara applications.
- **Software reuse**
Components enable software reuse. NiagaraAX 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 Niagara software architecture

There are four layers to the Niagara software architecture. The bottom layer, as shown in [Figure 1-2](#) is the host platform, either a JACE controller or a PC. The next layer is the Java virtual machine (JVM). The JVM provides a layer between the hardware and its operating system and the Niagara software, known as the Niagara run-time environment (NRE). On top of the NRE are the Niagara modules.

Figure 1-2 NiagaraAX software architecture



NiagaraAX software subsystems are illustrated in [Figure 1-3](#) and the software processes and protocols are shown in [Figure 1-4](#).

Figure 1-3 NiagaraAX major software subsystems

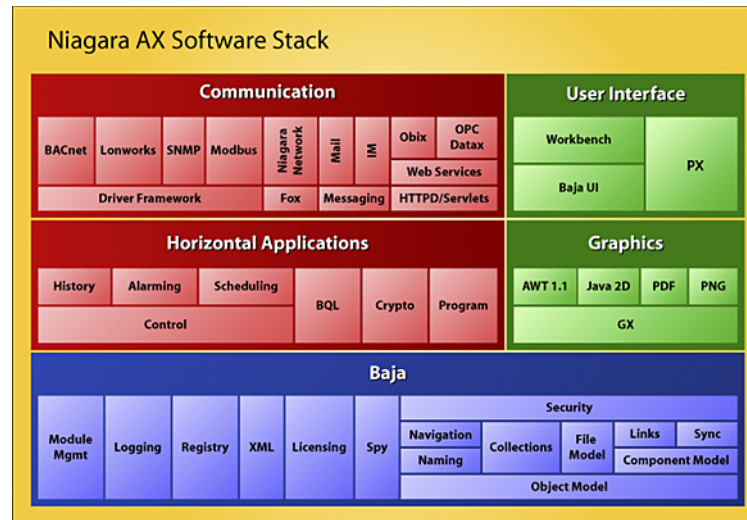
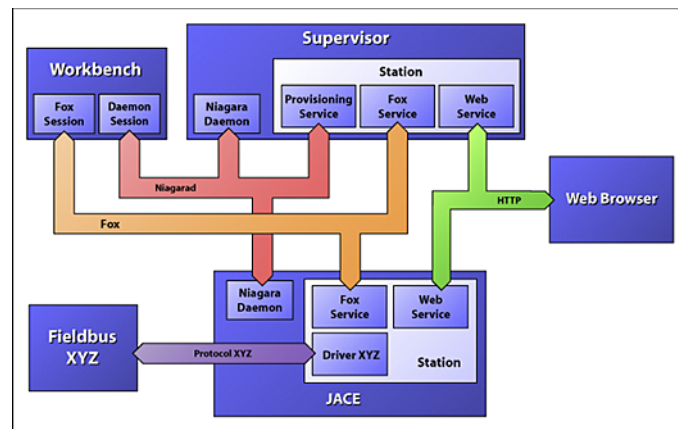


Figure 1-4 NiagaraAX software processes and protocols



About Baja

The core framework built by Tridium 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 Niagara Workbench, you can create and edit the control logic for your job site.

Many features found in Niagara 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 Niagara framework:

- [javax.baja](#)
- [com.tridium](#)

javax.baja

The APIs developed for Baja (see “[About Baja](#)” on page 1-5 for more about Baja) 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 Tridium feels 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 NiagaraAX Workbench. Concepts discussed in this section include the following:

- **Modules**
Modules are the most basic unit of the software that comprises NiagaraAX. For more details about modules, see “[About modules](#)” on page 1-6.
- **Components**
Components are the primary building blocks of the NiagaraAX framework. For more details about components, see “[About components](#)” on page 1-9.
- **Px**
For more details about Px, see “[About presentation xml \(Px\)](#)” on page 1-13.
- **stations**
For more details about stations, see “[About stations](#)” on page 1-15.
- **ORDs**
For more details about ORDs, see “[About ORDs](#)” on page 1-16.
- **Views**
For more details about views, see “[About views](#)” on page 1-19.

About modules

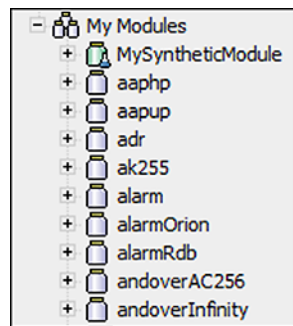
Modules are the smallest units of software in the Niagara architecture.

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 of Niagara, they may be distributed as independent revisions within that release.

Note: *Beginning in AX-3.7, two types of modules are possible in the Niagara framework, “standard” Java archive (.jar) modules which make up the Niagara software, and “synthetic Java archive” (.sjar) modules which allow 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.*

Figure 1-5 shows a partial list of modules, as displayed in the nav side bar pane.

Figure 1-5 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"](#) on page 1-9 for more information about components.

The following topics are covered in this section:

- [About module characteristics](#)
- [About benefits of modular software development](#)

About module characteristics

All (.jar) modules:

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

About jar files

JAR files are the mechanism for distributing Java modules. JAR stands for "Java archive." A JAR file (.jar) is basically a compressed package whose components can be viewed with WinZip 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.

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 "1.0.42". To understand which version of a module is more recent, simply observe the module version number. For example, 2.3.42 is later than 2.3.35 because $2.3.42 > 2.3.35$.

Figure 1-6 shows modules listed, by name, with the version number and installation status, as viewed in the Workbench software manager.

Figure 1-6 Module descriptions viewed in the software manager

File	Installed Version	Avail. Version	
aaphp	-	Tridium 3.7.38	Not Installed
aapup	-	Tridium 3.7.38	Not Installed
adr	Tridium 3.6.46	Tridium 3.6.46	Up to Date
ak255	-	Tridium 3.7.38	Not Installed
alarm	Tridium 3.7.38	Tridium 3.7.38	Up to Date
alarmOrion	-	Tridium 3.7.38	Not Installed

Niagara versioning uses the following set of conventions:

- Versions are specified using a base four-part version with optional patch versions.
- The first number specifies the specification revision. This number starts at one and is incremented each time the specification of the module is incremented. It is used with Baja modules to track the Baja specification version.
- The next two numbers specify a major Niagara release.

- The fourth number specifies a build number. A build number starts at zero for each major release and increments each time all the software's modules are built.
- Additional numbers may be specified for code changes made off a branch of a specific build. These are usually minor changes and bug fixes.

Every module has two versions. The first is the “bajaVersion” which maps the module to a Baja specification version. If the module is not published under the Baja process then this value is “0”. Secondly every module declares a “vendor” name and “vendorVersion”. The vendor name is a case insensitive identifier for the company who developed the module and the vendorVersion identifies the vendor's specific version of that module.

Tridium's vendorVersions are formatted as “major.minor.build[.patch]”:

- Major and minor declare a feature release such as 3.8.
- The third number specifies a build number. A build number starts at zero for each feature release and increments each time all the software modules are built.
- Addition numbers may be specified for code changes made off a branch of a specific build. These are usually patch builds for minor changes and bug fixes.

So the vendorVersion “3.7.22” represents a module of build 22 in Niagara release 3.7. The vendorVersion “3.7.45.2” is the second patch of build 45 in release 3.7.

About module directory structure

Every (.jar) module is managed in its own directory structure. [Figure 1-7](#) shows an example of a directory for the alarm module.

Figure 1-7 Module directory structure

```
[alarm]
+- build.xml
+- module-include.xml
+- module.palette
+- [src]
|   +- [javax]
|   |   +- [baja]
|   |   |   +- [alarm]
|   |   |   +- JavaClass1.java
|   |   |   +- JavaClass2.java
|   |   +- [com]
|   |   |   +- [tridium]
|   |   |   |   +- [alarm]
|   |   |   |   |   +- JavaClass3.java
|   |   |   |   |   +- [ui]
|   |   |   |   |   |   +- BAlarmConsole.java
|   +- [doc]
|   |   +- AlarmConsole-guide.html
+- [libJar]
```

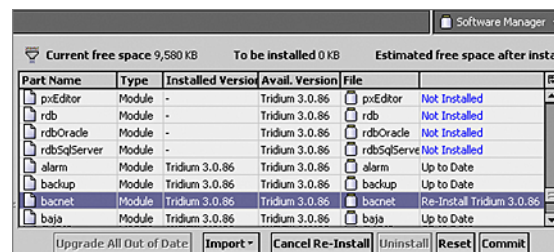
All of the source code that is used to build the module's jar file is located under the “src” directory. During the build process the “libJar” directory is used to build up the image, which will be zipped up for the module's jar file.

About 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 NiagaraAX Framework, it can be packaged, delivered, and added in as a single module or with multiple modules, using the NiagaraAX software manager, as shown in [Figure 1-8](#). Refer to the *Platform Guide* section “Software Manager” for more information about managing modules.

Figure 1-8 Software manager view

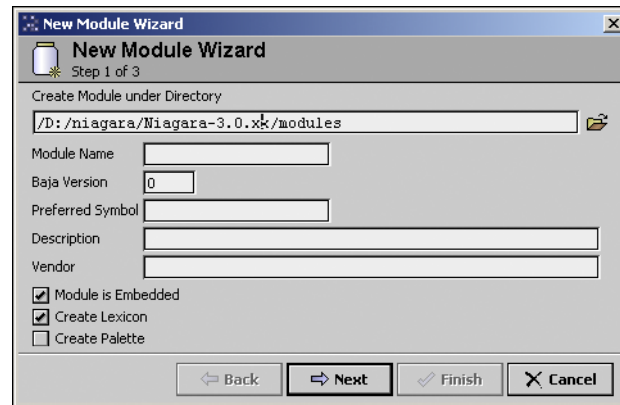


Part Name	Type	Installed Version	Avail. Version	File	
pxEditor	Module	-	Tridium 3.0.86	pxEditor	Not Installed
rdm	Module	-	Tridium 3.0.86	rdm	Not Installed
rdmOracle	Module	-	Tridium 3.0.86	rdmOracle	Not Installed
rdmSqServer	Module	-	Tridium 3.0.86	rdmSqServe	Not Installed
alarm	Module	Tridium 3.0.86	Tridium 3.0.86	alarm	Up to Date
backup	Module	Tridium 3.0.86	Tridium 3.0.86	backup	Up to Date
bacnet	Module	Tridium 3.0.86	Tridium 3.0.86	bacnet	Re-Install Tridium 3.0.86
baja	Module	Tridium 3.0.86	Tridium 3.0.86	baja	Up to Date

Buttons: Upgrade All Out of Date, Import, Cancel Re-Install, Uninstall, Reset, Commit

- **Requires less space on JACE**
Niagara’s modular development allows you to save space on your JACE by *only* installing the modules that you need and choose for that JACE.
- **Requires less space on the workstation**
When you install NiagaraAX, 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 with the Software Manager (as shown in [Figure 1-8](#)).
- **Create new modules**
Software developers can use the tools in NiagaraAX to create new (.jar) modules and deploy them using the **New Module Wizard**, as shown in [Figure 1-9](#). Refer to “[New Module wizard](#)” on page 8-10 for more details about the New Module Wizard.

Figure 1-9 New module wizard



Note: Starting in AX-3.7, you can also create “synthetic modules,” which does not require developer level knowledge. 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 NiagaraAX Workbench. As described in the “[About component software design](#)” on page 1-4, 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.

The following sections describe characteristics of components:

- [About slots](#)
- [About master/slave components](#)
- [About point components](#)

About slots

Niagara components are defined as a collection of “slots.” You can see all the slots that make up a component by viewing the “slot sheet” as shown in [Figure 1-10](#). Slots have the following types of attributes:

- [Slot type](#)
- [Slot name](#)
- [Slot definition](#)
- [Flags](#)
- [Facets](#)

Figure 1-10 Slot sheet for a single component

Slot	#	Name	Display Name	Definition	Flags	Type	Facets	
Property	0	code	Code	Frozen		program:ProgramCode		
Action	1	execute	Execute	Frozen	a	void (void)		
Property	2	BooleanArg	BooleanArg	Dynamic	rs	baja:Boolean		
Property	3	StringArg	StringArg	Dynamic	rs	baja:String		
Property	4	MonthArg	MonthArg	Dynamic	rs	baja:Month		
Property	5	wsAnnotation	wsAnnotation	Dynamic		baja:WsAnnotation		
Action	6	Active	Active	Dynamic		void (void)		
Action	7	Inactive	Inactive	Dynamic		void (void)		
Action	8	ChangeString	ChangeString	Dynamic		void (baja:String)		
Action	9	ChangeMonth	ChangeMonth	Dynamic		void (baja:Month)		

Slot type

There are three types of slots:

- **Property**
Property slots represent a storage location of another Niagara 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 issued by right-clicking on the component.
- **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 a event source.

Slot name

Every slot is identified by a 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 Niagara Framework is in providing a consistent model for both frozen (compile time) slots and dynamic (runtime) slots.

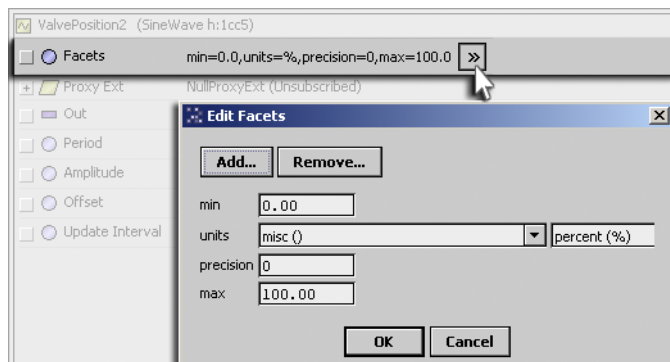
Flags

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 meta data – or additional data 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, as shown in [Figure 1-11](#).

Figure 1-11 Editing facets from the property sheet



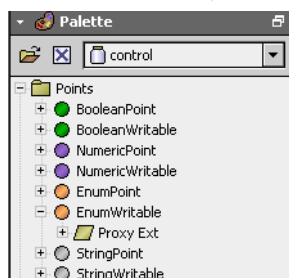
About master/slave components

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

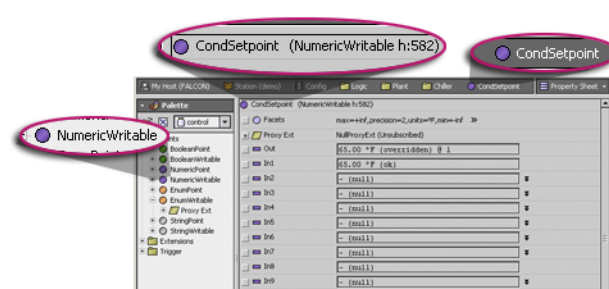
In any NiagaraAX station, all real-time data is *normalized* within the station database as *points*, a special group of components. Figure 1-12 shows several types of control points, as listed in the control module palette in Workbench.

Figure 1-12 Point types



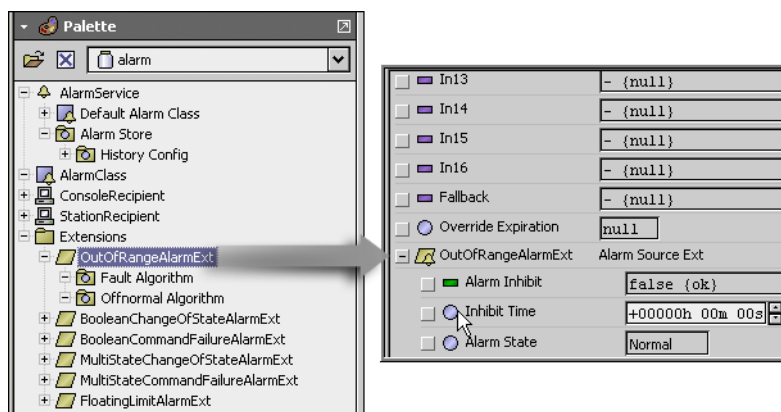
Each type of point may be used for different purposes in Workbench. When you engineer a job you may want to name a point, as shown in Figure 1-13 (a NumericWritable point named CondSetpoint). Points may be named and renamed but they retain their initial point type characteristics and their characteristic icon color.

Figure 1-13 Numeric writable point named



Points serve as a type of shell in NiagaraAX, 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.

Figure 1-14 Adding point extensions



For detailed information about points and point extensions, refer to “Data and Control Model” on page 3-1.

About component naming

In a NiagaraAX 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 next section “[Escaped 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, as needed. Examples of such component names are:

- `Floor1` OR `Floor_1`
- `ReturnAirTemp` OR `Return_Air_Temp`
- `Zone201_SAT` OR `Zone_201_SAT`

Escaped names

Workbench *does allow* you to name components “improperly,” such as with spaces or other non-alphanumeric characters, without any warning. Further, various NiagaraAX 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 “[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 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 palette nav side bar pane. You can also create custom palettes that you associate with a module and use for holding frequently used components. For more information about using the palette side bar, refer to “[About the palette side bar](#)” on page 2-8.

About presentation

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

Presentation is a term that is used to describe how Niagara 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**
 - bitmap and vector graphics
 - text documents
 - tables
 - charts
 - input controls (text fields, check boxes, trees)
- **media**
 - desktop web browsers (HTML, CSS, JavaScript)
 - workbench (Niagara stack)
 - pdf
 - printed pages
 - svg
 - handhelds
 - cell phones

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.
- **bajau**

The bajau module provides the widget toolkit. Widgets are the basis for graphical composition, layout, user input, and data binding. The bajau module builds upon gx to paint the widgets.
- **PDF**

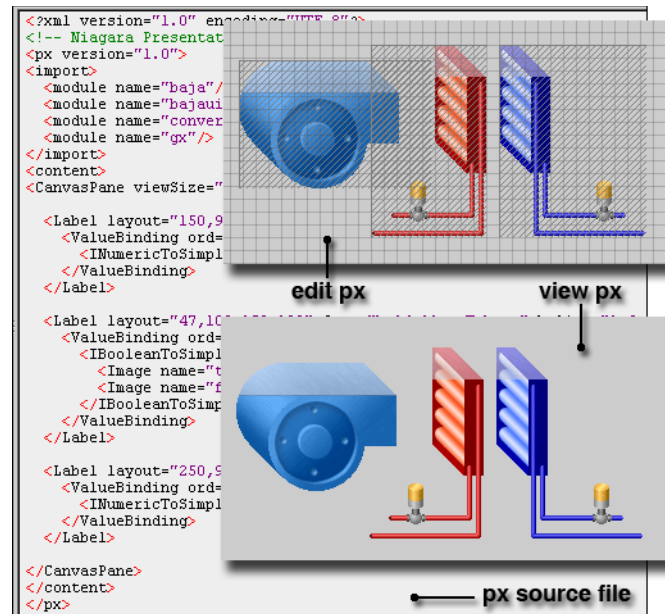
The PDF implementation of the gx APIs.
- **HTML**

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. [Figure 1-15](#) shows an example of a Px file in the Text Editor (Px source file), the Px Editor view, and as displayed in the Px Viewer.

Figure 1-15 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.

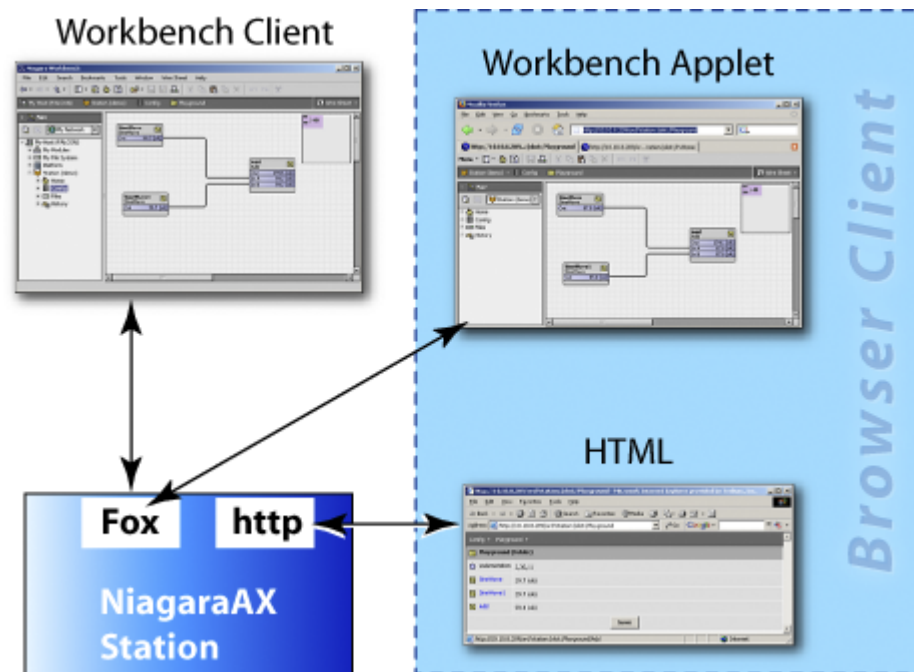
About presentation media

There are four primary target medias that NiagaraAX supports.

- **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 Niagara presentations with their full power. With the WbProfile feature, new custom applications can easily be built using Workbench framework and its presentation engine.
- **Workbench applet**
Workbench applet allows the full Niagara stack to be downloaded to a client machine using a web browser with the Java plug-in. Once loaded to the client, Workbench may be run 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 be printed to paper. 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.
- **HTML**
For the casual user where installing the Java plug-in or downloading Workbench is impractical, Niagara supports an engine for generating plain HTML pages from Px. A simple set of HTML, CSS, and

JavaScript provide a web interface using common standards. This interface may also be used to support handheld devices and cell phones. Designers are limited to a subset of widgets when creating HTML presentations (the Px tools provide this support). Refer to the *NiagaraAX Graphics Guide* section “Types of Px target media options” for information about presentation media technologies.

Figure 1-16 Presentation media options



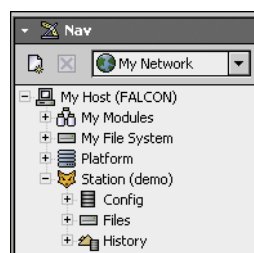
About stations

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 the 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 1-17 Station in nav tree

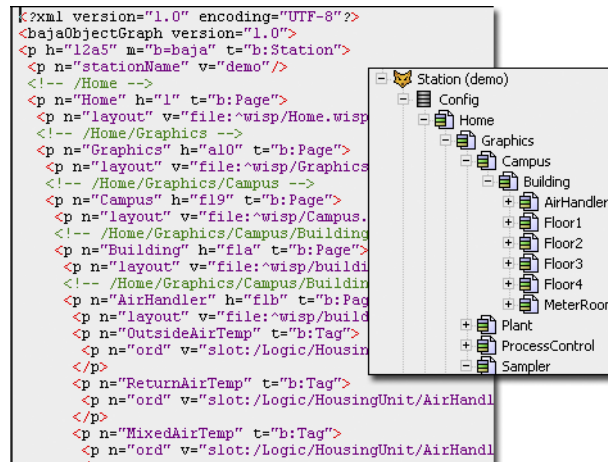


A system can be a single station or multiple stations depending on the size of the project and it is defined by a bog file. See “About BOG files” on page 1-16.

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 1-18 Sample bog file and nav tree presentation

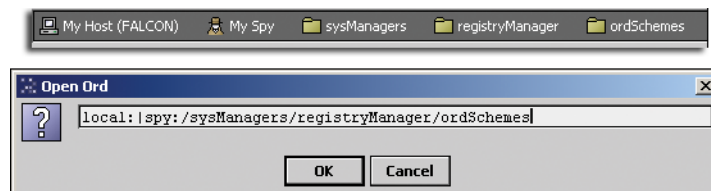


About ORDs

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 (see [Figure 1-19](#)).

Figure 1-19 Open ORD and graphic locator system



ORDs can be relative or absolute. An absolute ORD usually takes the general format of “host|session|space”, as illustrated in [Figure 1-20](#).

Figure 1-20 Absolute ORD typical structure

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

- **host**
The host query identifies a machine – usually by an IP address such as “ip:hostname”. For example “fox:” indicates a fox session to the host.
- **session**
The session is used to identify a protocol being used to communicate with the host.
- **space**
The space query is used to identify 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.

Examples of ORDs:

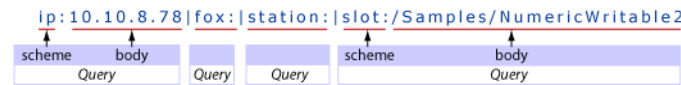
- ip:somehost|fox:|station:|slot:/MyService
- ip:somehost|fox:|station:|h:/42
- ip:somehost|fox:|file:/C:/dir/file.txt
- local:|file:lib/log.properties
- local:|module://icons/x16/cloud.png
- local:|spy:/

In Niagara you may view the complete list of installed ORD schemes at “spy:/sysManagers/registry-Manager/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 1-21 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 (see Figure 1-20).

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 BLocal-Host.INSTANCE is both a host scheme and a session scheme. It represents objects found within the local VM.

Types of schemes

- **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: “!lib/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

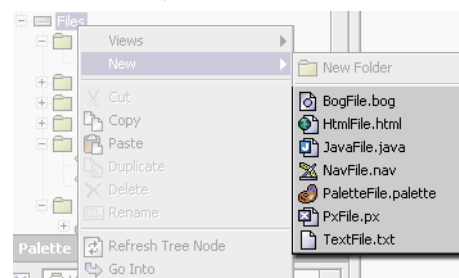
Space defines a group of objects that share common strategies for loading, caching, lifecycle, naming, and navigation. Following, is a list of some of the different types of space:

- **component**
- **file**
- **history**
- **module**
- **orion (starting with NiagaraAX-3.4)**
- **station**
- **view**

Types of files

In the file system, you may create and edit various types of files, as shown in [Figure 1-22](#). Following, is a list of some of the different types of files that reside in the file space:

Figure 1-22 Types of files available from the **new file** menu



- **BogFile.bog**
bog files are database files. Refer to “About BOG files” on page 1-16 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.
- **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” on page 9-11 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)” on page 1-13 for more information about Px files.
- **TextFile.txt**
Text files are edited and viewed in the text file editor.
- **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”.

About file naming

When working in a station, you often create files of various types in its file space—for example, a Px file if adding a new view, if creating a new Nav 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

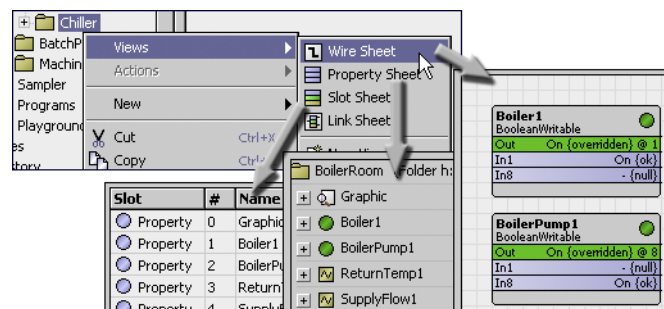
specials= space | . | : | - | _ | \$ | + | (|) | & | ' | ' | @ | [|]

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). Note that earlier revisions of NiagaraAX automatically restricted file names in dialogs to only the original set of characters above. However, starting in NiagaraAX 3.6.47 and later, these file naming rules were “relaxed” to deal with other issues relating to Web Workbench and locally cached files.

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 side bar nav tree. 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 side bar tree may have a wire sheet view, a property sheet view, and a slot sheet view, that all display in the view pane as shown in Figure 1-23. Each component has a **default view** that appears whenever you activate a component (double-clicking, for example) without specifying a particular view.

Figure 1-23 Views



See “About the view pane” on page 2-12 for more information on the view pane. Refer to the “Plugin Guides” on page 11-1 for a comprehensive list of views.

About lexicons

NiagaraAX provides *non-English* language support by use of *lexicons*. Lexicons are identified by Java *locale codes*, such as “fr” (French) or “de” (German). Starting in Workbench 3.7, all of the lexicons are distributed as *modules* (niagaraLexiconXx.jar) included in the Workbench installation.

Each lexicon module contains one or more lexicon files (*moduleName.lexicon*). Lexicon files are simple text files that map various entity “keys” (such as interface names or error and status values) to localized language character values.

Note: *Although module-based lexicons are introduced in Workbench 3.7, file-based lexicons continue to be supported. If you have used file-based custom lexicons in a prior release, you can continue to use those. In Workbench 3.7, you can deploy custom lexicons in any file-based/module-based combination.*

To edit lexicons, you typically use the Workbench lexicon tool (**Tools > Lexicon Tool**). “Lexicon Report”, the default view, lists all lexicons installed on your Workbench PC and specifies details about a selected lexicon and its contained files. The “Lexicon Editor” view provides edit features and shows the default (English) value for each line entry. The “Lexicon Module Builder” view lets you bundle multiple lexicon files in a module for ease of distribution.

Note: *In some cases, lexicon editing provides utility even in English language scenarios. For more details, see the **Lexicon Guide** section on “Notes on English (en) lexicon usage”.*

After editing a lexicon file, you can then install the file in a remote JACE using the platform Lexicon Installer view. An alternative is to create (or update) a lexicon module with the edited lexicon file, using the new Lexicon Module Builder view. Install the new (or updated) module in a remote JACE using the platform Software Manager view.

For detailed information on using the Lexicon Tool, see the *NiagaraAX Lexicon Guide*.

CHAPTER 2

About Workbench

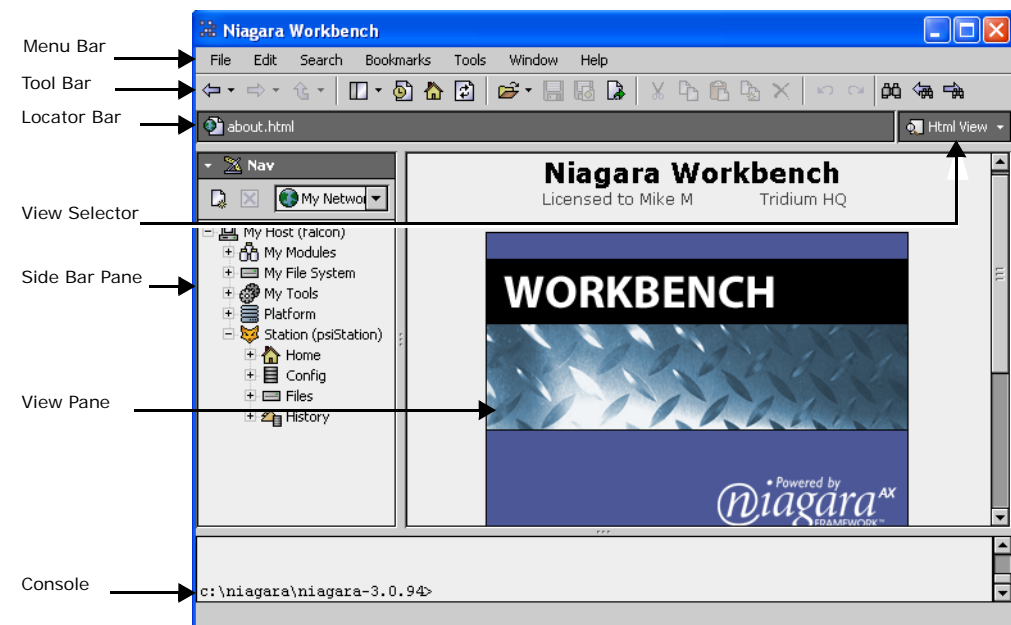
Workbench is the term for the NiagaraAX graphic user interface. The following sections describe the general layout and many of the features of the Workbench user interface. Refer to the following sections for more information:

- [Tour of the Workbench GUI](#)
- [Customizing the Workbench environment](#)

Tour of the Workbench GUI

When you start Workbench, you will see the Workbench screen, as shown in [Figure 2-1](#).

Figure 2-1 Niagara AX Workbench



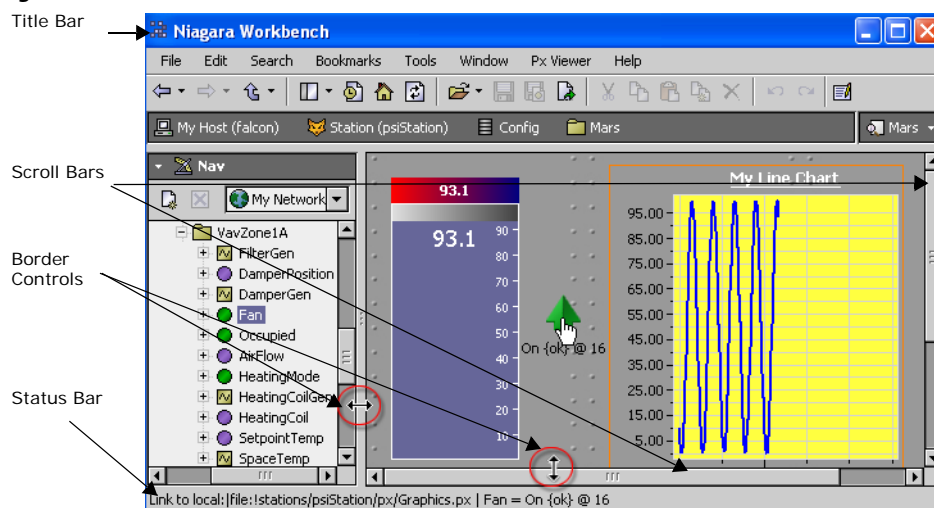
The primary window of the Workbench GUI is divided into three main areas:

- **Side bar pane**
Top-left area displays one or more side bars that you may choose from the **Windows** menu. For details, see [“About the side bar panes”](#) on page 2-2.
- **View pane**
Top-right area displays the currently selected view. For details, see [“About the view pane”](#) on page 2-12.
- **Console**
Bottom area displays a command line that allows you to issue certain commands directly to the system. For details, see [“About the console”](#) on page 2-13.

About Workbench window controls

The Workbench window provides typical Windows-type controls plus other features unique to Niagara, as shown in [Figure 2-2](#).

Figure 2-2 Window controls



You may create additional windows after starting Workbench—all have these basic features:

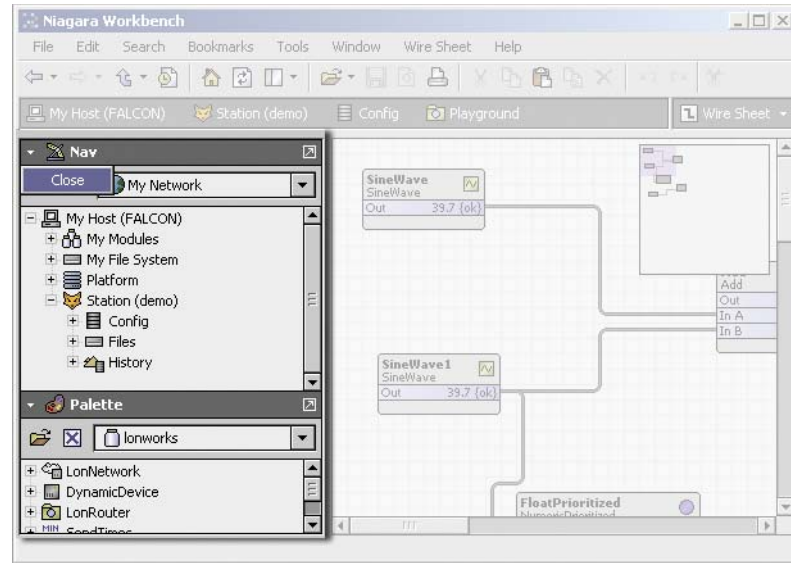
- **Title bar**
Has standard Windows title bar features, including windows name and icons to minimize, maximize, and close. Double-click the title bar to toggle between maximized and a sizable window.
- **Border Controls**
As needed, drag any outside border to resize the entire window. Drag the inside border between the side bar and view areas, or (if shown) the console area to change their relative sizes.
- **Scroll bars**
Scroll bars appear in window areas when some content portions are not visible. They are along the right and/or bottom portions of a window area (side bar, view, or console). Simply drag a scroll slider (highlighted area) to scroll quickly. Or, click an ending scroll arrow to move in small increments.
- **Status bar**
At the bottom left of the Workbench window is a status bar.
The status bar displays meanings of toolbar buttons. When working in a Wire Sheet view, other details are revealed in the status line.

About the side bar panes

Side bar panes are normally visible only if a side bar is open. When you close all side bars, the side bar pane will collapse and the view pane and console pane (if open) will expand to fill the window. 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, as shown in Figure 2-3.
- **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.

Figure 2-3 Side bar pane



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 title bar to open, close, and resize the side bars.

- For more information about the side bar title bar, see [“About the side bar title bar”](#) on page 2-3.
- For more information about types of side bars see [“Types of side bars”](#) on page 2-3.

About side bars

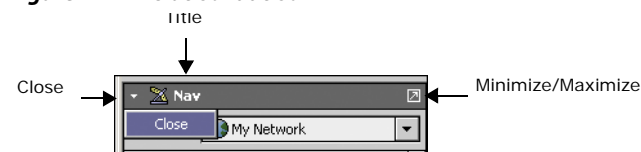
All side bars display in the side bar pane and have some common features, such as the side bar title bar. The following sections describe characteristics of each type of side bar:

- [About the side bar title bar](#)
- [Types of side bars](#)

About the side bar title bar

All side bars have a title bar across the top as shown in [Figure 2-4](#). 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.

Figure 2-4 Side bar title bar



In addition, all side bars have the following controls:

- Close menu
Drop down control used to close the side bar pane.
- Pane title
Icon and text that identifies the side bar.
- Maximize button
When more than one side bar is open, this button appears on the right side of any non-maximized side bar. Clicking on the maximize button will expand the side bar to fill the side bar pane and the restore button will change to the maximize button.
- Restore button
If the side bar pane is at maximum size, the restore button appears on the top right corner of the title bar. Clicking on this button will restore the side bar pane to its previous size and the maximize button will change to the restore button.

To find out more about specific side bar panes and their controls see [“Types of side bars”](#).

Types of side bars

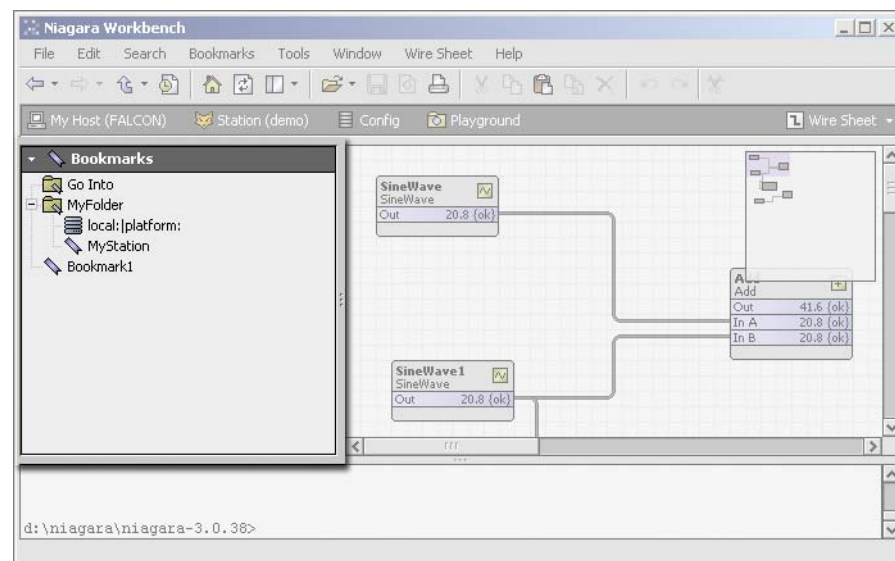
The workbench interface may be customized by adding unique side bars that are designed to fit particular applications. The following side bars may be displayed in the side bar pane by default:

- **Bookmarks side bar**
Displays a list of bookmarks. For details, see [“About the bookmark side bar”](#) on page 2-4.
- **Help side bar**
Provides a tree view of available help documentation. For details, see [“About the help side bar”](#) on page 2-5.
- **Jobs side bar**
Provides a tree view of available help documentation. For details, see [“About Jobs side bar”](#) on page 2-5.
- **Navigator side bar**
Provides a tree view of the system. For details, see [“About the nav side bar”](#) on page 2-6. For more information about the nav side bar, see [“Types of nodes in the nav side bar tree”](#) on page 2-6, and [“About the nav side bar toolbar”](#) on page 2-7.
- **Palette side bar**
Provides a tree view of components that are available in specific palettes. For details, see [“About the palette side bar”](#) on page 2-8 and [“About the palette side bar toolbar”](#) on page 2-8.
- **Todo side bar**
Provides a customizable list of tasks or notes. For details, see [“About the Todo list sidebar”](#) on page 2-9.
- **Bound ords side bar**
Provides a listing of all bound ords used in a Px file. For details, see [“About the bound ords side bar”](#) on page 2-10.
- **Widget tree side bar**
Provides a tree view of all widgets used in a Px file. For details, see [“About the widget tree side bar”](#) on page 2-10.
- **Px Properties side bar**
Provides a listing of all Px Properties that are defined for the viewed Px page. For details, see [“About the Px properties side bar”](#) on page 2-11.
- **Properties side bar**
Provides a listing of all properties for a component that is selected in the Px Editor. For details, see [“About the properties side bar”](#) on page 2-11.

About the bookmark side bar

When you open the bookmark side bar, it appears in the side bar pane, as shown in [Figure 2-5](#).

Figure 2-5 Bookmark 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”](#)

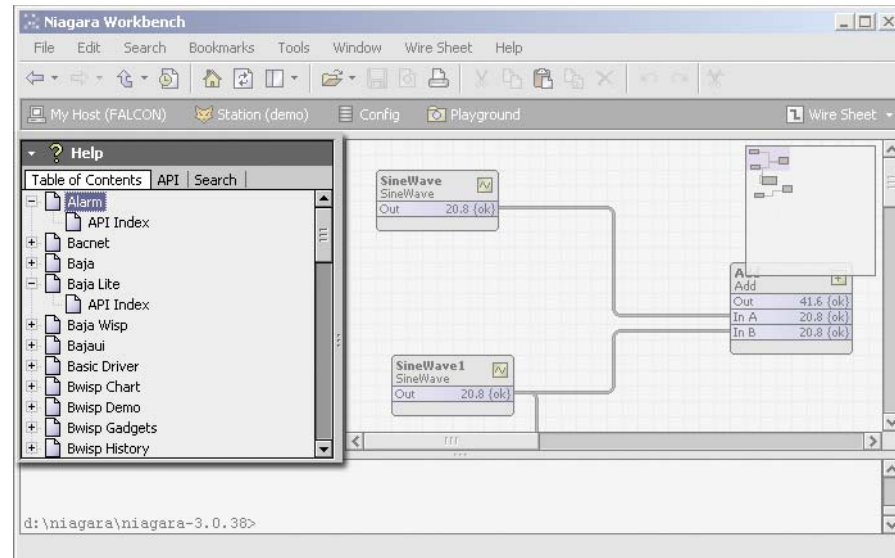
on page 9-8.

- For details about managing bookmarks, see [“To manage bookmarks”](#) on page 9-8
- For details about editing bookmarks, see [“To edit a bookmark”](#) on page 9-8

About the help side bar

When you open the help side bar, it appears in the side bar pane, as shown in [Figure 2-6](#).

Figure 2-6 Help side bar



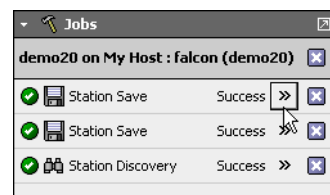
The help side bar has three tabs that you may select by clicking on the tab. The three help tabs are listed and briefly described, as follows:

- **Table of Contents**
Contains a tree view of help topics, listed in alphabetical order by topic.
- **API**
Contains a tree view of help topics, listed in alphabetical order by module.
- **Search**
Contains a **Find**: text entry field and **Search** button. For details on using the Search, see [“Using the help side bar”](#) on page 9-8.

About Jobs side bar

When you open the Jobs side bar, it appears in the side bar pane as shown in [Figure 2-7](#). The Jobs side bar contains a list of jobs that have been performed or that are currently being performed.

Figure 2-7 Jobs side bar

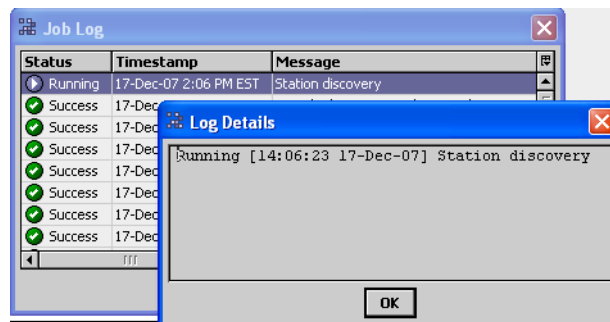


The following icons on the Jobs side bar indicate job status:

- **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 box, as shown in [Figure 2-8](#).

Figure 2-8 Log Details dialog box

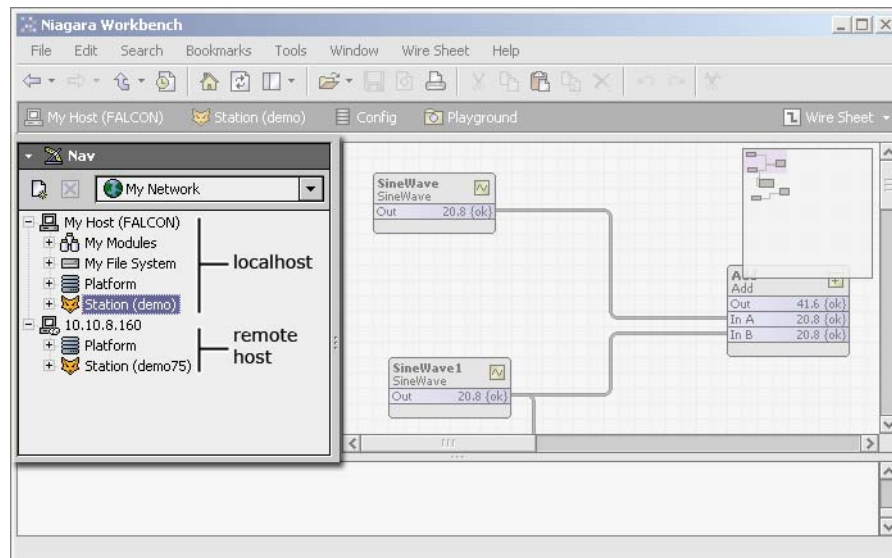


About the nav side bar

When you open the nav side bar, it appears in the side bar pane, as shown in [Figure 2-9](#). The nav side bar contains the tree view that provides a hierarchical view of the whole system. From the nav side bar, you can double click on nodes in the nav tree or use popup menus to perform all operations that are available from the nav side bar (for example, connect or disconnect to a station, refresh a tree node, and more). The expandable tree provided here is very useful for performing actions on nodes and for navigating through various screens and views in the Workbench. Items are displayed in the tree with an icon that represents the associated function or file type.

Items are displayed in the tree with a symbol based on type. If the item is a file, it will be based on file type. Refer to “[Types of nodes in the nav side bar tree](#)” on page 2-6 for more details about file types. Refer to “[About the nav side bar popup menu items](#)” on page A-9 for more details about the nav side bar popup menu.

Figure 2-9 Nav side bar



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

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

Types of nodes in the nav side bar tree

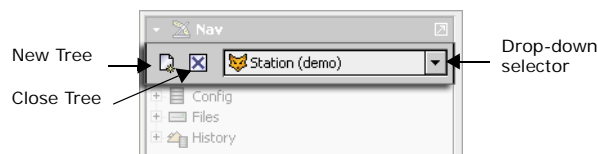
The nav side bar tree may include the following types of nodes and their child nodes:

- **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”](#) on page 1-4.
- **Module node**
When expanded, displays a tree view of available modules, listed in alphabetical order by module. For more details on modules, see [“About modules”](#) on page 1-6.
- **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.
- **Station node**
Represents a station (connected or disconnected). When expanded, the station node displays the station contents in a hierarchical tree. For more details on stations, see [“About stations”](#) on page 1-15.
- **Platform**
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**
When connected and expanded, displays a hierarchical view of the Niagara station. 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). For details on the station node, refer to [“About stations”](#) on page 1-15.
- **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:
 - **Component**
When expanded, components (as containers) display a tree view of the sub-components that they contain. Various component types may be displayed either in containers or at the root of the component node. For more details on components, see [“About components”](#) on page 1-9.
 - **Control**
Control points may be displayed directly in the root of the Config node. For more information about Control points, see [“About point components”](#) on page 1-11 and [“Data and Control Model”](#) on page 3-1.
 - **Drivers**
Provides a place to store driver modules (such as the Niagara Network, 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*.
 - **Services**
Component for storing services, such as alarm service, history service, program service, and more.

About the nav side bar toolbar

In addition to the standard side bar title bar (see [“About the side bar title bar”](#) on page 2-3 for more details) the nav side bar has a toolbar, located just below the title bar as shown in [Figure 2-10](#).

Figure 2-10 Nav side bar toolbar



The nav side bar toolbar includes the following:

- **New tree button**
When clicked, 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 Drop-down tree selector.
- **Close tree button**
When clicked, closes the currently displayed side bar.

- **Drop-down tree selector**

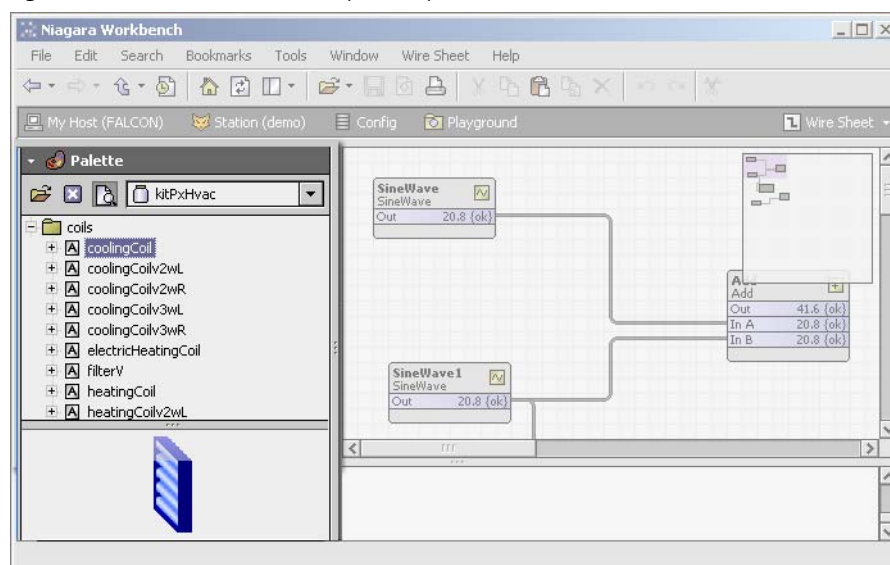
When more than one nav side bar is open, this selector allows you to choose which one to display

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, as shown in [Figure 2-11](#). The palette side bar provides a place to open and view sets of modules or custom palettes that you build for yourself. From the palette side bar, you can open multiple palettes (displaying them one at a time), 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 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 that displays an image (when available) of the selected component (shown below).

Figure 2-11 Palette side bar with 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 `compPreviewWidget` property. If no preview is associated with a component, in AX-3.5 and later, you can add a `compPreviewWidget` 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”](#) on page 9-12.

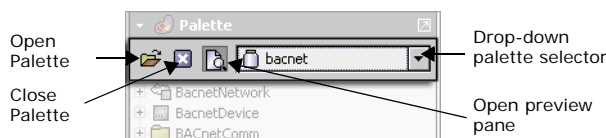
For more details about the palette side bar, refer to the following:

- To find out how to use the palette side bar, see [“Using the palette side bar”](#) on page 9-10.
- For details about adding palettes to the palette side bar, see [“To open a palette”](#) on page 9-10

About the palette side bar toolbar

In addition to the standard side bar title bar (see [“About the side bar title bar”](#) on page 2-3 for more details) the palette side bar has a toolbar, located just below the title bar as shown in [Figure 2-12](#).

Figure 2-12 Palette side bar tool bar



The palette toolbar includes the following:

- **Open palette button**

When clicked, opens the **Open Palette** dialog box. Refer to [“To open a palette”](#) on page 9-10 for information about opening palettes.

- **Close palette button**

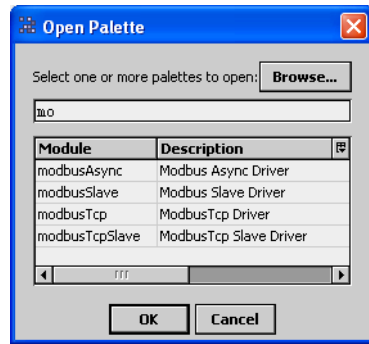
When clicked, closes the currently displayed palette.

- **Drop-down palette selector**
When palettes are open in the palette side bar, this selector allows you to choose which palette to display.
- **Open preview pane button** (AX-3.5 and later)
This control button toggles the preview pane on and off. Previews are available on some components.

About the Open Palette dialog box

The **Open Palette** dialog box 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 2-13 Open Palette dialog box



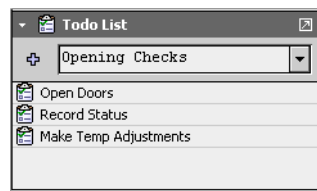
The **Open Palette** dialog box is shown above and has the following features:

- **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** dialog box 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 Todo list sidebar

The Todo list sidebar is a convenient way to create and access Todo List items from a palette. Click the + button on the toolbar to open the **Add** dialog box. The **Add** dialog box provides a text field for adding and categorizing Todo list items. For more details about the Todo List see [“Todo List”](#) on page 8-15.

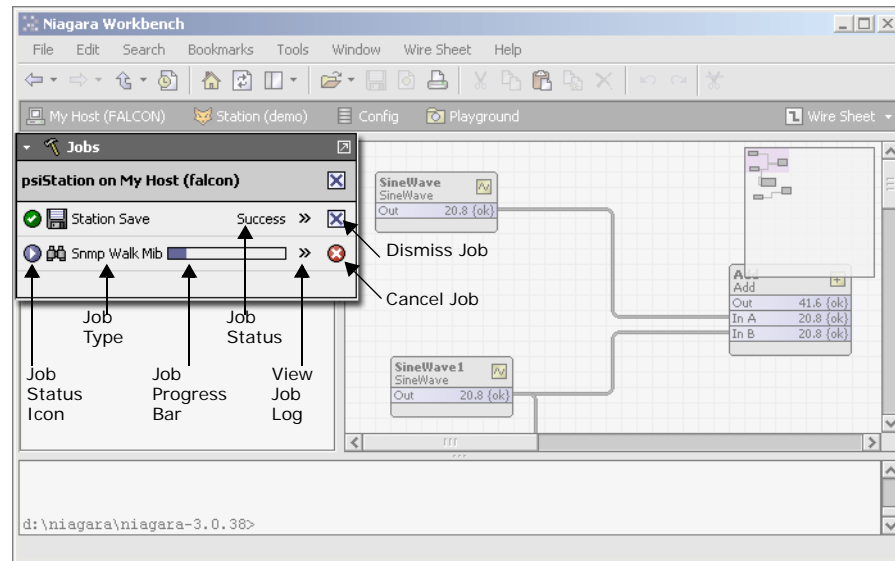
Figure 2-14 Todo bar



About the jobs side bar

The Jobs side bar shows all the current jobs in all the stations with which you have a connection. The current status of each job is shown as: running, canceling, canceled, success, or failed. If the job is running then a progress bar displays estimated progress. You may cancel a running job by pressing the **Cancel** button. Normally, once a job has completed you are notified via the async notification feature. You may then dismiss the job by pressing the **Close** button. The details of the job may be accessed using the “>>” button to display the **Job Log** dialog box. For details about using the job side bar, refer to [“Using the jobs side bar”](#) on page 9-10.

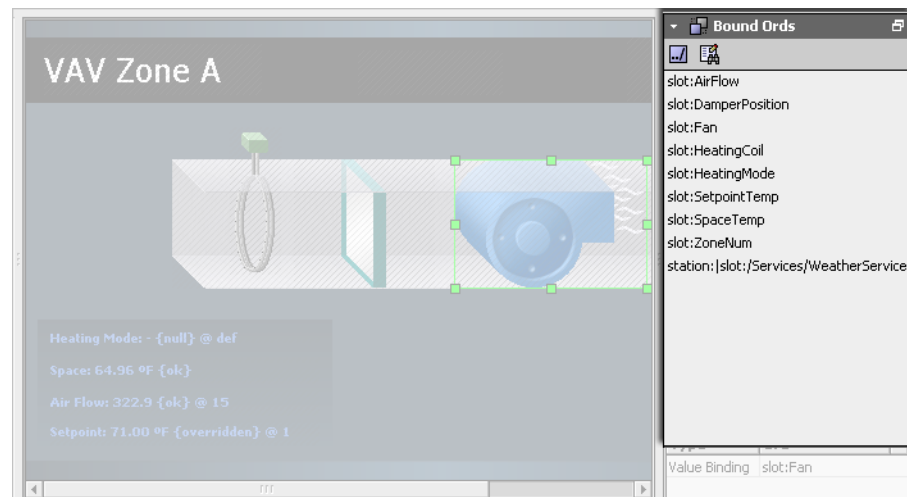
Figure 2-15 Jobs side bar



About the bound ords side bar

The bound ords side bar, in [Figure 2-16](#), is available when the Px editor view is active. It displays a listing of all the bound ords in the current Px view. Double click on any ORD in the list to display the ORD in the ORD editor dialog box.

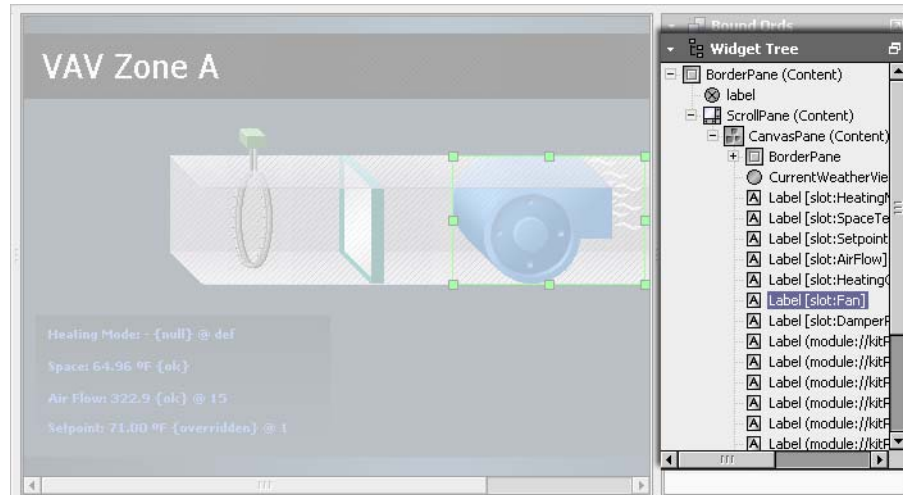
Figure 2-16 Bound ords side bar



About the widget tree side bar

The widget tree, in [Figure 2-17](#), displays a tree hierarchy of the widgets (panes, labels, graphic elements, and so on) that are in the current Px view.

Figure 2-17 Widget tree side bar

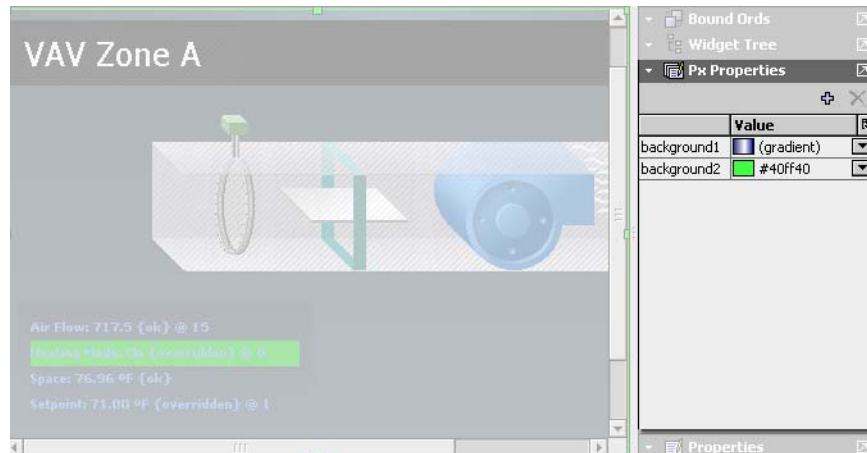


Note: It is often easier to use the Widget Tree to select objects when you have a lot of objects on a view—especially when there are several layers of objects. When you select an object in the tree view it is selected in the Px view as well and displays the selection borders and handles.

About the Px properties side bar

Starting in NiagaraAX-3.3, the Px properties side bar, shown in [Figure 2-18](#), 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. Use the menu bar icons to add, define, assign, and delete Px properties. For more information about px Properties, see the *NiagaraAX Graphics Guide* section “About Px Properties”.

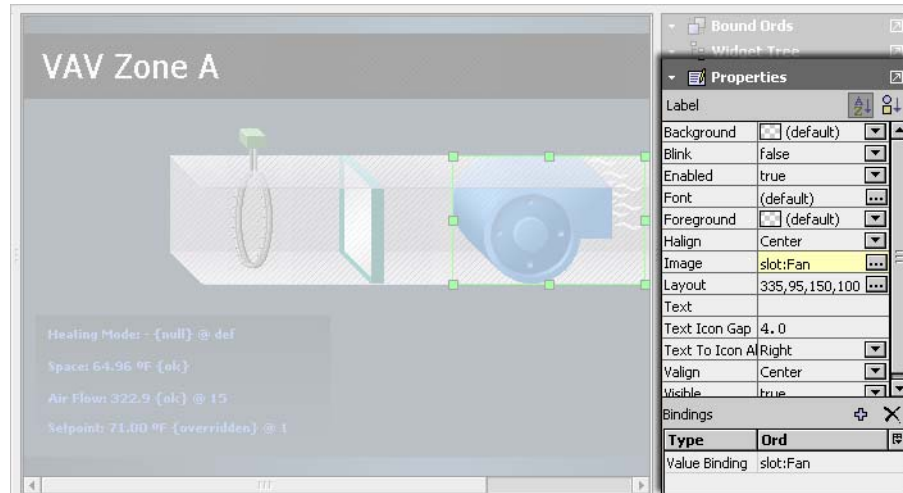
Figure 2-18 Px properties side bar



About the properties side bar

The properties side bar, shown in [Figure 2-19](#), 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. Double click on any object in the widget tree or in the in the Px viewer to display the **properties** dialog box (same information as the properties side bar).

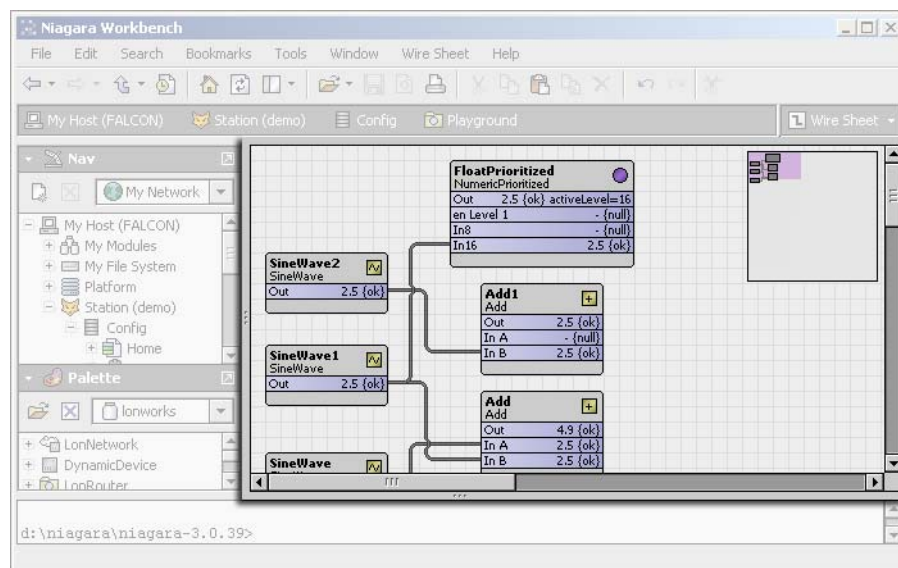
Figure 2-19 Properties side bar



About the view pane

The View pane is the largest window area and is located on the right side, below the locator bar, as shown in [Figure 2-20](#). Features of the view pane include tabbed views and a thumbnail view.

Figure 2-20 View pane



Multiple tabbed views may be added to the view pane by using the tab feature. You can add, close, and select tabs in the view window. The view area displays the currently selected view for the active tab. You can change the selected view by doing any of the following:

- double-click on an item in the nav palette tree
- select a view or action from the nav palette popup menu
- select a command from a menu or submenu
- select a command from the locator bar
- select a view from the view selector or from the view popup menu

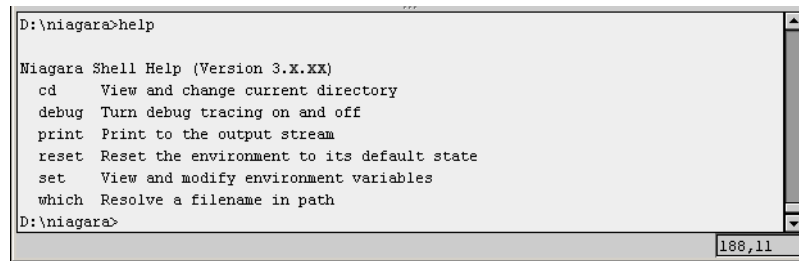
The thumbnail view, when active, appears in the top right corner of the window to help you find your way around the wire sheet. For more details about the thumbnail view, refer to [“Show Thumbnail”](#) on page 11-16.

For more details about using tabs, refer to [“Creating tabs in the view pane”](#) on page 2-21.

About the console

The console, see [Figure 2-21](#), is located along the bottom of the Workbench interface and may be alternatively hidden or shown by selecting **Window > Hide Console** or **Window > Console** from the menu bar.

Figure 2-21 Niagara AX console



The console has scroll bars on the right side and the window size may be adjusted by dragging the top border bar. The console provides you with access to a command line prompt without leaving the Workbench environment. From the console you may type in commands directly, including the help command for additional help. The following lines are some console level commands, parameters, and options:

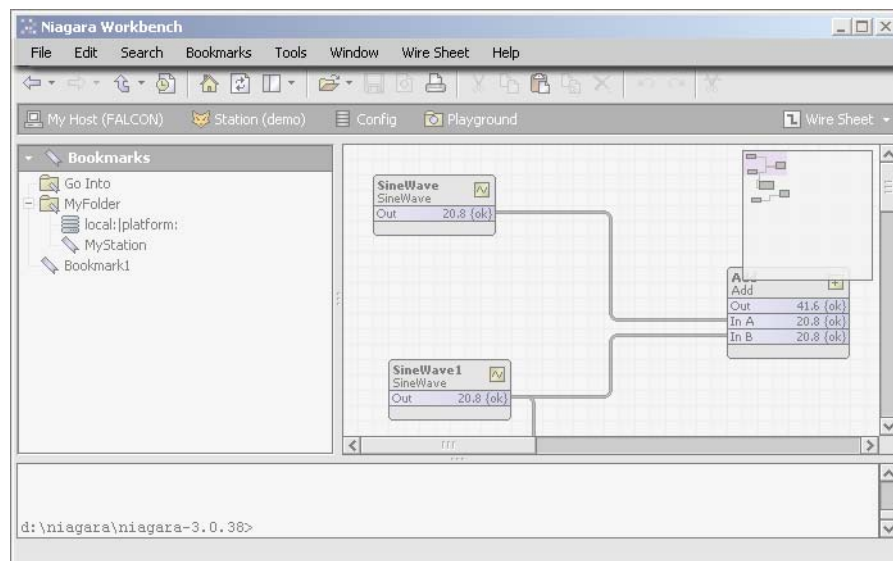
```
>wb -help

usage:
  wb [options] <ord>
parameters:
  ord          initial ord to display
options:
  -profile:<type> workbench:WbProfile to use
  -locale:<x>    set the default locale (en_US)
  -@<option>    pass option to Java VM
```

About the menu bar

The menu bar is the row of pulldown menus located along the very top of the Workbench GUI, as shown in [Figure 2-22](#). You may use these menus to perform operations throughout the Workbench GUI. Many of the menus on the menu bar are context-sensitive and only appear when certain views are active.

Figure 2-22 Menu bar

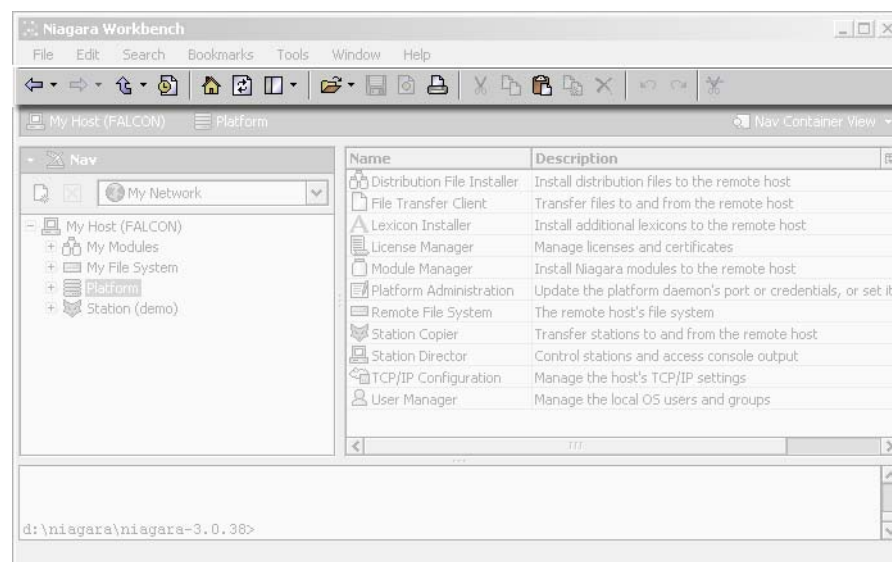


For detailed information about the menus, refer to [“Types of menu bar items”](#) on page A-2.

About the toolbar

The toolbar is the row of icons, just below the menu bar, as shown in [Figure 2-23](#). The toolbar provides menu choices for 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 2-23 Toolbar

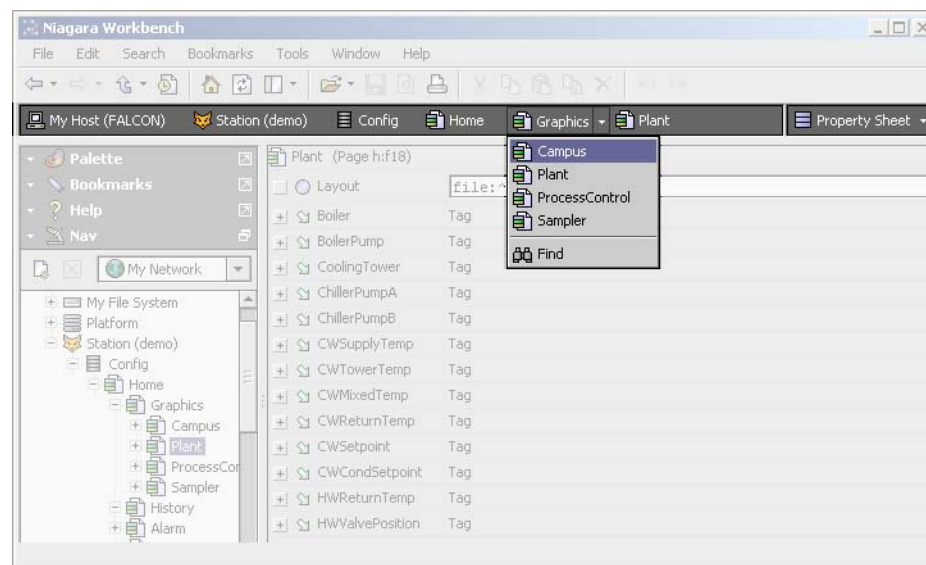


In addition to the primary (always visible) toolbar, additional sets of toolbar icons are added to the toolbar when different views are active. For example, when the Wire Sheet view is active, the Delete Links icon and Zoom icons are available. When an icon is dimmed, it is unavailable. For more information about specific toolbar icons, refer to “Types of toolbar icons” on page A-13.

About the locator bar

The locator bar is the area just below the toolbar that spans horizontally from the left edge of the Workbench GUI all the way to the right side, including the view selector on the right side, as shown in [Figure 2-24](#). The part of the locator bar that shows your current path (not including the view selector) is referred to as the PathBar.

Figure 2-24 Locator bar



The purpose of the locator bar is to provide a graphical navigation field for selecting and displaying destination references. The locator serves two functions.

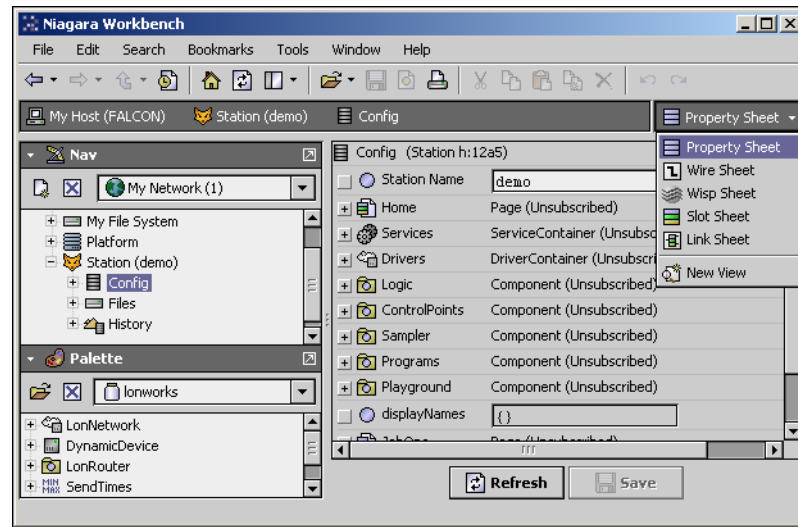
- First, it is automatically updated each time a new view is selected - so that it shows you the Ord of each view.
- Second, since the Ord is displayed in a graphical row of icons, you may click on any icon along that Ord or any child node from the Ord graphical drop-down menu.

Find the **Open Ord** command by selecting **File > Open** from the menu bar.

About the view selector

The view selector appears on the right side of the locator bar, just below the tool bar, as shown in [Figure 2-25](#).

Figure 2-25 View selector



The view selector provides a context sensitive menu with commands that allow you to quickly display different views of the information that is currently in the view pane. The commands in the view selector differ, depending on the current view pane contents. For example, the view selector commands that are available when you are viewing the *platform* in the view pane are different from the commands that are available when you are displaying the *driver manager* view.

Types of popup menus

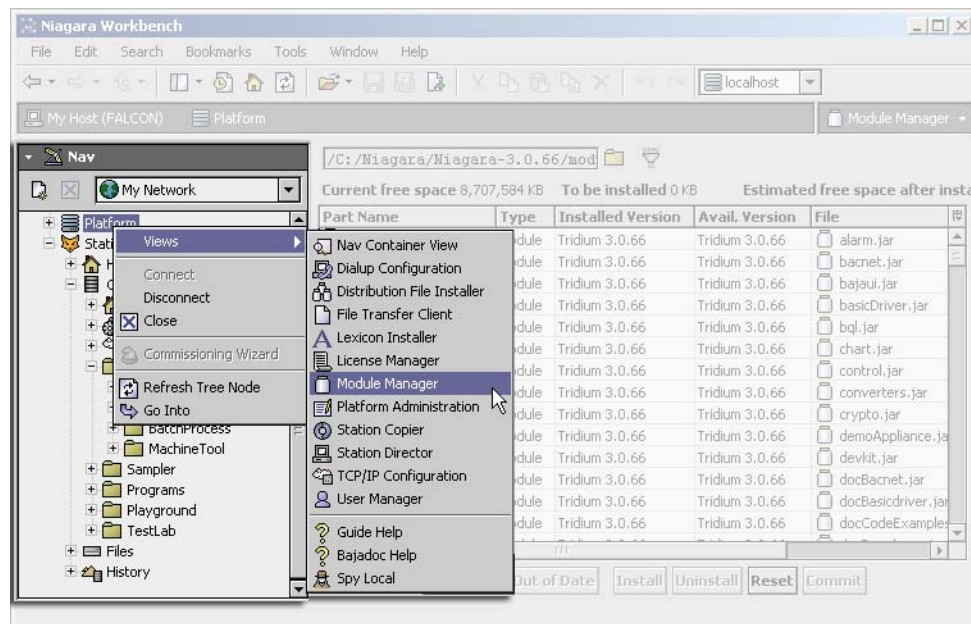
Workbench provides view-specific, or context-specific commands for editing components in many of the views. Following, is a list of the standard Workbench popup (right-click) menus.

- **Nav side bar**
For an overview of the popup menus in the nav side bar, refer to [“About the nav side bar popup menu”](#) on page 2-15.
For an overview of the nav side bar, refer to [“About the nav side bar”](#) on page 2-6.
- **Wire sheet**
For an overview of the popup menus in the wire sheet, refer to [“About the wire sheet popup menu”](#) on page 2-16.
- **Property sheet**
For an overview of the popup menus in the property sheet, refer to [“About the property sheet popup menu”](#) on page 2-16.
For descriptions of the property sheet popup menu items, refer to [“About the property sheet popup menu items”](#) on page A-10.
- **Px Editor**
For an overview of the popup menus in the Px Editor, refer to [“Px Editor popup menu”](#) on page 2-17.

About the nav side bar popup menus

The nav side bar menu is the popup menu that displays command options when you right-click on a component item in the nav side bar. The nav side bar popup menu is context-sensitive and displays items that relate to the object that is currently selected. For example, the popup menu for a station is different from a popup menu for a folder or a point. [Figure 2-26](#) shows a popup menu for a selected node in the nav side bar.

Figure 2-26 Nav side bar popup menu

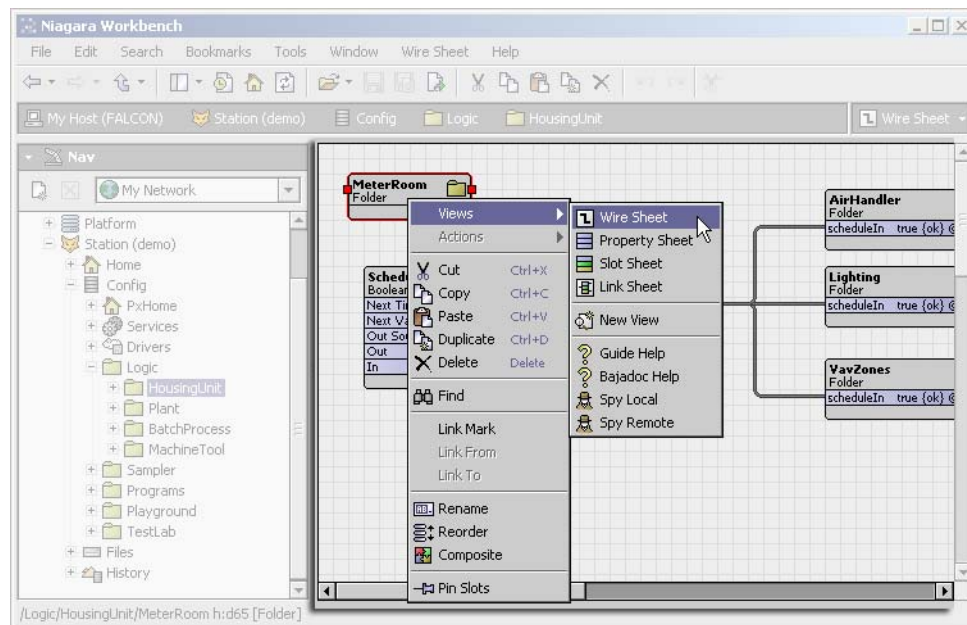


If you right-click in a blank area of the pane (not selecting any items any the tree), the nav side bar pane will display the **Refresh Tree Node** and **Sync Tree** commands. For descriptions of nav side bar popup menu items, refer to [“About the nav side bar popup menu items”](#) on page A-9.

About the wire sheet popup menu

[Figure 2-27](#) shows the wire sheet popup menu with a wire sheet object selected. The menu displays different items depending on whether you right-click on an item or on a blank area of the wire sheet. For descriptions of wire sheet popup menu items, refer to [“About the wire sheet popup menu items”](#) on page A-10.

Figure 2-27 Wire sheet popup menu

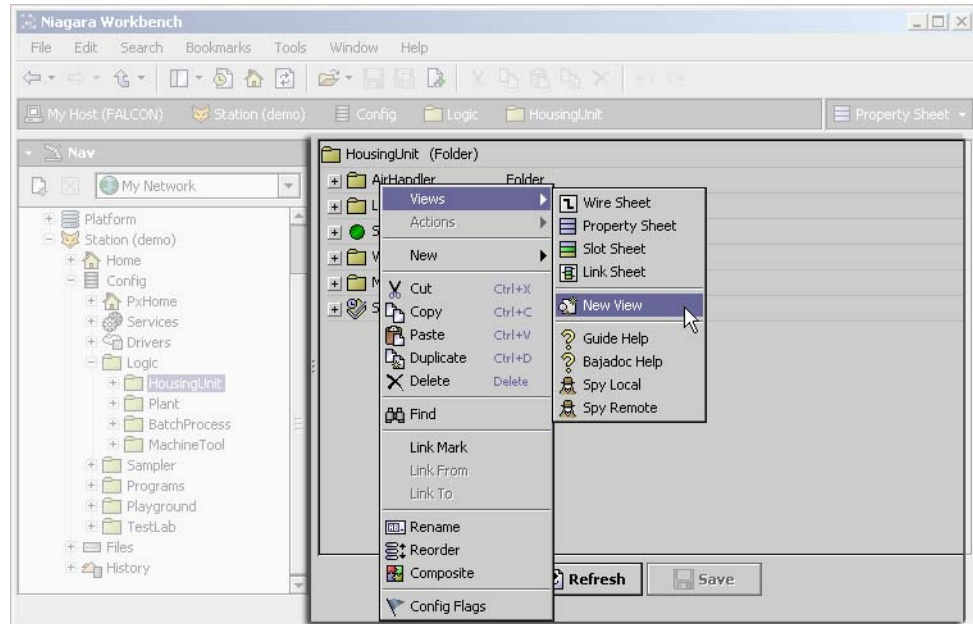


For a description of the wire sheet, refer to [“wiresheet-WireSheet”](#) on page 11-14.

About the property sheet popup menu

[Figure 2-28](#) shows the property sheet popup menu. For descriptions of wire sheet popup menu items, refer to [“About the property sheet popup menu items”](#) on page A-10.

Figure 2-28 Property sheet popup menu



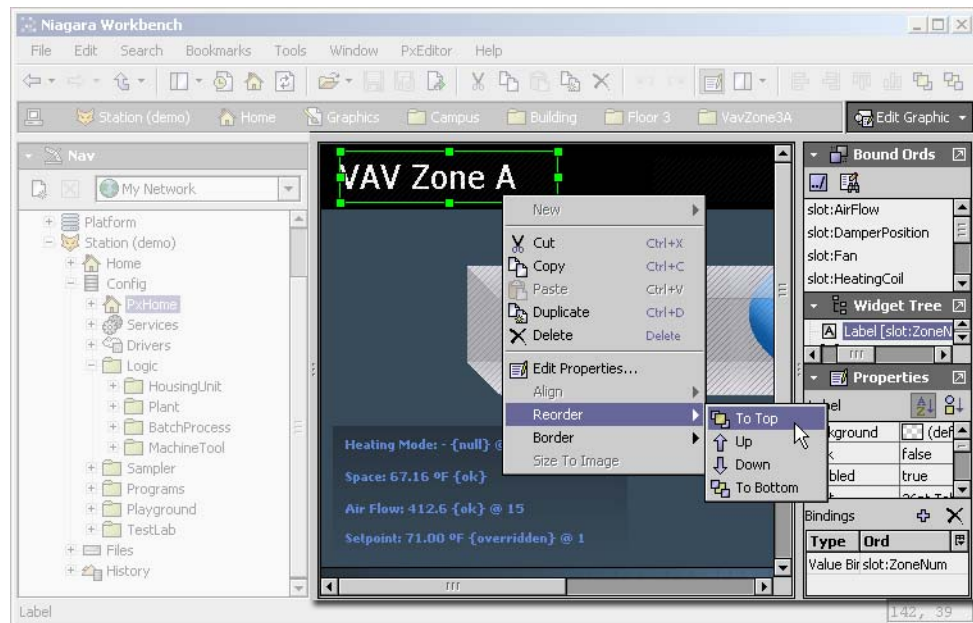
For a description of the property sheet refer to “[workbench-PropertySheet](#)” on page 11-19.

Px Editor popup menu

The Px Editor 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 2-29 shows the Px Editor popup menu. For descriptions of the Px Editor popup menu items, refer to “[About the Px Editor popup menu items](#)” on page A-10.

Figure 2-29 Px editor popup menu



For a description of the Px Editor view, refer to “[workbench-WbPxView](#)” on page 11-23.

Types of data–presentation controls and options

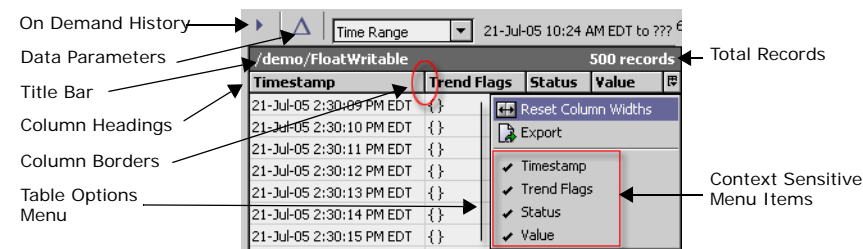
Workbench views often present information in text fields, charts, or tabular layouts. Similar views share many of the same types of controls and options features. Those common features are described, by category, in the following lists:

- **Table controls and options**
These controls affect the display of information in a table view. For details see [“Table controls and options”](#) on page 2-18.
- **Batch editing**
Batch editing is available in many table views. For details see [“Batch editing \(or batch processing\)”](#) on page 2-19.
- **Chart controls and options**
These controls affect the display chart views. For details see [“Chart controls and options”](#) on page 2-19.

Table controls and options

Many views that present information in a table have one or more of the following display features and use one or more of the controls and options described in the following list:

Figure 2-30 Table controls and options



- **On Demand History**
Starting in NiagaraAX-3.4, this “play” button icon is available for the History Table view. Click the button to start On Demand History updating. The button changes to a “pause” button icon while On Demand History is active. See also, [“About On Demand history views”](#) on page 4-20.
- **Data parameters**
These controls include Delta (for history logging) and Time Range settings.
 - **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”](#) on page 4-23 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.
- **Title bar**
This area of the table 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.
- **Table Options menu**
This menu is located in the top right corner of the table and provides one or more of the following controls and options. The standard **table options** menu includes the following items:
 - **Reset column widths**
This menu item 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**
This menu item 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 in the **table options** menu, depending on the component that you are viewing. These additional menu items allow you to select or deselect the item in order to display or hide the column data in the table.

Batch editing (or batch processing)

Batch editing is available in many of the table views in Workbench. This is a process of editing or processing more than one record at a time and typically includes the following general steps:

- Select multiple rows in a table by pressing the Ctrl or Shift key while clicking the desired rows.
- Use the popup (right-click, context-sensitive) menu or a toolbar icon to select an editing command that performs an action or opens a dialog box editor that is appropriate for the view.
- Edit appropriate fields if a dialog box is used and click the **OK** button to process the edits.

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.

Figure 2-31 shows an example of batch processing three alarms by selecting three rows in a table view and using the popup menu. In this case all three alarms are acknowledged when the **Acknowledge** command is selected.

Figure 2-31 Batch processing (acknowledging) multiple alarms

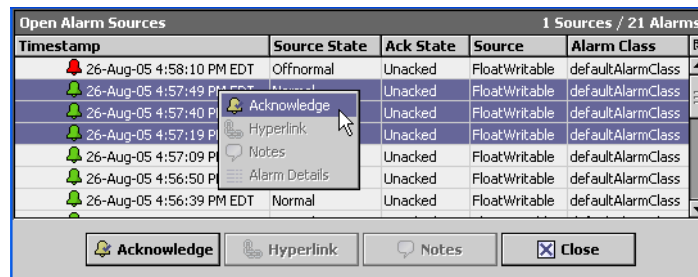
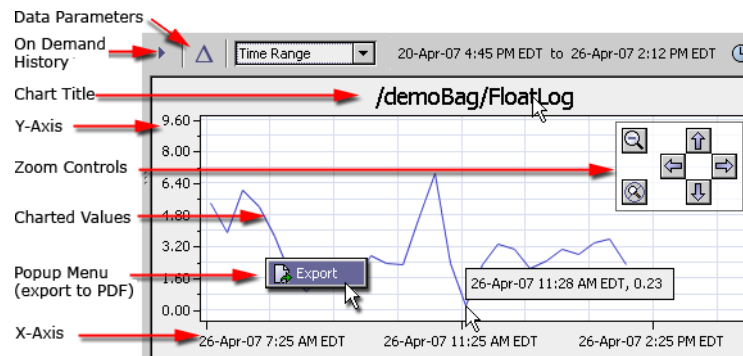


Chart controls and options

Note: Some of the control and option features in the following description are available only in NiagaraAX-3.1 and later versions.

Many views that present information in a chart have one or more of the following display features and use one or more of the controls and options described in the following list:

Figure 2-32 Chart view controls and options



- **Data parameters**
These controls include Delta (for history logging) and Time Range settings.
 - **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”](#) on page 4-23 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.
- **On Demand History**
Starting in NiagaraAX-3.4, this “play” button icon is available for the History Chart view. Click the button to start On Demand History plotting. The button changes to a “pause” button icon while On Demand History plotting is active. See also, [“About On Demand history views”](#) on page 4-20.
- **Chart Title**
This area of the chart displays the name of the chart. This title is editable in the chart builder view.



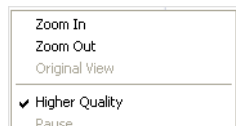
- **Y Axis**
Displays units for the y axis.
- **X Axis**
Displays units for the x axis.
- **Zoom Controls**
Zoom controls appear when you drag the mouse cursor left-to-right or top-to-bottom in order to zoom into the plotted data area. Use the zoom control to reduce the magnification amount, or to move the chart left, right, up, or down. You can reduce the zoom level in increments by using the “decrease zoom” icon . You can restore normal view (no zoom) in one click by clicking the “normal view” icon . Zoom Controls in Hx Web profile view are available from a popup menu, shown in [Figure 2-33](#), that is activated after you click once on the chart area.

Figure 2-33 Hx chart popup menu (for zoom controls)



- The popup menu includes the following control-related menu items:
 - Zoom In—zooms in one increment each time you select this menu item.
 - Zoom Out—zooms out one increment each time you select this menu item.
 - Original View—returns to the non-zoomed display (only available if you have zoomed in).
 - Higher Quality—toggles between higher and lower quality graphic display for the chart.
- **Charted Values**
The color of the line and type of line is editable in the Chart Builder view.
- **Popup menu (Export)**
The popup menu displays the **PDF Export** menu item. This menu item opens the **Export** dialog box where you can choose to export the chart to PDF, text, HTML, or CSV (comma separated variable).
Note: This export function works only with charts that are using a single history. Multiple histories do not export in a usable format.
- **Tool Tip**
When you “hover” the mouse pointer over the 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. [Figure 2-32](#).

Customizing the Workbench environment

You can customize your Workbench GUI as well as many of the settings in the Workbench environment. Some settings require an acknowledgement via the **OK** button on a dialog box, while others are invoked immediately and saved automatically on exit from Workbench. For example, when you exit Workbench with four tabbed windows in your view pane, those same four windows will be displayed the next time you open Workbench.

You can customize the Workbench environment in the following ways.

- [Creating Additional Windows](#)
- [Editing popup menu “new” item](#)
- [Creating tabs in the view pane](#)
- [Working with the Wire Sheet view](#)
- [Types of Workbench options](#)

Creating Additional Windows

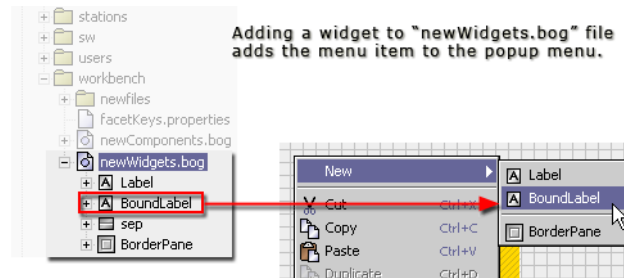
Workbench allows you to create multiple fully functioning Workbench windows. Choosing **File: > New Window** from the menu bar, creates a duplicate of your current Workbench window.

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 window.

Editing popup menu “new” item

You can change the menu items that display on the context-sensitive popup menu by editing the files under the workbench subfolder. For example, Figure 2-34 shows the “newWidgets.bog” file that has been edited so that a “boundLabel” widget appears in the popup menu under the **New** submenu.

Figure 2-34 Customizing the popup menu



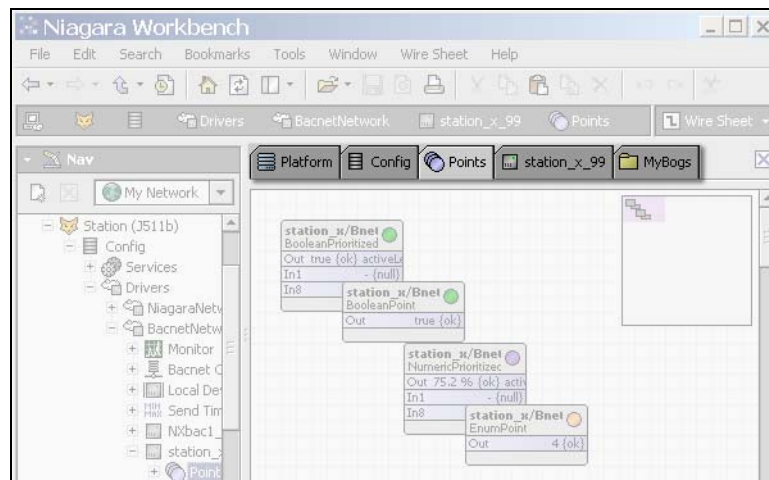
Creating tabs in the view pane

Workbench allows you to create multiple tabbed views within the view pane of a Workbench window, as shown in Figure 2-35, by doing one of the following:

- 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.

The new tab has a view identical to the previous one.

Figure 2-35 Tabs in view



As needed, you can use each tab to display a different view, component, or even host, all within the same Workbench window. When you select a tab (make it *active*), the locator bar shows the current path and view. Also, the menu and tool bars update to show appropriate options for the current view.

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.

In the Workbench view pane you can use tabs to help organize and selectively display information.

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, or you can move to the last tab by choosing **File: > Last Tab** from menu bar.

To close a tab, do any of the following:

- Right-click a tab and choose **Close Tab** to close that tab, *or*
- Right-click a tab and choose **Close Other Tabs** to close all tabs *except* that tab.
- From the menu bar, choose **File: > Close Tab**.

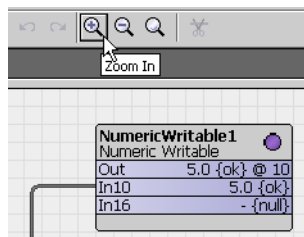
The currently active tab is closed.

Working with the Wire Sheet view

The Wire Sheet view has the following unique controls and features.

- **Zoom controls**
Zoom controls appear on the Workbench toolbar when the Wire Sheet view is active. Click the **Zoom In**, **Zoom Out**, and **Zoom Reset** buttons, as needed to adjust the size of objects in the view.

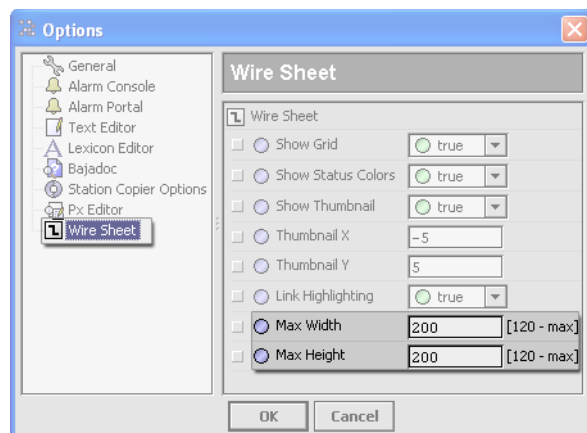
Figure 2-36 Zoom controls in the Wire Sheet view



- **Wire Sheet maximum size limits**
Starting in NiagaraAX-3.6, you can limit the size of the wire sheet by setting size of the wire sheet in the **Options** dialog box. From the workbench main menu, select **Tools > Options** to open the dialog box. Then select the **Wire Sheet** option and enter a desired maximum value in the **Max Width** and **Max Height** fields, as shown, below. Click the **OK** button to complete the setting change. If you decrease the maximum wire sheet size, objects are repositioned to stay within the view, after the size change.

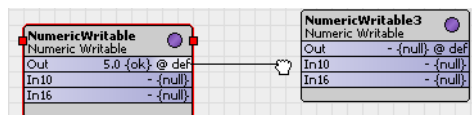
Note: You may need to refresh the wire sheet view to see the changes in wire sheet size and object repositioning. Also, the wire sheet borders may not be visible until you zoom out in the wire sheet view.

Figure 2-37 Setting maximum wire sheet size



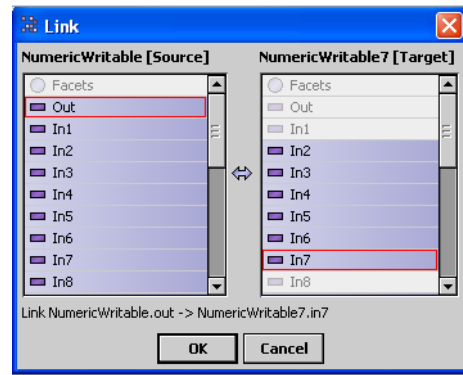
- **Object linking**
Following are methods for performing object linking.
 - **Drag wire**
Move the mouse over the In or Out node of a wire sheet object (glyph) until the area is highlighted and the mouse pointer changes to a white arrow. Click and drag from the point of origin to the desired In or Out node of the target object and release the mouse button to complete the link. See [Figure 2-38](#).

Figure 2-38 Basic object linking



Dragging a “wire” to a vacant slot on a wire sheet object causes the **Link** dialog box to open, as shown below.

Figure 2-39 Link dialog box



Select the desired slots in the **Link** dialog box and click the **OK** button to create the link.

- You cannot select invalid (dimmed) slots in the dialog box.
- Slot selection is indicated by a red rectangle surrounding the selection.
- Object names are listed at the top of their respective columns to identify them and to indicate which object is the “Source” and “Target” object.
- A summary of the currently selected slots displays above the **OK** and **Cancel** buttons.

- **Multi-linking (continuous link state)**

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. To do this, hold down 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. Release the Shift key and click anywhere or touch any key to deactivate the link state.

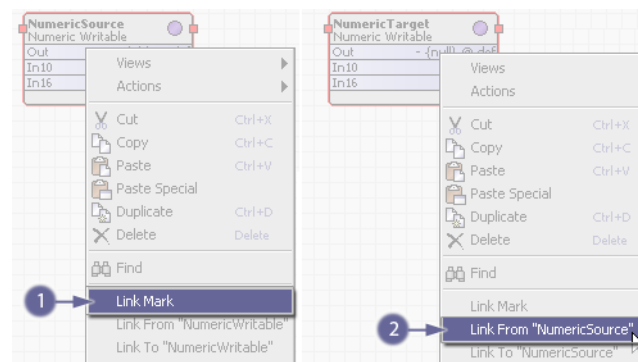
- **Link Mark**

Link Mark is a feature that allows you to perform “one to many”, “many to one”, or “many to many” linking in a single operation. Use the following three menu item commands to create many links in at the same time.


- **Link Mark**

Select the **Link Mark** command to define one or more selected objects as a link source or link target. 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.

Figure 2-40 Using Link Mark



- **Link From**

Select the **Link From** command to define one or more selected objects as a link source or link target. 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 .

- **Link To**

Select the **Link To** command to define one or more selected objects as a link source or link target. This command opens the **Link** dialog box with the “marked” object as the tar-


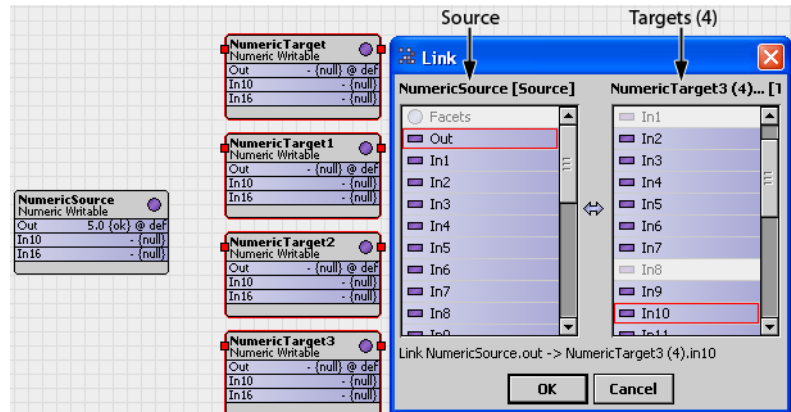
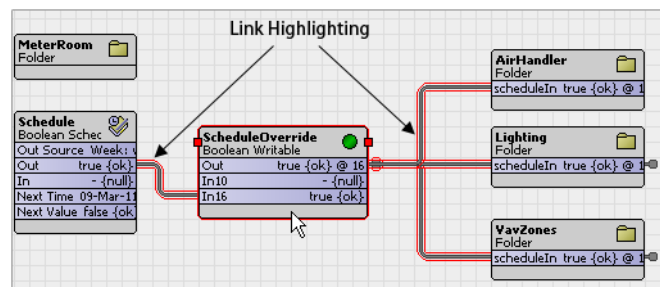
get object. You can still change source and target roles in the dialog box using the **Reverse** button .

Figure 2-41 Multiple links with Link Mark



- **Link Highlighting** (NiagaraAX-3.6)
Starting in NiagaraAX-3.6, a 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. Link highlighting is comprised of the following features:

Figure 2-42 Link highlighting on the wire sheet



- **Colored link highlights**
When you 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.
- **Distinct coloring**
Selecting additional components causes subsequent link highlights to display a different color highlighting (up to 20 colors before repeating).
- **Custom colors**
You can customize the colors by editing the “system.properties” file (in the nav tree “lib” folder, under **MyFiles > Sys Home**) as follows:

```
niagara.wiresheet.linkHighlightColor1=#00FF00
```


...and following, to...

```
niagara.wiresheet.linkHighlightColor20=#00FFFF
```


Each color is specified using the standard hexadecimal notation used for HTML color display.
- **Off-view link navigation** (NiagaraAX-3.6)
Starting in NiagaraAX-3.6, link “knob” indicators (used for off-view or crowded view links) have additional properties.
 - **Off-view linking**
Double-clicking on the knob hyperlinks to the component on the opposite end of the link, displaying the wiresheet 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 wiresheet view with the linked knob highlighted.

- [Text Editor options](#)
- [Lexicon Editor options](#)
- [Bajadoc options](#)
- [Station Copier options](#)
- [R2 Platform Tool options](#)
- [Px Editor options](#)
- [Wire Sheet options](#)

General options include the active Workbench “Theme”. See [“About Workbench themes”](#) on page 2-32.

General options

General Workbench options, shown in [Figure 2-46](#), include settings for a variety of Workbench display and behavior options.

Figure 2-46 General Workbench options

Option	Value
<input type="checkbox"/> Time Format	(default)
<input type="checkbox"/> Unit Conversion	None
<input type="checkbox"/> Auto Logoff Enabled	false
<input type="checkbox"/> Auto Logoff Period	+00000h 15m 00s
<input type="checkbox"/> Prompt For Name On Insert	true
<input type="checkbox"/> Font Size	Normal
<input type="checkbox"/> Active Theme	Lucid
<input type="checkbox"/> Allow User Credential Caching	true
<input type="checkbox"/> Locale	en_US

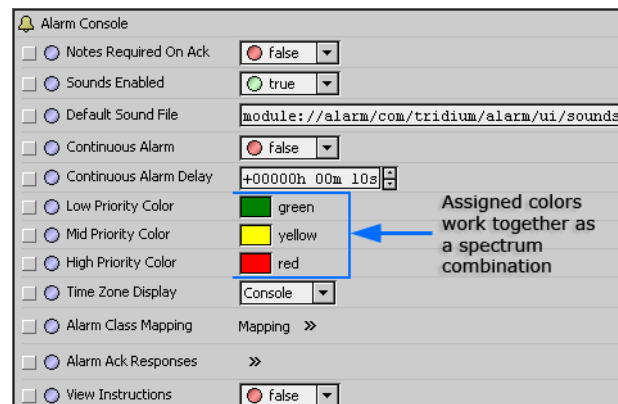
Note: The Time Format and Unit Conversion parameters affect values that are displayed when connected to a station using Workbench—regardless of the User preferences (set under User Manager). The User preferences that are set under the User Manager are in effect when connected to a station by a browser.

- **Time Format**
Choose from a format option to set the way that Workbench 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 will display a **Name** dialog box, 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 Frost, Lucid or Palladium “built-in” theme options for Workbench display. See [“About Workbench themes”](#) on page 2-32.
- **Allow User Credential Caching**
(AX-3.7 and later) 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 best security 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”](#) on page 8-8.
- **Locale**
(AX-3.8 only) 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, shown in [Figure 2-47](#), allow you to customize both the appearance and behavior of the alarm console.

Figure 2-47 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, shown in [Figure 2-48](#), allow you to customize both the appearance and behavior of the alarm console.

Figure 2-48 Alarm portal options

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

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](#)” on page 10-31, the *NiagaraAX Graphics Guide* section “About Kiosk Profiles”, and the engineering notes document *NiagaraAX Kiosk Mode*.
- **Reconnect Interval**
The alarm portal checks for “disconnected” alarm consoles. If a console is disconnected, a reconnect is attempted within the Reconnect Interval time.

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 in [Figure 2-49](#).

Figure 2-49 Text editor options

Text Editor	
Text Editor	
<input type="checkbox"/> Show Spaces	<input type="radio"/> false
<input type="checkbox"/> Show Tabs	<input type="radio"/> false
<input type="checkbox"/> Show Newlines	<input type="radio"/> false
<input type="checkbox"/> Tab To Space Conversion	2 [0 - max]
<input type="checkbox"/> Show Margin	0 [0 - max]
<input checked="" type="checkbox"/> Color Coding	Color Coding
<input checked="" type="checkbox"/> Key Bindings	Key Bindings
<input type="checkbox"/> Undo Navigation	<input checked="" type="radio"/> true
<input type="checkbox"/> Match Pairs	<input checked="" type="radio"/> true
<input type="checkbox"/> Match Braces	<input checked="" type="radio"/> true
<input type="checkbox"/> Match Brackets	<input checked="" type="radio"/> true
<input type="checkbox"/> Word Right To End Of Word	<input checked="" type="radio"/> true

Lexicon Editor options

These options offer ways to customize the default settings of the lexicon editor. The lexicon editor options properties are shown in [Figure 2-50](#) and described in “[Lexicon Editor view](#)” on page 8-5. All of these properties may be overridden using the options that are available in the Lexicon Editor.

Figure 2-50 Lexicon editor options



Bajadoc options

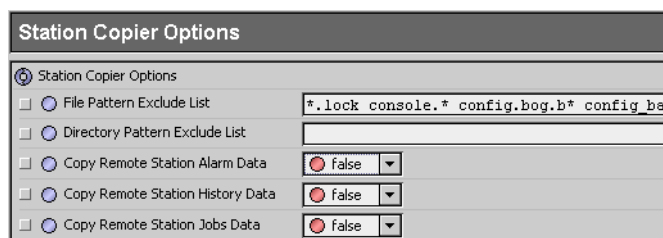
Baja reference documentation includes both Java API details as well as Baja slot documentation.

- **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 2-51 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: (Station Alarm Data, Station History Data, Station Jobs Data)**
These properties allow you to choose to copy (`true`) or not copy (`false`) certain station data.

R2 Platform Tool options

These options apply when platform connected to a JACE-603 or JACE-645 (or RB-603/RB-645 “retrofit board” JACE) being configured for Niagara R2.

Figure 2-52 R2 Platform Tool Options

- **Local R2 Installation**
Specifies the local file path to the Niagara R2 installation directory. The default value of NiagaraAX installation folder (!) is typically inappropriate. However, you can alternatively specify the source Niagara R2 installation folder when running the “**Install Niagara R2**” wizard, launched from the **Platform Administration** view of a JACE-603 or JACE-645.
- **File Pattern Exclude List**
Use this option to set a pattern filter that excludes any specified file types from being copied with the station, when using the **R2 Station Copier**.
- **Directory Pattern Exclude List**
Use this option to set a pattern filter that excludes any specified directories from being copied with the station, when using the **R2 Station Copier**. 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.

For related details, refer to the *Retrofit Board Niagara R2 Install & Startup Guide*, including sections “R2 Platform Tool” and “Current R2 Installation”.

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 in [Figure 2-53](#) and described in the following list:

Figure 2-53 Px editor options

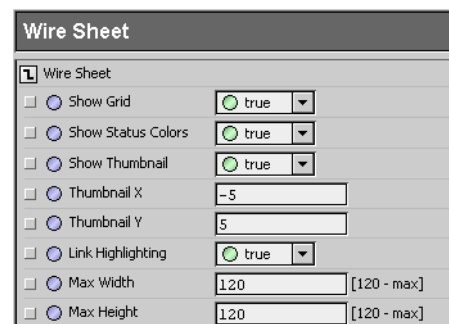
- **Show Grid**
This property sets the default condition of the Px editor grid. Select `true` to make the grid visible by default or select `false` to make the grid hidden by default. Either setting may be changed at any time using the PxEditor menu.
- **Grid Size**
This property sets the size of the 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, shown in [Figure 2-54](#), allow you to customize the appearance of the wire sheet.

Figure 2-54 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 (NiagaraAX-3.6 and later)**
When set to `True` – turns on link highlighting. See also, “[Working with the Wire Sheet view](#)” on page 2-22.
- **Max Width (NiagaraAX-3.6 and later)**
Use this property to specify a maximum size for the wiresheet width. See also, “[Working with the Wire Sheet view](#)” on page 2-22.
- **Max Height (NiagaraAX-3.6 and later)**
Use this property to specify a maximum size for the wiresheet height. See also, “[Working with the Wire Sheet view](#)” on page 2-22.

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 Workbench display. Choosing a different theme affects only the Workbench appearance, there is no other impact.

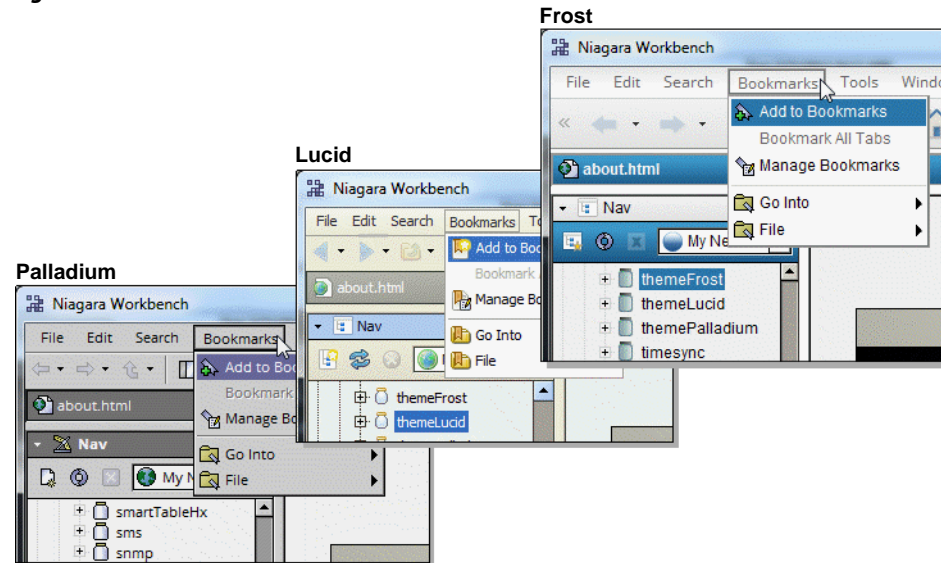
You can customize your Workbench display by selecting the theme that you prefer. In addition to permitting you to customize your Workbench display, themes can also be used to give Workbench a “branded” appearance.

Types of themes

Within Workbench, there are standard “built-in” themes that you can use. You can also create and use custom themes. The built-in themes are installed as part of the Workbench application. Starting in AX-3.7, there are *three* built-in themes available for use, as shown below in [Figure 2-55](#):

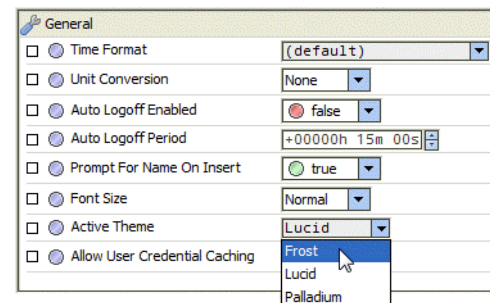
- **Frost**
Displays a medium-blue color scheme with many custom icons.
- **Lucid**
Displays a blue and gray color scheme. This is the theme applied to Workbench by default.
- **Palladium**
Displays a gray color scheme.

Figure 2-55 Workbench Built-in Themes



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 in [Figure 2-56](#). In order for the theme change to take effect, you must exit Workbench and restart it.

Figure 2-56 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.

Define a default custom theme for your brand

You can do this by creating a new custom theme module that specifies colors (such as, corporate colors), fonts, and icons to be used in the Workbench display and designating this as the default theme. Workbench automatically selects this theme if the user does not make another selection. You can also “force” the use of the default theme, effectively locking Workbench into using the default theme.

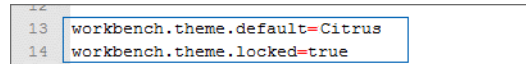
Designate a default theme

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

- Step 1 Open the `brand.properties` file in the `/lib` folder and add the following two lines of text, replacing “themeName” with your custom theme name, as shown in [Figure 2-57](#):

```
workbench.theme.default=themeName
workbench.theme.locked=true
```

Figure 2-57 Text to designate default theme and lock it.



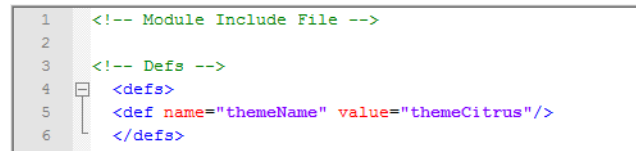
- Step 2 Click **File > Save** to save your changes in the `brand.properties` file.
- Step 3 Restart Workbench for this change to take affect.

Note: Locking the theme hides the **Active theme** drop-down, normally visible in **Workbench Options**, so that no other theme can be selected.

About theme modules

As with all Workbench modules, a theme module contains the `build.xml` and `module-include.xml` files. A module is identified as a theme module by adding a `<defs>` tag with a themeName definition within `module-include.xml`, as shown below:

Figure 2-58 `<defs>` tag within `module-include.xml` file.



Also required in a theme module, is at least one text file with the extension `.nss` located in `/src/nss`. A Niagara Style Sheets (NSS) file is used to enable styling of fonts, colors, and other attributes of the Workbench display using syntax and keywords similar to those used in HTML Cascading Style Sheets (CSS).

Note: Although NSS shares some syntax and keywords with CSS, the languages are not the same. NSS performs cascading like CSS, however the syntax is different. Also, NSS has no positioning selectors while CSS has many. Refer to the “NiagaraAX Developers Guide” for more details.

All of the resources files (NSS files, .PNG, etc.) for you theme module must be located under the `/src` folder and must be specified within `build.xml` with the standard `<resources>` tag:

```
<resources name="nss/*.*)" />
```

Note: With all resources tags, you must omit the `/src` directory from the value provided for the “name” attribute.

You can override any Workbench icons with custom icons in your theme module. Your custom icon image files (*.PNG) must be located in the `/src/imageOverrides` folder within your theme module. Subfolders within `/imageOverrides` correspond to the name of the module containing the icons you wish to override. For example, to override a Workbench icon located in the icons module your .PNG file must be located in `/imageOverrides/icons/x16` in your theme module and that exact path must be specified in a resources tag within the `build.xml` file for your theme module:

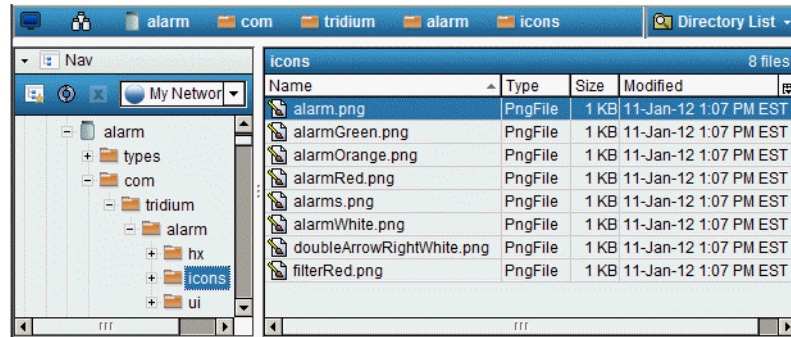
```
<resources name="imageOverrides/icons/x16/*.*)" />
```

Note: When overriding icons you must duplicate the original filenames and folder structure exactly. Otherwise, the path to the icons will be incorrect and the custom icons will not display. You can determine the correct path and filenames to use by examining the feature whose icons you want to override and look inside the corresponding module for the path to those icon files.

For example, if overriding the alarm icon in the alarm module, first examine the contents of the alarm module and note the filename and path to that icon (alarm/com/tridium/alarm/icons/ alarm.png), as shown in [Figure 2-59](#). Next, duplicate that path under the /imageOverrides folder in your theme module, placing your image file named alarm.png in: /imageOverrides/ alarm/ com/tridium/ alarm/ icons. Next, specify the exact path in a resources tag within the build.xml file for your theme module:

```
<resources name="imageOverrides/ alarm/ com/tridium/ alarm/ icons/  
alarm.png" />
```

Figure 2-59 Determine filename and path to icon for override



Also, you can have several themes sharing the same set of custom icons by placing multiple NSS files inside the /src/nss folder. Each of the NSS files will be individually selectable as a separate theme in the **Active Theme** drop-down. Doing this eliminates the need to have duplicate sets of icons in separate theme modules.

The following procedures describe how to:

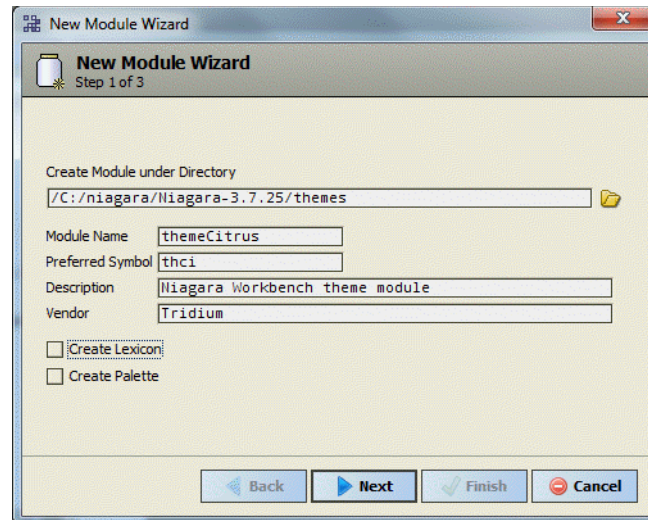
- [Create a new theme module.](#)
- [Copy resources from a built-in theme module.](#)
- [Modify design elements in the NSS file.](#)
- [Build a.jar file for the theme module.](#)
- [Activate the new theme module.](#)

Create a new theme module

This procedure describes the steps used to create a new theme module.

- Step 1 In the Workbench Nav pane, under My File System, navigate to Niagara 3.7.x and expand it. Create a new /themes subfolder there to store your custom themes. This is where the new theme module that you create will be located.
- Step 2 Click **Tools > New Module** and fill-in the fields in the New Module dialog box, as shown in [Figure 2-60](#).
 - Enter the path to the /themes folder created above (for example: C:/niagara/Niagara-3.7.x/themes).
 - Enter a name for the theme using this naming convention, *themeName* (for example: themeCitrus).
 - Enter a Preferred Symbol (four-letter abbreviation) for the new theme name (for example: thci).
 - Enter a Description (for example: Niagara Workbench theme module).
 - Enter a Vendor (for example: Tridium).
 - Clear the checked check box for **Create Lexicon** and click the **Next** button to advance to the next screen.

Figure 2-60 New Module dialog box

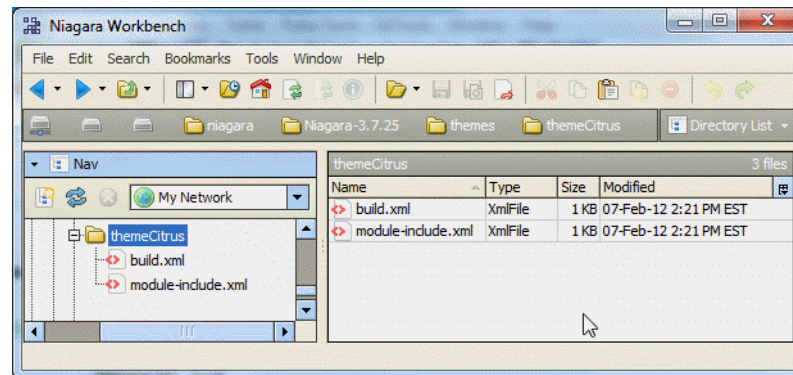


Step 3 Remove any Dependencies listed in the dialog box and click the **Next** button.

Step 4 No Packages are required. Click the **Finish** button to create the new module.

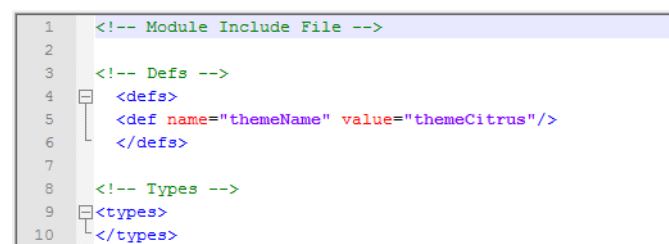
The new module contains the two required XML files: `build.xml` and `module-include.xml`, as shown in Figure 2-61.

Figure 2-61 New module



Step 5 Using a text editor, open `module-include.xml` and add the `<defs>` tag to identify this module as a theme module, as shown in Figure 2-62:

Figure 2-62 Add `<defs>` tag in `module-include.xml`



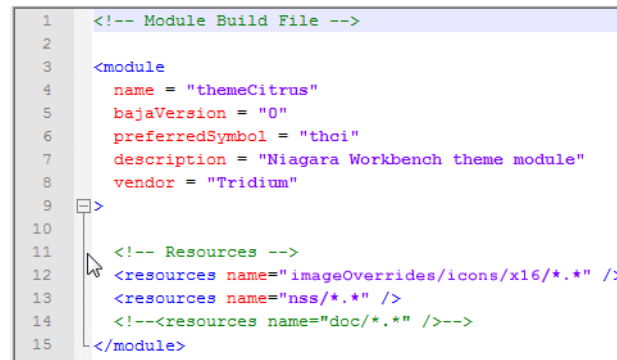
Copy resources from a built-in theme module

This procedure describes how to obtain resources from another theme module and add appropriate resources tags to the `build.xml` file in the new theme module.

In addition to the .xml files mentioned earlier, the theme module requires that at least one Niagara Style Sheet (NSS) file be located in: `/src/nss`. Copying an existing NSS file, along with other resources, from one of the built-in themes will save time and reduce the chance of introducing errors.

- Step 1 In the Nav pane under My File System, navigate to your new theme module in the /themes folder. Create a new subfolder there named /src (for example: /themes/themeCitrus/src). This is where all resources must be located.
- Step 2 In the Nav pane under My Modules, scroll down to locate the built-in theme modules (themeFrost.jar, themeLucid.jar, themePalladium.jar). Double-click themeLucid.jar to display its resources. Select all the resource folders and copy them.
- Step 3 Under My File System, navigate to your new theme module in the /themes folder, and paste the copied resource folders into the /src subfolder.
- Step 4 Using a text editor, open the build.xml file in your theme module and add a <resources> tag for each folder copied from the built-in theme module, as shown in Figure 2-63:

Figure 2-63 Specify resources in build.xml



```
1 <!-- Module Build File -->
2
3 <module
4   name = "themeCitrus"
5   bajaVersion = "0"
6   preferredSymbol = "thci"
7   description = "Niagara Workbench theme module"
8   vendor = "Tridium"
9 >
10
11 <!-- Resources -->
12 <resources name="imageOverrides/icons/x16/*.*" />
13 <resources name="nss/*.*" />
14 <!--<resources name="doc/*.*" />-->
15 </module>
```

At this point, your new theme module has everything needed to build a .jar file. However, since the NSS file was copied from another theme, your new theme will have an identical appearance. To make your new theme distinguishable from the other, you must change design elements within the NSS file.

Modify design elements in the NSS file

This procedure briefly describes an approach to modifying design elements in an NSS file to give Workbench a unique appearance.

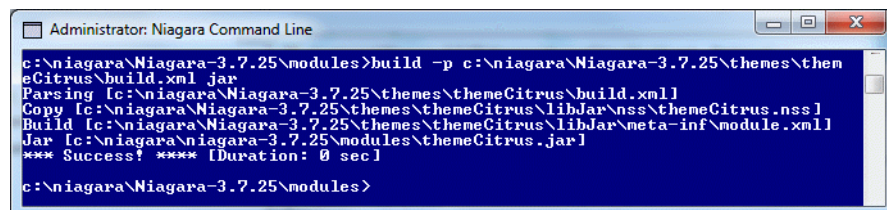
- Step 1 In the Nav pane under My File System, navigate to your theme module and expand it to access the NSS file in /src/nss. Using a text editor, open the NSS file and scroll to the Define statements (#define...) located at the top of the NSS file. These statements define certain font and color styles that typically are applied repeatedly throughout the NSS file. Modifying a few color definitions in the #define statements will have the most noticeable impact on the Workbench display.
Colors may be specified as RGB colors in hexadecimal format (for example: #000, #ffffff); or in many cases, colors may be specified according to their common English names, such as gray, white, blue, etc.
Note: Refer to the NiagaraAX Developer Guide section on HX Theming for more information on NSS file properties. Also, an Internet search for "html color codes" will result in many color code examples.
- Step 2 After completing modifications, save the NSS file.

Build a .jar file for the theme module

This procedure describes building a .jar file for a new theme module so that the module can be deployed in Workbench.

- Step 1 Open the Niagara Console and run the following build command, as shown in Figure 2-64:
`build -p [path to build.xml file] jar`

Figure 2-64 Enter command in Niagara Console to build .jar file



```
Administrator: Niagara Command Line
c:\niagara\Niagara-3.7.25\modules>build -p c:\niagara\Niagara-3.7.25\themes\themeCitrus\build.xml jar
Parsing [c:\niagara\Niagara-3.7.25\themes\themeCitrus\build.xml]
Copy [c:\niagara\Niagara-3.7.25\themes\themeCitrus\libJar\nss\themeCitrus.nss]
Build [c:\niagara\Niagara-3.7.25\themes\themeCitrus\libJar\meta-inf\module.xml]
Jar [c:\niagara\Niagara-3.7.25\modules\themeCitrus.jar]
*** Success! *** [Duration: 0 sec]
c:\niagara\Niagara-3.7.25\modules>
```

- Step 2 Confirm that the .jar file (for example: themeCitrus.jar) was created in the /modules folder.

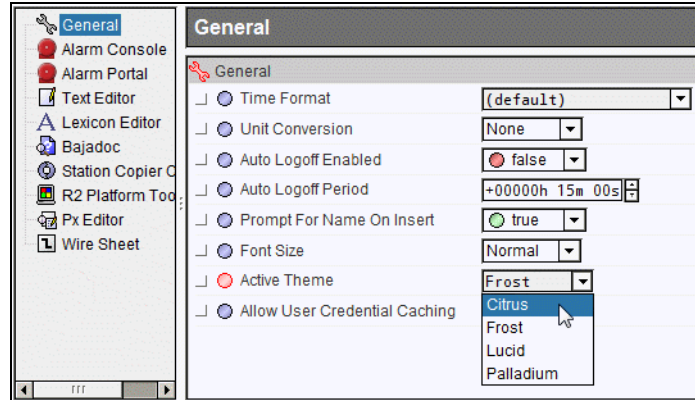
- Step 3 Exit and restart Workbench to load the .jar file.

Activate the new theme module

This procedure covers activating a new theme module.

- Step 1 Select the new theme module in **Tools > Options** under the Active Theme drop-down, as shown in [Figure 2-65](#).

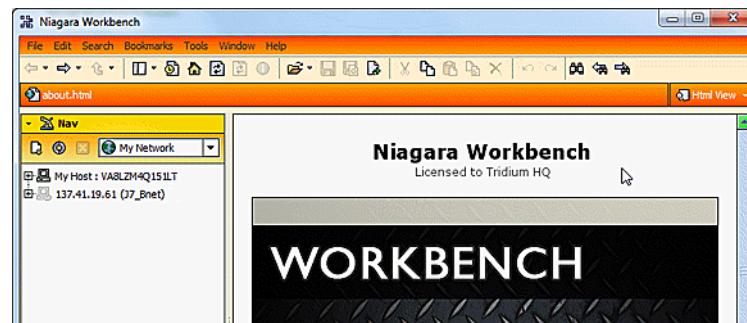
Figure 2-65 New theme is selectable in the Active Theme drop-down



- Step 2 Click **File > Close** and restart Workbench, the new theme will be active, as shown in [Figure 2-66](#).

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.

Figure 2-66 New theme is applied to Workbench



CHAPTER 3

Data and Control Model

In any NiagaraAX station, all real-time data is *normalized* within the station database as *points*, a special group of components. Each point represents one data item.

Note: *This applies to a Supervisor station as well as a JACE station, a shift from r2 Niagara data modeling. Stations no longer have “external links” between them. Instead, proxy points under a Niagara network are used. For more details, refer to the Drivers Guide section “About the Niagara Network”.*

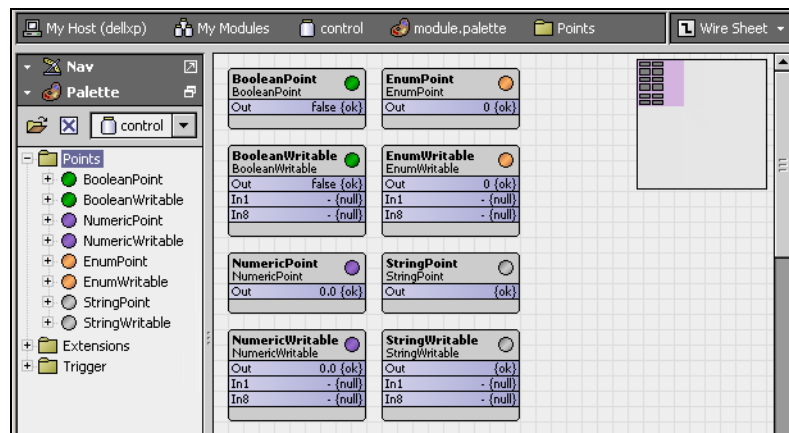
The following sections provide more details:

- [About control points](#)
- [About point extensions](#)
- [About control triggers](#)
- [About point status](#)
- [About writable points](#)
- [About composites](#)

About control points

In the most simple form, points are *control points*. Find them in palette **control: Points**.

Figure 3-1 Control points



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

- [Types of control points](#)
- [Other control components](#)
- [About data value categories](#)
- [About point types](#)
- [About point Out](#)
- [About point inputs](#)
- [About point actions](#)
- [About point facets](#)

Types of control points

As shown in [Table 3-1](#) there are 8 different simple control points. Each type reflects a combination of a data (value) category and a point type.

Table 3-1 Control point types

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

See “[About data value categories](#)” on page 3-2 and “[About point types](#)” on page 3-2.

Other control components

Apart from the 8 simple points, many other components are found in both the control palette and the kitControl palette. Briefly, these components include:

- **Extensions**
To “extend” point functionality. See “[About point extensions](#)” on page 3-12.
- **Triggers**
To provide periodic “actions.” See “[About control triggers](#)” on page 3-17.
- **Other control objects**
Found in various folders in the kitControl palette. Use these to provide station control logic based on data obtained from points. Example objects include (numeric) math types, (Boolean) logic types, and a PID loop, among others. Also, a few point extensions are in kitControl. See the section “Application for kitControl components” in the *kitControl Guide*.

About data value categories

There are four distinct *data value categories* for points, as well as for other components (for example, weekly schedules). These value types are:

- **Boolean**
Represents a *binary value* with only *two* states, such as “Off” or “On.”
 - **Numeric**
Represents an *analog value* such as a temperature, level, rate or similar floating point number, or a varying count (integer). Double-precision (64 bit) values are used.
 - **Enum**
Represents an *enumerated state* (more than two), such as a multi-speed fan with states “Off,” “Slow,” and “Fast.” Enums are often called *multi-states* or *discretes*. States typically derive from established integer value/state name pairs.
 - **String**
A string of one or more *ASCII characters* (and if alpha-numeric), often with some literal meaning.
- Among the four categories, numeric and boolean points are the most common, as data usually falls into these two categories.

About point types

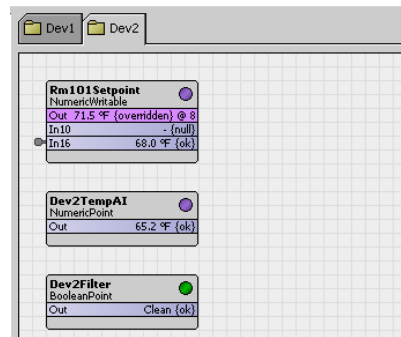
Within each of the four point data categories, there are two point types:

- **(read-only) e.g. BooleanPoint or NumericPoint**
Represents a *read-only* data item. Unlike with the writable point type, there are no “input” type properties.
Note: *As copied directly from the control palette, there is no application for read-only points. However, proxy points based upon read-only points, 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](#)” on page 3-13, or refer to “[About proxy points](#)” in the Drivers Guide.*
- **Writable**
Represents a data item that can be *written*, as well as read by the station (typically). An array of 16 “InN” inputs, each with a different priority level, is available to write the 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](#)” on page 3-3. Writable points also have other features. See, “[About writable points](#)” on page 3-21.

About point Out

Each point has a property “Out” that provides real-time information, as shown in [Figure 3-2](#).

Figure 3-2 point Out provides real-time information



At a minimum, the Out property provides:

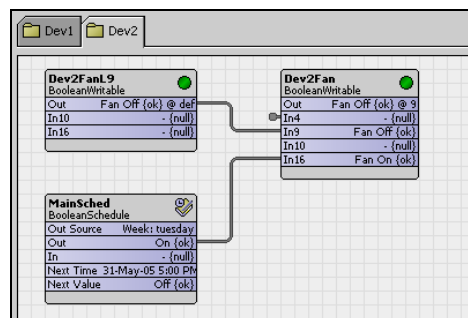
- **value**
Current *value* of the data item, within one of four possible data categories. See “About data value categories” on page 3-2.
- **facets**
Items affecting the *display* of the value, such as decimal places, engineering units, or text descriptors for Boolean/enum states. See “About point facets” on page 3-7.
- **status**
Current *status* of the data, meaning the health and validity of the associated 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 by default as “{ok}.”
Status flags are set in a number of ways. See “About point status” on page 3-18.
- **priority**
Currently active Niagara priority level (from a 16-level priority scheme), for writable control points only. For more details on priority, see “About the priority scheme” on page 3-21.
Note: All writable control points include priority in status, including writable proxy points.
Priority appears at the *end* of the Out value, by default formatted as “@ n”, where n is 1 to 16, or if the fallback value is in effect, formatted as “@ def”. For example: “Off {ok} @ 1”.

About point inputs

Writable points have 16 input properties: “In1” to “In16,” corresponding to priority levels. As needed, you can link up to 14 of these inputs to another component in the station. Figure 3-3 shows a BooleanWritable with links made to 3 inputs.

Note: Read-only (plain) points do not have any input properties.

Figure 3-3 Example links made to priority inputs

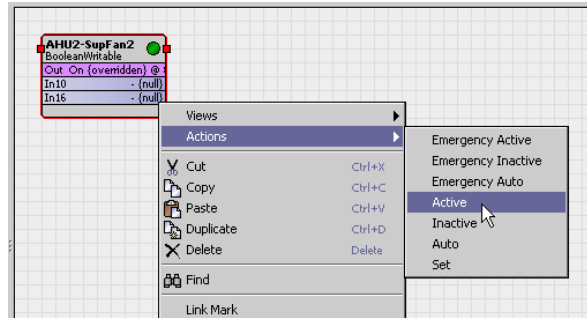


By default, a writable point’s glyph (shape on wire sheet) shows only inputs In10 and In16 “pinned” for linking. However, when linking to a writable point, the Link editor shows all available priority inputs. For more details, see “About the priority scheme” on page 3-21.

About point actions

Writable points have *actions*—by default, these appear in a right-click menu when you select a point in a Workbench view or in the Nav tree (Figure 3-4).

Figure 3-4 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 4 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 dialog. See [“About override actions”](#) on page 3-4.
- Set the value of the point’s “Fallback” property. See [“About set \(Fallback\) action”](#) on page 3-4.

Often, you modify a writable point’s default actions—see [“Modifying default actions”](#) on page 3-5.

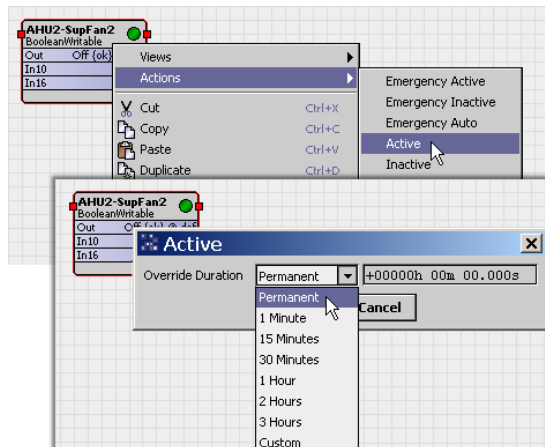
About override actions

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 in [Figure 3-4](#). For more point status details, see [“How status flags are set”](#) on page 3-19.

A manual (level 8) override action to a point (not “auto”) prompts for an override duration, see [Figure 3-5](#).

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

Figure 3-5 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”](#) on page 3-10.

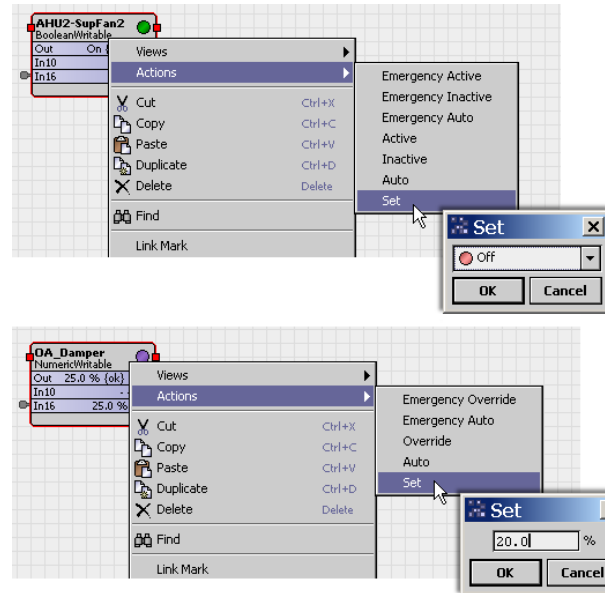
After clicking OK, the override action is issued to the point—if this is the highest effective priority level, the writable point operates under this control. If this is a timed override, the action is automatically auto’ed upon expiration of the override period.

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”](#) on page 3-21.

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 3-6](#)).

Figure 3-6 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” on page 1-9, and “Using the slot sheet” on page 9-19.

Or, using facets you can limit the duration of manual overrides—see “Maximum override duration facet” on page 3-10.

The following sections provide more details on *slot sheet* techniques:

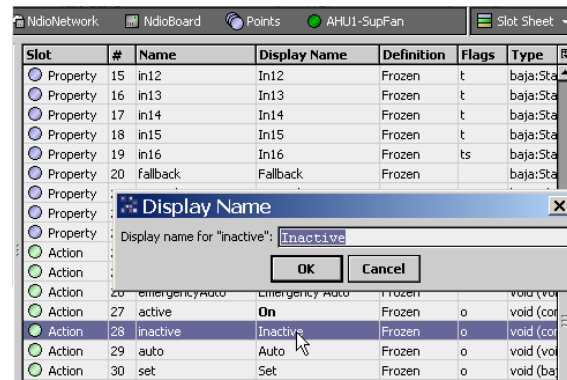
- **Display names** — Change how the point's popup action menu lists available actions
- **Action access** — Limit the actions that are made available, either by user level or for all users

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 NiagaraAX Lexicon Guide.

Note: 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 > Program-Service). This is one of many examples of using the Batch Editor, available in all NiagaraAX releases. 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 (Figure 3-7). When you change a display name from defaults, it appears in listed in bold.

Figure 3-7 Editing action display names from slot sheet



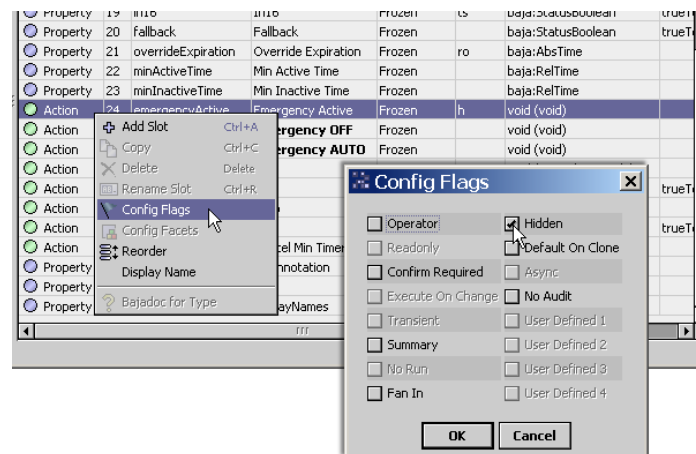
When a user invokes an action, the popup menu lists possible actions by more meaningful descriptors. For example, you could change the “set” action display name from “Set” to “Set Fallback.”

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 in Figure 3-8).

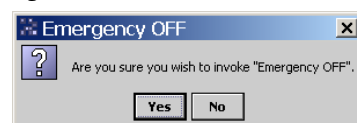
Figure 3-8 Editing config flags of action to “hide” or change permission level



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” on page 6-10.
- **Confirm Required**
If checked, a second (confirmation) dialog appears after the action is invoked, before the action executes. An example confirmation dialog is shown in Figure 3-9. By default, this flag is cleared.

Figure 3-9 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.*

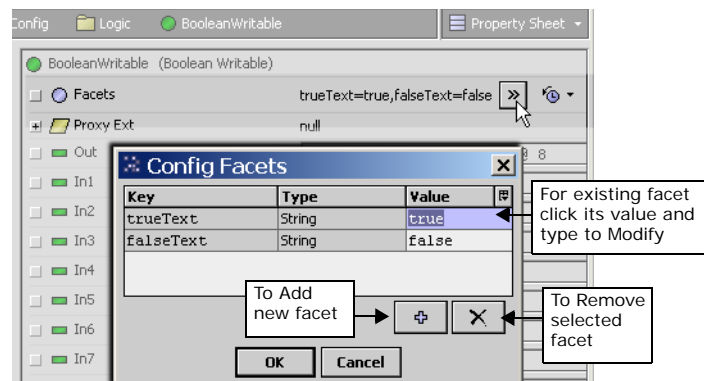
Topics about facets include:

- [Accessing and editing facets](#)
- [Facets importance for Enum points](#)
- [Effect of facets on point actions](#)
- [Maximum override duration facet](#)

Accessing and editing facets

Facets is a frozen slot in all control points and objects in the kitControl module. As shown in [Figure 3-10](#), you modify facet values in a point’s property sheet, using the **Config Facets** dialog.

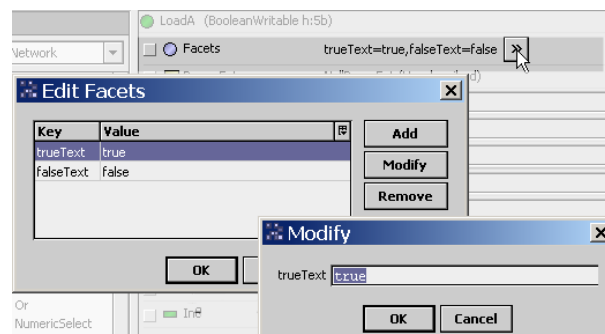
Figure 3-10 Point facets and edit dialog (AX-3.3 and later shown)



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 [Figure 3-11](#) below. However, the basic operation is the same.*

Figure 3-11 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](#)” on page 3-10).

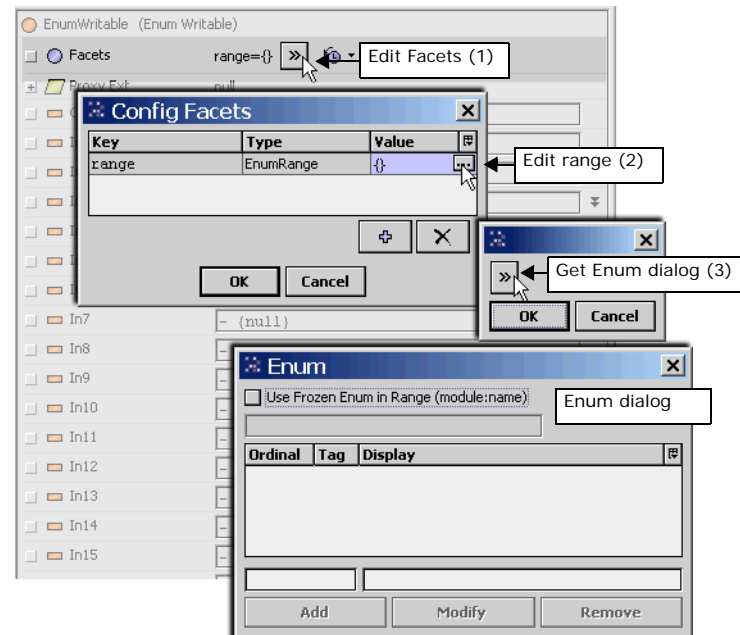
Note: For string-type points (*StringPoint*, *StringWritable*), facets typically have little practical application. By default, the Facets slot is empty for string-type points.

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.

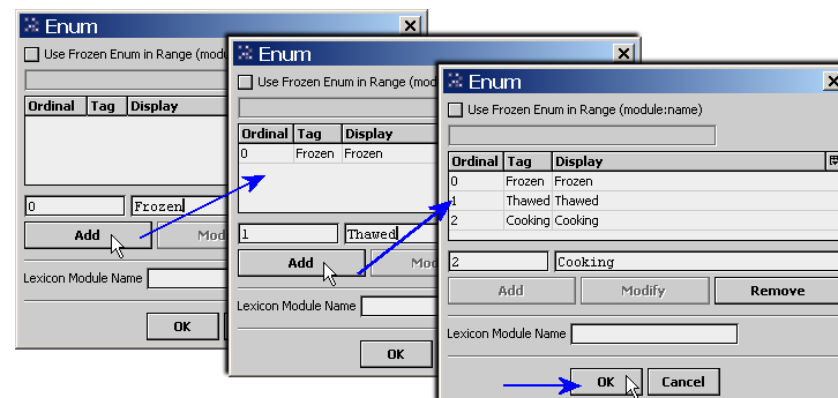
If you add an enum point from the control palette, its Facets slot has a *blank* range entry. Until you edit this facet and supply the ordinal-tag values, it can display only integer values. As shown in [Figure 3-12](#), a special **Enum** dialog appears when you edit range facets.

Figure 3-12 Producing Enum dialog for enum point “range” facets



In the **Enum** dialog when adding an entry, the **Add** button becomes available after you enter an integer value in the left side Ordinal box, as shown in [Figure 3-12](#). 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.

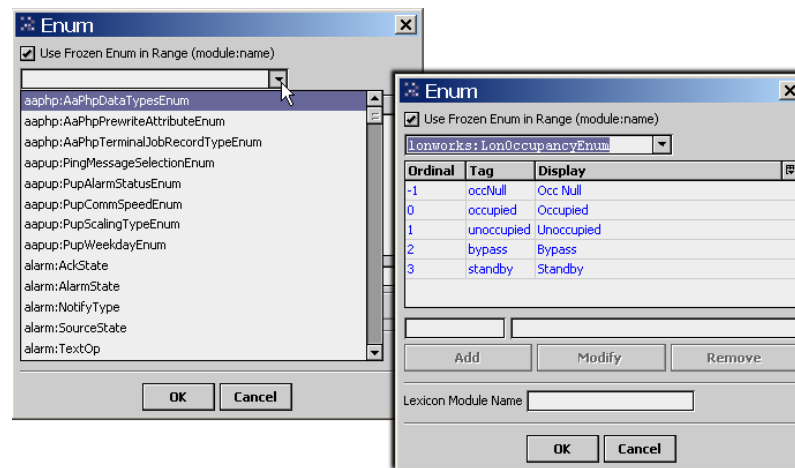
Figure 3-13 Type unique integer value in Ordinal box and associated text in Tag box



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 (see [Figure 3-14](#)).

Figure 3-14 Frozen enum selection in Enum dialog



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

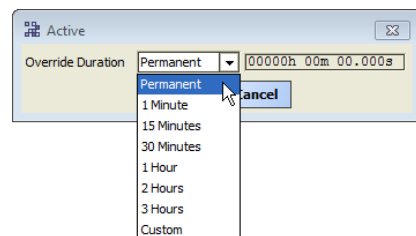
For some points with actions (see [“About point actions”](#) on page 3-3), facets also affect availability in the point’s action (command) menu, as follows:

- EnumWritable**
 Upon an override or emergency action, a secondary drop-down selection dialog 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 dialog 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”](#) on page 3-10.

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 3-15).

Figure 3-15 Default override action menu for writable control point

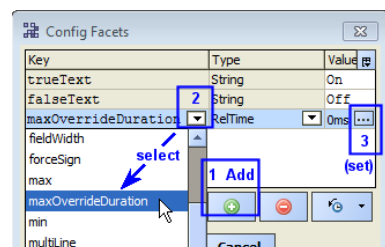


If needed, change this by adding a “maxOverrideDuration” facet (choosing type baja:RelTime), with specified duration time, to either or both:

- [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 3-16 Config Facets editor when picking maxOverrideDuration

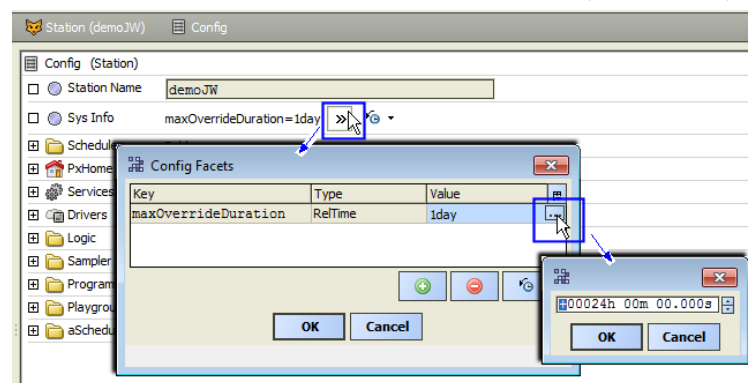


When a writable point is limited by a maxOverrideDuration facet, its action menu adjusts to show the allowable range. See “Action menu examples” on page 3-11.

For details about editing facets, see “Accessing and editing facets” on page 3-7.

Config, Sys Info property The “Sys Info” property of the station’s root Config component has a facets control, shown in the Config component’s property sheet (Figure 3-17).

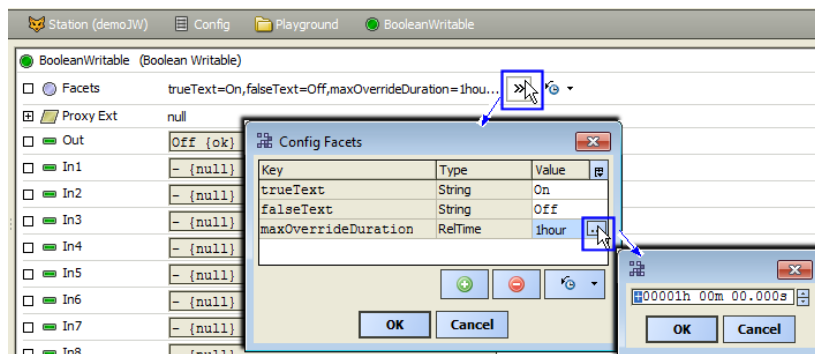
Figure 3-17 Global maxOverrideDuration facet added to Sys Info property of station’s Config component



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.

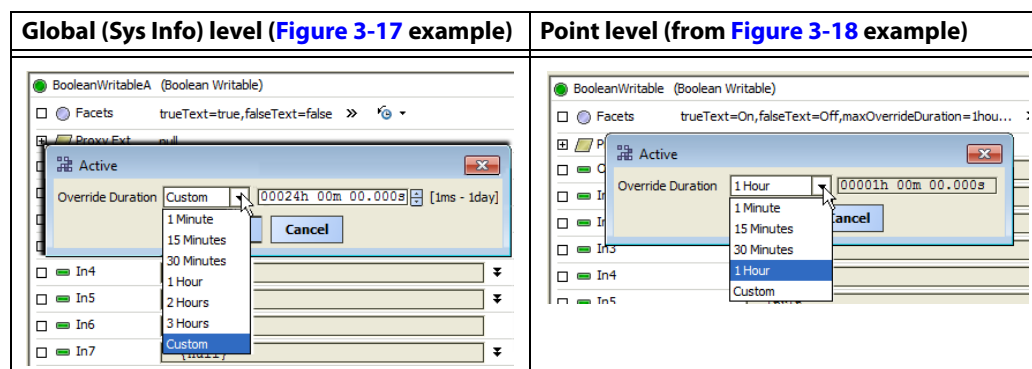
Figure 3-18 Added maxOverrideDuration facet added at point level (overrides global setting)



As shown in the Figure 3-18 example, 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.

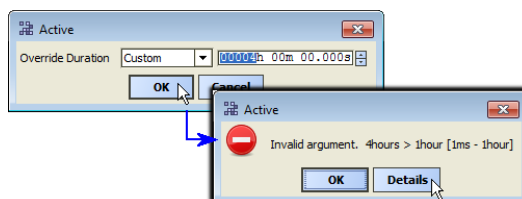
Action menu examples Example action menus from writable points with a maxOverrideDuration facet in effect are shown in Figure 3-19.

Figure 3-19 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 in Figure 3-20.

Figure 3-20 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.

The following topics provide more details on extensions:

- [Extension process overview](#)
- [Types of point extensions](#)
- [About the proxy extension](#)
- [About control extensions](#)
- [About alarm extensions](#)
- [About history extensions](#)

Note: *Extensions also apply to most kitControl objects. See the kitControl Guide section “Extensions and kitControl components” for more details.*

Extension process overview

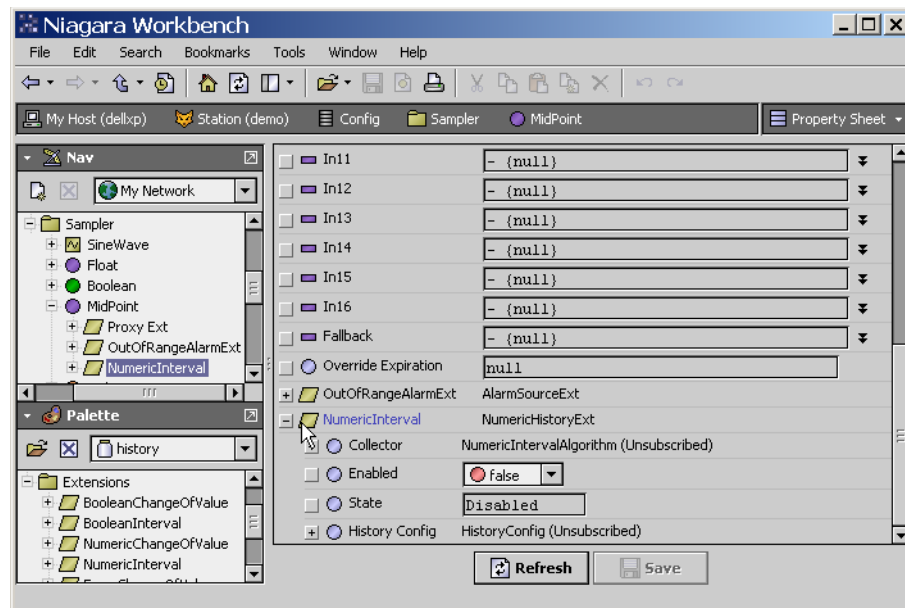
Extensions are found in several palettes, including alarm, control, history, and kitControl.

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 in [Figure 3-21](#).

Figure 3-21 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” on page 3-23.*

Types of point extensions

Point extensions include the *standard* proxy extension (one for each point, see “[About the proxy extension](#)” on page 3-13) and also *optional* extensions.

Optional extensions are in three main categories:

- **control extensions**
from control palette. See “[About control extensions](#)” on page 3-13.
- **alarm extensions**
from alarm and kitControl palette. See “[About alarm extensions](#)” on page 3-14.

- **history extensions**
from history palette. See “About history extensions” on page 3-16.

About the proxy extension

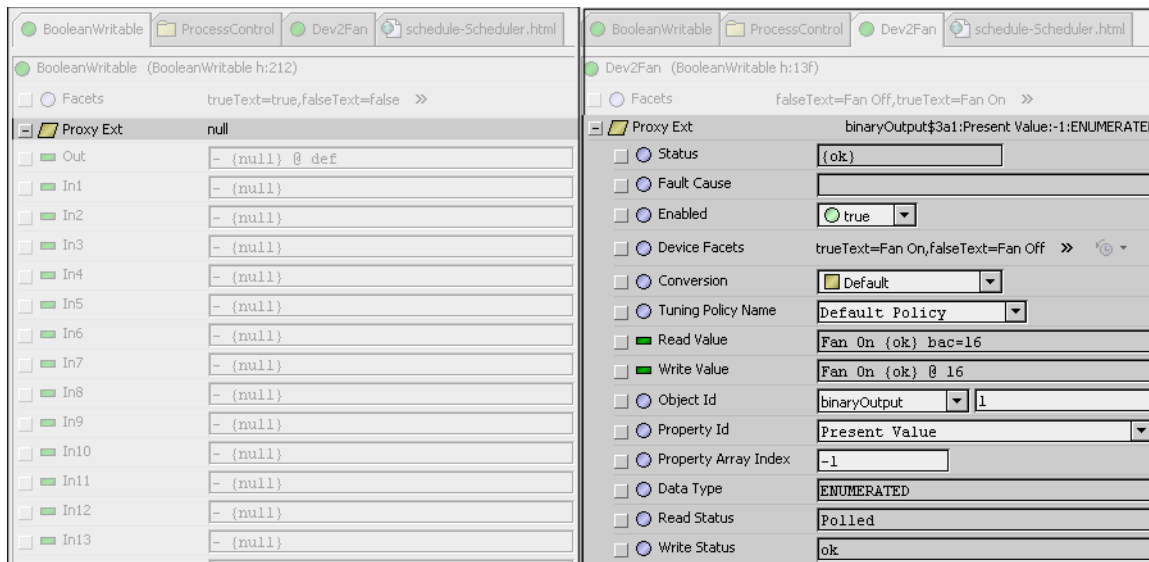
Each point has a proxy extension (Proxy Ext), a frozen property. The proxy extension is *important*—it indicates how the point’s data originates, including details specific to the parentage of the point’s network and communications (driver).

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.

Figure 3-22 compares property sheet views of the proxy extension properties for two BooleanWritable points, on the left a control point, on the right a BACnet proxy point.

Figure 3-22 Proxy extension for control point (null) and a Bacnet proxy point.

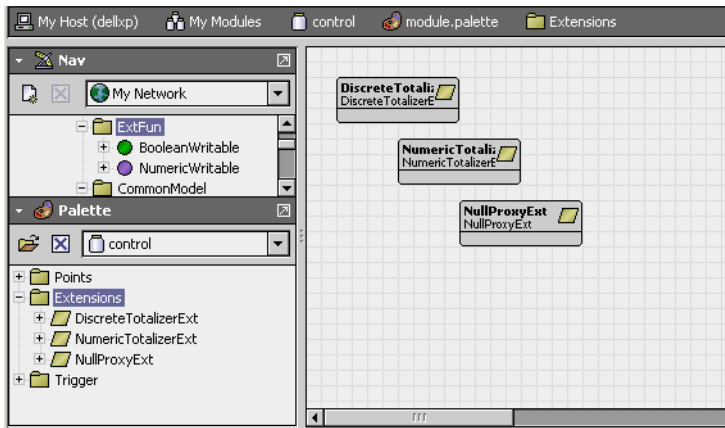


For more details, refer to Drivers Guide sections “About proxy points” and “ProxyExt properties”.

About control extensions

Control extensions are found in palette **control: Extensions** (Figure 3-23). As needed, you can add them to points along with alarm and history extensions.

Figure 3-23 Control extensions



Control extensions perform additional processing on a point’s received value. There are relatively few types of control extensions. See “Types of control extensions” on page 3-14.

Types of control extensions

Table 3-2 lists all available control extension types and the applicable point parents.

Table 3-2 Control extension types

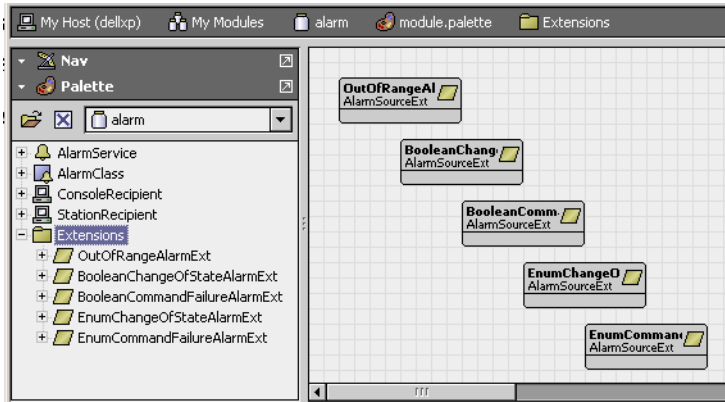
Control extension type (palette:Folder)	Applies to point types		What it does
	(read-only)	Writable	
DiscreteTotalizerExt (control:Extensions)	BooleanPoint	BooleanWritable	Accumulates runtime and change of state (COS) count. Extension actions permit resetting (zeroing) the runtime and COS count.
	EnumPoint	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” on page 3-13)		Provides methods to driver communications.

About alarm extensions

Use an alarm extension for any point you wish to monitor for offnormal values, and show alarm indication when a limit or value is met or exceeded. In an alarm extension, you also specify whether an alarm *annunciation* should occur upon alarm transitions, plus many other parameters.

Find the alarm extensions in palettes **alarm: Extensions** (Figure 3-24) and **kitControl: Alarm**.

Figure 3-24 Alarm extensions



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. For more details, see "About alarm extension properties" on page 5-3.

Types of alarm extensions

Table 3-3 lists all alarm extension types and the applicable point parents.

Table 3-3 Alarm extension types

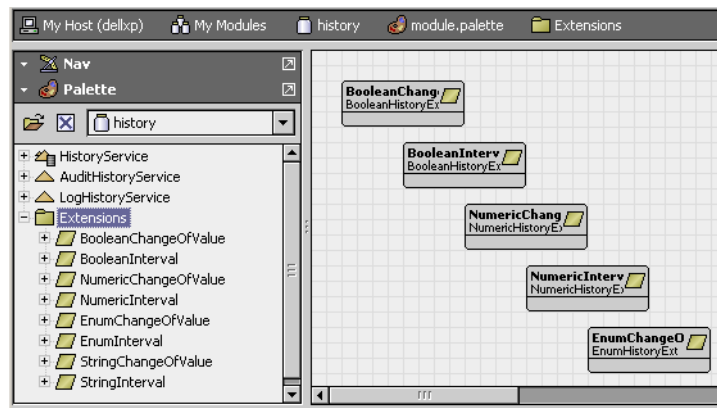
Alarm extension type (palette:Folder)	Applies to point types		General description
	(read-only)	Writable	
OutOfRangeAlarmExt (alarm:Extensions)	NumericPoint	NumericWritable	Provides alarming based upon numeric alarm high and low limits. Includes configurable deadband.
	—	any object with single numeric Out, e.g. kitControl: Math object "Add"	
StringChangeOfValueAlarmExt (alarm:Extensions) <i>Note: AX-3.6 or later required</i>	StringPoint	StringWritable	Provides alarming based upon either <i>inclusion</i> or <i>exclusion</i> of the entered string value (or "regular expression", as needed).
	—	any object with single String Out, e.g. kitControl: String object "StringSubString"	
BooleanChangeOfStateAlarmExt (alarm:Extensions)	BooleanPoint	BooleanWritable	Provides alarming based upon one of two possible values (state) as an alarm condition.
	—	any object with single Boolean Out, e.g. kitControl: Logic object "And"	
BooleanCommandFailureAlarmExt (alarm:Extensions)	—	BooleanWritable	Provides alarming based upon mismatch between commanded value and actual (sensed) value. Extension has feedback-Value input property for linking.
EnumChangeOfStateAlarmExt (alarm:Extensions)	EnumPoint	EnumWritable	Provides alarming based upon one of multiple possible values (state) as an alarm condition.
EnumCommandFailureAlarmExt (alarm:Extensions)	—	EnumWritable	Provides alarming based upon mismatch between commanded value and actual (sensed) value. Extension has feedback-Value 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.
LoopAlarmExt (kitControl:Alarm)	—	LoopPoint	Sliding alarm limit for LoopPoint 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), e.g. 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), e.g. kitControl: Logic object "And"	

About history extensions

Add a history extension to any point for which you want to log historical data, that is 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 3-25).

Figure 3-25 History extensions



Each history extension has various properties in two major groups:

- **Collector**
Determines how or when data is collected, such as active time, and (if applicable) either change tolerance or the collection interval time.
- **History Config**
Determines how the data is stored and accessed in the system, such as history name, capacity, and full policy.

For more details, see “[Configure history extensions](#)” on page 4-25.

Types of history extensions

[Table 3-4](#) lists all history extension types and the applicable point parents.

Table 3-4 History extension types

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

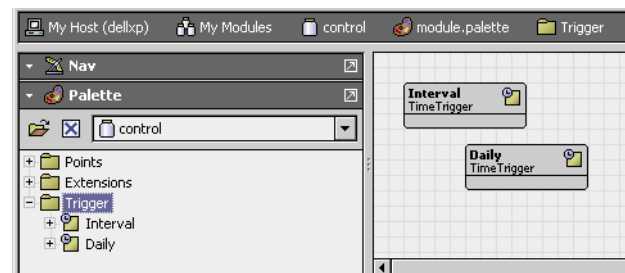
History extension type	Applies to point types		General description
	(read-only)	Writable	
StringChangeOfValue	StringPoint	StringWritable	Collects upon each change of string value or status.
	—	any object with single String Out	
StringInterval	as above	as above	Collects upon a repeating time interval,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

About the Update History Id action 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”](#) on page 4-26 for more information about history naming.

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 3-26 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”](#) on page 7-23.

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” on page 1-20 and “Lexicon Editor view” on page 8-5.

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).

Table 3-5 Status flag types

Type	Default Colors, Example	Meaning
alarm	white text, red background 65.0 °F	Point currently has a value in an alarm range, as defined by property in its alarm extension.
fault	black text, orange background 208.8 °F	Originates from a proxy point only. Typically indicates a configuration or licensing error. If it occurs after normal operation, it may indicate a “native fault” in device, or the point’s parent device has a fault status.
overridden	black text, magenta background 72.0 °F	Current point control is from an action, meaning a user-invoked command at either priority level 8 (override) or priority 1 (emergency).
disabled	gray text, light gray background 79.0 °F	Originates from a proxy point only. Point (or its parent device or network) has been manually disabled (property enabled = false).
down	black text, yellow background 68.4 °F	Originates from a proxy point only. Driver communications to the parent device are currently lost, based upon the device status (Monitor) configuration for that network.
stale	black text, tan background 73.4 °F	Originates from a proxy point only. Driver communications have not received a requested response for this data item within the configured times (Tuning period).
null	(no color indication)	Current point control has entered a null state, vs. a specific value and priority level. Typical to fallback operation for a writable point. Note: If linking a null status Out to a “simple data” slot, the point’s “null value” is processed. See the NiagaraAX kitControl Guide 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.

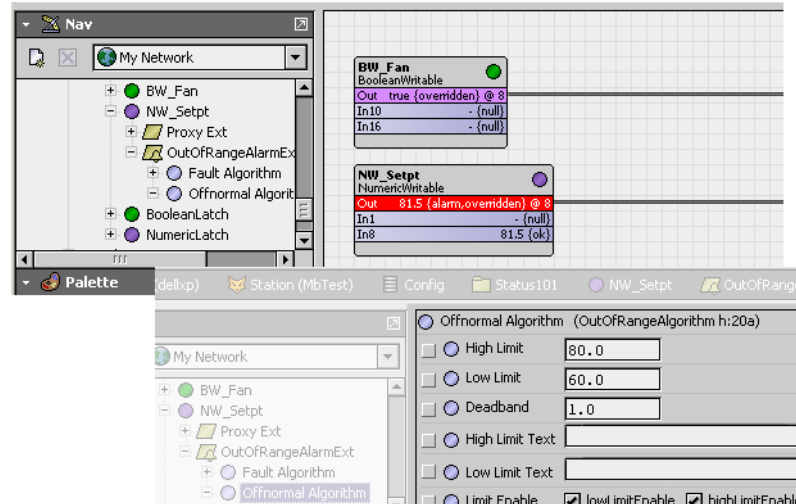
Simple control point status

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 “Fall-back” value. See “[About “isValid” status check](#)” on page 3-21.

See [Figure 3-27](#) for examples of override and alarm status in simple control points.

Figure 3-27 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:

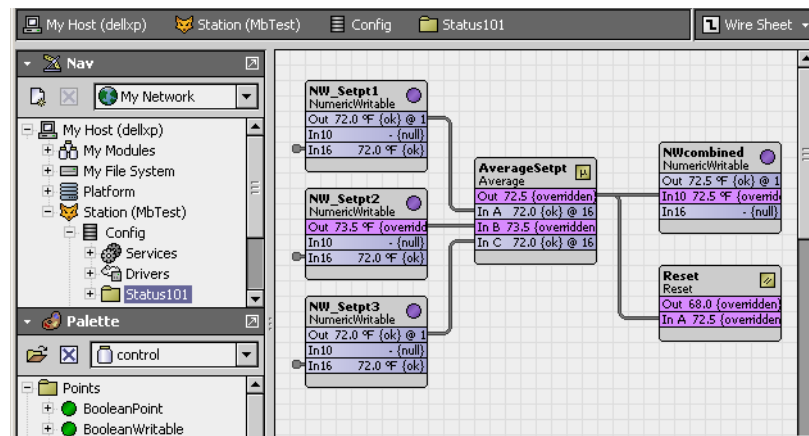
- **disabled**
- **fault**
- **down**
- **alarm**
- **overridden**

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” on page 3-21.

As an example of status propagation, some number of NumericWritable points are used to establish setpoints, and you link them all to a Math: Average object for downstream zone control. In turn, the Average object feeds a Math: Reset object. Both math objects have “override” enabled in their “propagate-Flags” property. A user issues an override (action) to one of the NumericWritable points, to override a setpoint (Figure 3-28).

Figure 3-28 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: *Status propagation did not occur in Niagara r2, and is a new configuration option in NiagaraAX. 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”](#) on page 3-21.

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”](#) on page 3-3). 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”](#) on page 3-4. 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”](#) on page 3-5.

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”](#) on page 3-3. Priority level 6 in a BooleanWritable is reserved for minimum on/off times. See [“About minimum On and Off times”](#) on page 3-22.

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 AX writable points are strictly a station-centric NiagaraAX 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](#)” on page 3-22.

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 (composed) component. This can simplify linking and promote reuse of control logic.



Caution

Composites have associated issues—see “[Composite issues](#)” on page 3-26. 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](#)” on page 3-23.

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:

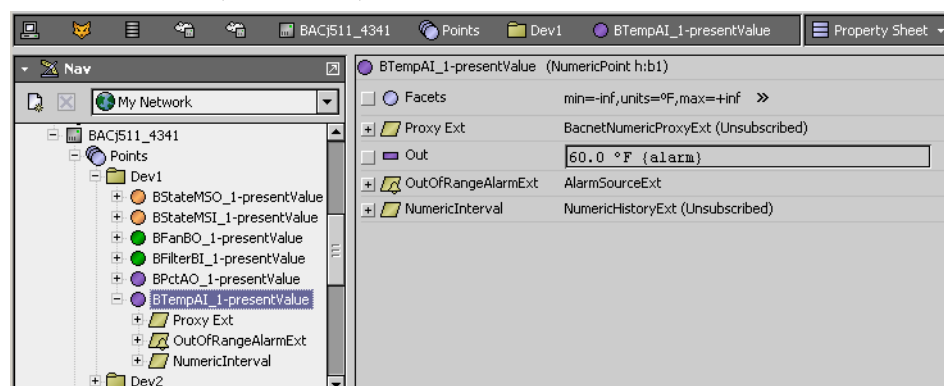
- [Point-level composite](#)
- [Folder-level composite](#)

Point-level composite

[Figure 3-29](#) shows a proxy NumericPoint representing a space temperature value that has two extensions:

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

Figure 3-29 Property sheet of proxy point with two extensions

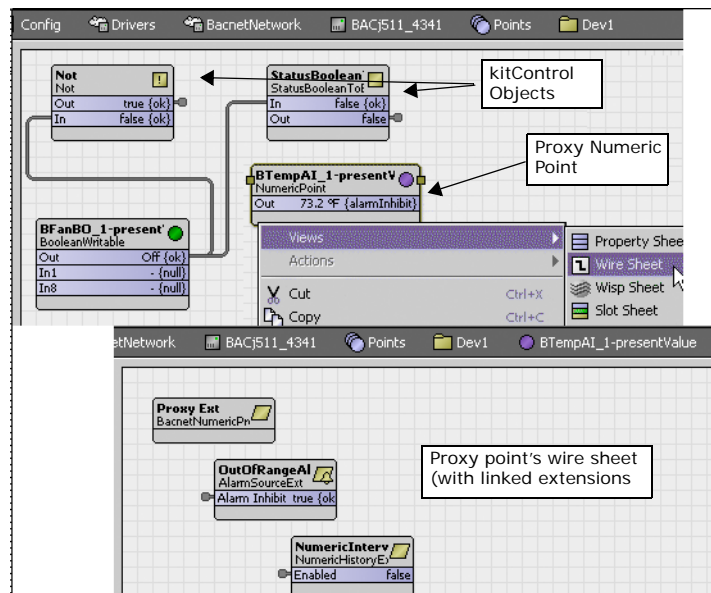


In this example, when another system-related BooleanWritable (representing the system fan) has a false (Off) status, it is desired that this temperature point be:

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

To achieve these “disable” functions, you must link the controlling source fan out to a slot of each *extension* (visible in point’s wire sheet view, but not in the parent wire sheet for the point itself). Furthermore, other kitControl objects needed. Without making a composite, the necessary objects and links may appear as shown in [Figure 3-30](#).

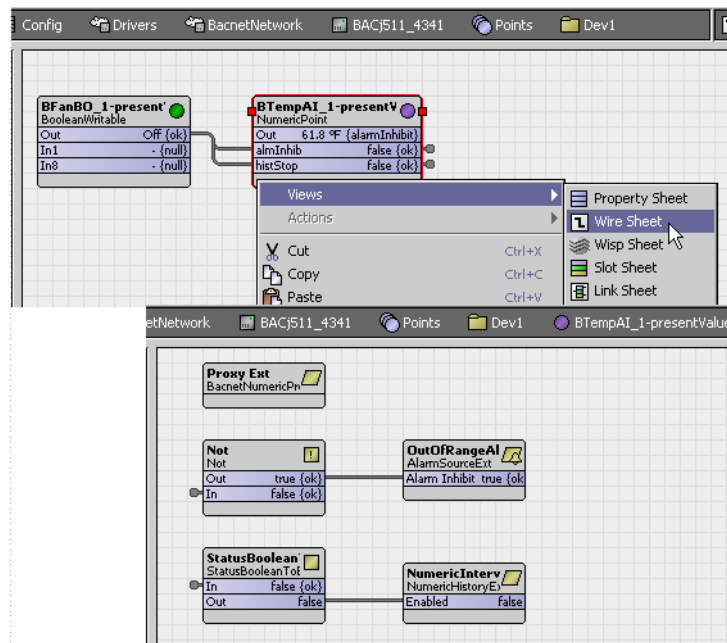
Figure 3-30 Proxy point example without composite



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.

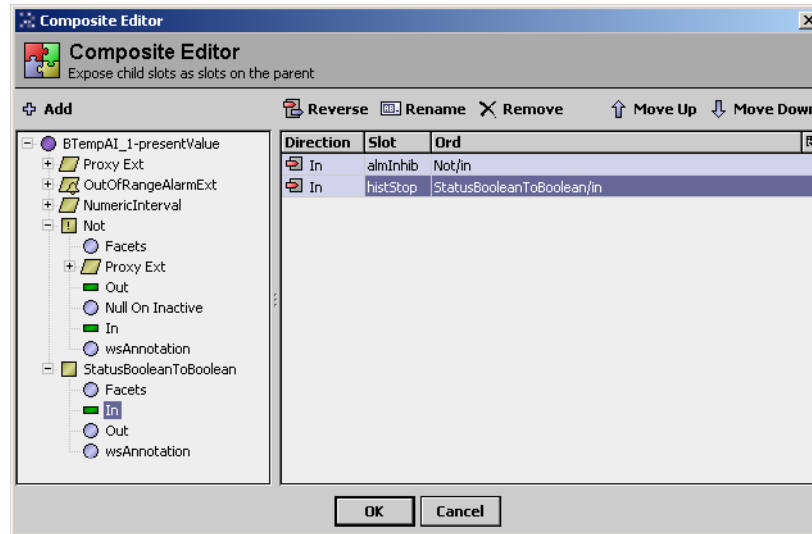
By making a composite of the NumericPoint (acting as the container for both the extensions plus the additional kitControl objects) you can simplify reuse and clarify available links. [Figure 3-31](#) shows the now-composited NumericWritable linked to the controlling BooleanWritable, and the wire sheet view of the NumericWritable that contains the needed kitControl objects.

Figure 3-31 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 as shown in [Figure 3-32](#).

Figure 3-32 Composite Editor for example NumericPoint

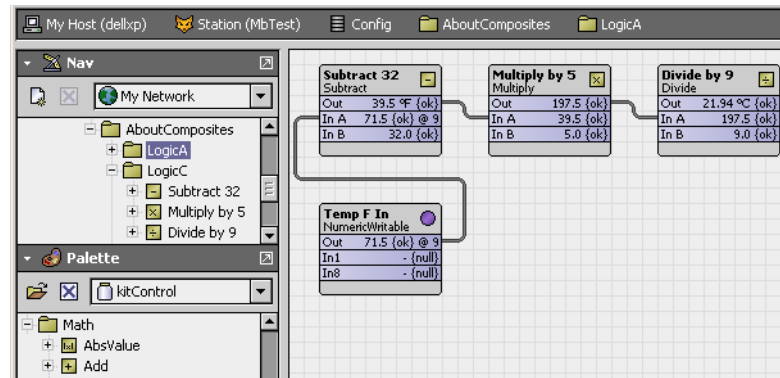


Folder-level composite

Note: Please see the related [Caution](#) on page 23.

Figure 3-33 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 3-33 Simple example of folder before compositing

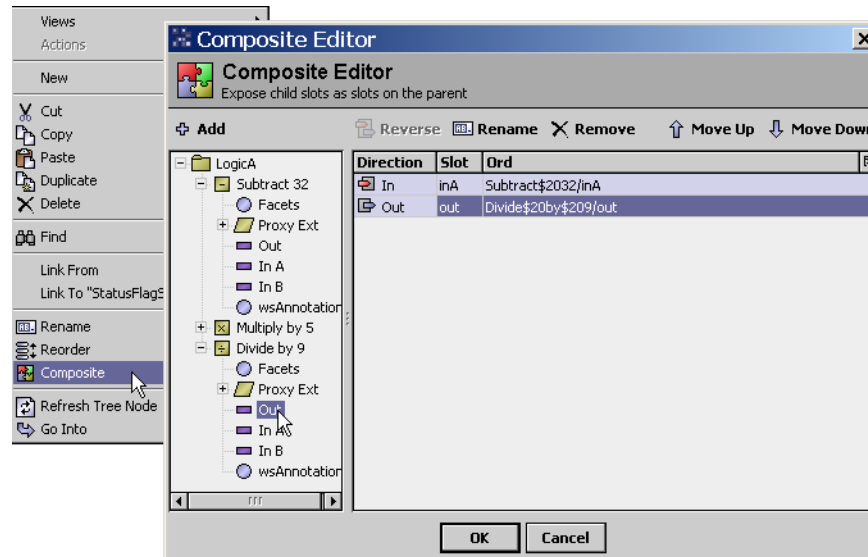


If this application was needed later, you could copy all 3 linked objects again and insert in another (perhaps already crowded) wire sheet. However, the middle “Multiply” object reveals an intermediary result that is distracting. Or, you could just create a new subfolder with only the 3 linked objects 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 (Figure 3-34). The Composite Editor lets you expand the tree of all contained components and “expose” those items of interest.

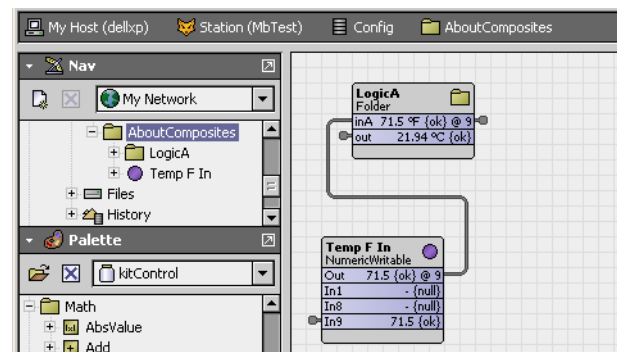
Figure 3-34 Launching and using the Composite Editor



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. [Figure 3-35](#) shows the test NumericWritable now linked to the composited LogicA folder.

Figure 3-35 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).

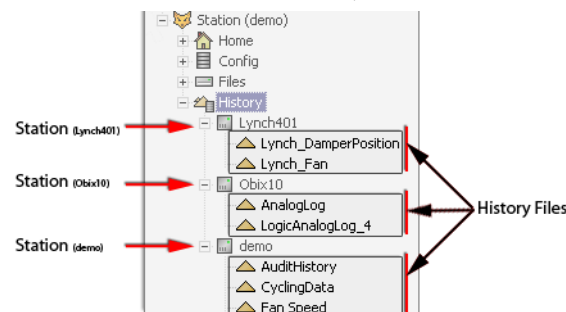
In addition, each item exposed in a composite represents a link, where each link consumes some small amount of station resources. If used excessively, composites could noticeably reduce the total capacity of the station.

CHAPTER 4

About Histories


In Niagara, a data log is referred to as a **history**. Histories are ordered collections of timestamped records. A single “history” is a collection of specific data values from a component within any station - local or remote. Histories are organized by their source station (device), as shown in [Figure 4-1](#).

Figure 4-1 Histories organized by station



The following topics are relevant to understanding histories:

- **History Services**
The History Service, the Audit History service and the Log History service provide support for logging data in a Niagara station. In order to provide database support for histories in a Niagara station, the station must contain the History Service. Refer to [“About the history service”](#) on page 4-2 for details about the History Service. Refer to [“About the audit history service”](#) on page 4-4 and [“About the log history service”](#) on page 4-5 for information about those services.
- **History ORD scheme**
Once you have a history service running, you can access histories that you create in the database using the “history” ORD scheme. The unique history scheme name “history” and each unique **history ID** provide identification for the individual histories. All history collections are identified by this unique id. For information on using the ORD scheme to access individual histories, refer to [“About ORDs”](#) on page 1-16.
- **History space**
History space provides a means for viewing and working with histories in the history database. History space is visually represented in Workbench as a node in the nav tree and may be accessed by using the nav tree or by using the Open ORD dialog box. Views on the History space include the following: History Chart Builder, Database Maintenance, Nav Container View. For more information about history space and history space views, refer to [“Types of history space views”](#) on page 4-5.
- **History views**
History views present history information in various formats for both analysis and editing. Refer to [“Types of history views”](#) on page 4-10.
- **History logging process**
Using histories involves a process of collecting, storing and archiving data. You can configure the history collection process to collect the data that you need and store the history records where you want them - locally or remotely. This process is described in more detail in [“About the history process”](#) on page 4-22.
- **History Configuration**
History configuration includes working with parameters such as ID, history source, timezone, record type, and more. For more details on configuration, refer to [“Configure history extensions”](#) on page 4-25.

- **History data editing**
You can edit and filter the history data in workbench using the history editor view (described in [“About the history editor view”](#) on page 4-15). The editing process is described in more detail in [“About editing history data”](#) on page 4-28.
- **History Grouping**
Starting with AX-3.5, this component allows you to use history properties to provide custom organization and display of history space contents. See [“About history grouping”](#) on page 4-30 for more details.
- **History Nav Shortcuts**
Available starting with AX-3.5, history nav shortcuts  provide convenience navigation links to histories. See [“About history nav shortcuts”](#) on page 4-32 for more details.

Types of History Services

History services are available in the History Palette, as shown in [Figure 4-2](#).

Figure 4-2 History services in the history palette



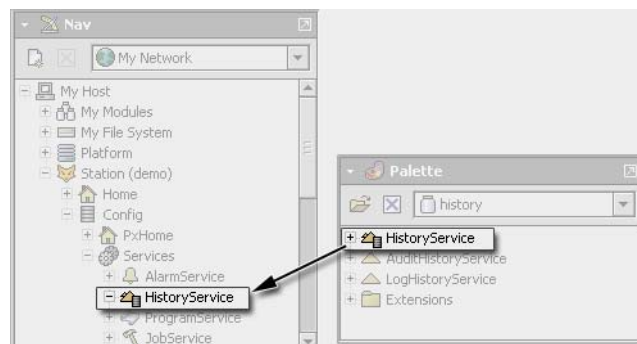
Each service is described in the following sections:

- [“About the history service”](#) on page 4-2
- [“About the audit history service”](#) on page 4-4
- [“About the log history service”](#) on page 4-5

About the history service

To use histories, each station must contain a single history service that provides http access to all of the histories in a station. This service is responsible for creating the history database, as well as enabling the collection and storage of histories in the database. If you do not have the history service in your active station, you can add it by dragging and dropping a copy from the History Palette, as shown in [Figure 4-3](#).

Figure 4-3 Adding the history service



The history service manages histories in ways that are not always visible to the user, including the following:

- **identifies histories**
when the service is started (normally this is whenever the station is started) all existing histories are added to the service.
- **life cycle management**
handles history management functions, such as creation and deletion of individual histories.
- **namespace**
maintains the naming convention (namespace) for all histories.
- **configuration**
maintains a global default configuration for the histories.

The history extension manager and the history property sheet are two views of the history service that provide ways to work with history extensions. Refer to the following sections for more information about these views:

- [About the history extension manager view](#)
- [About history service property sheet](#)

About the history extension manager view

As the default view of the history service, the history extension manager shows all history extensions in the station and displays their current status. Using this view as both a management and navigational tool, you can double-click on any entry-row to go directly to the property sheet view of that extension. This table has the standard table features. Use the **Table Options** menu in the top right corner of the history table to modify the table view or to export the data in the view, as desired. Refer to [“Table controls and options”](#) on page 2-18 for information about using the tabular view, including the **Table Options** menu.

Figure 4-4 History extension manager

Point	Extension	History Name	Status
/Sampler/FloatWritable	ChannelOfValue	My FloatWritable History	{ok}
/Sampler/FloatWritable	Views	atWritable1	{ok}
/Sampler/FloatWritable	Actions	atWritable2	{disabled}
/Sampler/NumericWritable		numericWritable	{disabled}
/Sampler/NumericWritable		3	{ok}

Go To Point
Go To History
Enable Collection
Disable Collection
Rename History

You can also **Enable**, **Disable**, **Rename**, or (starting in NiagaraAX-3.4) **Edit System Tags** for any collection from this view by selecting the desired history extension in the table and using the History Ext Manager menu, popup menu, or toolbar icons, as described in the sections listed below:

- [“About the History Ext Manager menu”](#) on page A-8
- [“About the history extension manager popup menu items”](#) on page A-11
- [“About the history extension manager toolbar icons”](#) on page A-15

Note: If a history is disabled, its row is dimmed with a gray background in the history extension manager view.

About history service property sheet

The history service property sheet, shown in [Figure 4-5](#), includes the following properties:

- **Max Open Time**
this is an editable parameter for setting the amount of time that the history database may remain open without being accessed *before* it is subject to being closed by the **Close Unused Histories** action. If the history is accessed more frequently than the Max Open Time, it is not closed when **Close Unused Histories** action is invoked.
- **History Groupings**
Starting in AX-3.5, this property is available for organizing histories according to history properties that you can designate and enable using the History Group component.

Figure 4-5 History service properties

History service actions

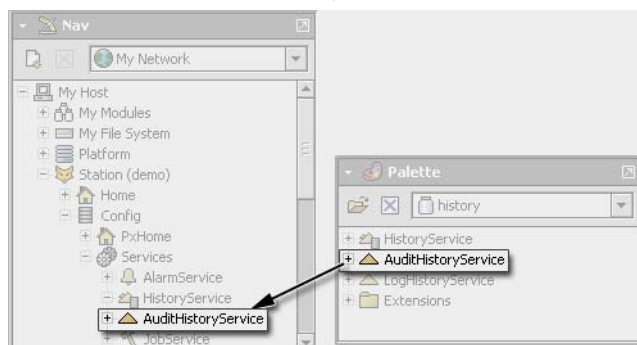
History service actions are available from the **popup** menu when you right-click the HistoryService node in the nav side bar pane. Available actions include:

- **Save Db**
this action initiates a save of all histories to the history database.
- **Close unused histories**
this action closes any histories that have not been accessed within the Max Open Time (this time is set in the history service property sheet).

About the audit history service

The Audit History Service keeps a history of the *changes* that are made by users. If this service is enabled, it registers itself as the auditor for the system when the service is started. The easiest way to add the audit history service (if not already present) is through a simple drag and drop method from the History Palette (under the HistoryService) to the services area of the nav side bar pane, as shown in [Figure 4-6](#).

Figure 4-6 Adding audit history service



When enabled, the Audit History Service logs all property changes and all actions taken on a component, such as the following:

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

You can view the Audit History Service properties, enable or disable the Audit History Service, and set record capacity parameters from the Audit History Service Property Sheet, as shown in [Figure 4-7](#).

Note: Starting in AX-3.5, schedule changes are tracked in detail, as shown below.

Figure 4-7 Audit history service properties

AuditHistory (Audit History Service)	
<input type="checkbox"/> Enabled	<input checked="" type="checkbox"/> true
<input type="checkbox"/> History Config	History Config
<input type="checkbox"/> Id	/demo/AuditHistory
<input type="checkbox"/> Source	station: h:45b2
<input type="checkbox"/> Time Zone	America/New_York (-5/-4)
<input type="checkbox"/> Record Type	history AuditRecord
<input type="checkbox"/> Capacity	Record Count 500 records
<input type="checkbox"/> Full Policy	Roll
<input type="checkbox"/> Interval	irregular
<input type="checkbox"/> System Tags	
<input type="checkbox"/> Last Record	24-Nov-09 1:15 PM EST[admin] operation: baja:Sl...
<input type="checkbox"/> Timestamp	24-Nov-2009 01:15:17 PM EST
<input type="checkbox"/> Operation	Added
<input type="checkbox"/> Target	/Logic/HousingUnit/Schedule/schedule/wee
<input type="checkbox"/> Slot Name	time
<input type="checkbox"/> Old Value	
<input type="checkbox"/> Value	4:00 AM to 9:00 PM, true
<input type="checkbox"/> User Name	admin

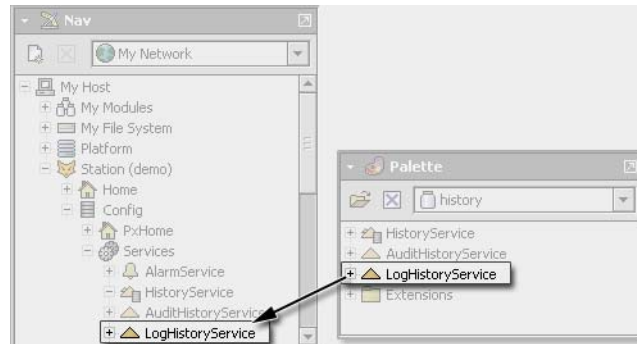
Refer to [“About user audits”](#) on page 6-12 for more details.

About the log history service

The Log History Service keeps a History of Niagara log records when it is enabled. This service maintains a buffered history (“LogHistory”) of *some* of the messages seen in the station’s standard output. This can be very helpful when you are troubleshooting problems with a station. If a station has Log History Service enabled, you can check the log history for recent error messages. Refer to the *Platform Guide* section “Application output” for more information about output messages.

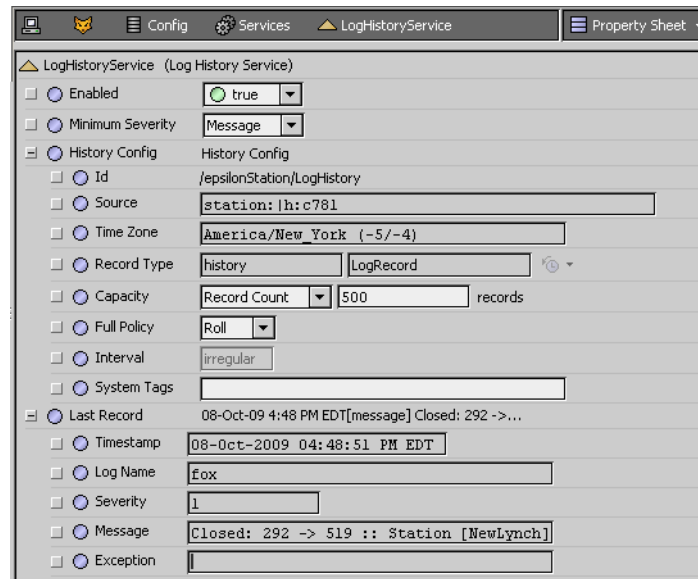
The Log History Service is available in the History Module. The easiest way to add this service (if not already present) is through a simple drag and drop from the History Palette (under the HistoryService) onto the services area of the nav side bar pane, as shown in [Figure 4-8](#).

Figure 4-8 Adding log history service



You can edit the Log History Service properties and set configuration properties in the property sheet, as shown in [Figure 4-9](#).

Figure 4-9 Log history service properties



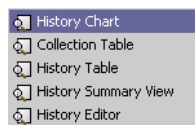
In addition to the standard HistoryConfig properties, the following properties are available for editing:

- **Enabled**
select the `true` option to enable or the `false` option to disable the Log History Service.
- **Minimum Severity**
you can choose the level of output that you want to record by setting the log level Minimum Severity property. This property is the lowest-level station output message that you want to log. The default log level for Minimum Severity is `Error`. You can change this to `Warning`, `Trace`, or `Message`.

Types of history space views

The history space views shown in [Figure 4-10](#) and listed below are particularly helpful in working with multiple histories:

Figure 4-10 History space views



- **History chart builder view**
This is the default view of the History Space node (in the nav tree). Use this view to build any one of several types of chart options from the data that is stored in one or more histories. See [“About the history chart builder view”](#) for details.
- **Database maintenance view**
Use this view to clear records or delete histories. Refer to [“About the database maintenance view”](#) on page 4-8 for more information about this view.
- **Nav container view**
Use this view to display all histories in the station. See [“About the nav container view”](#) on page 4-9 for more details about this view.

About the history chart builder view

The chart builder view displays in Workbench, in all Workbench (Wb)Web profiles, and with NiagaraAX-3.2 and later, the Default Hx, and Basic Hx profiles, as well.

Note: The following description discusses the Chart Builder view in terms of the Wb Web and the desktop Workbench display profiles. Stations using NiagaraAX-3.2, can display the Chart Builder view in the browser using the Default and Basic Hx (non-Java applet) profile views. Views displayed using Hx look different but behave very similarly to the Workbench view. All field descriptions apply to both views.

The Chart Builder Workbench view (also Wb Web profile view) is shown in [Figure 4-11](#), the Chart Builder Hx view is shown in [Figure 4-13](#). Using this view, you can select the histories (or in AX-3.5 and later History Nav Shortcuts), title, and configure the charts that you want to generate.

Figure 4-11 Chart builder view (NiagaraAX-3.1 and later)

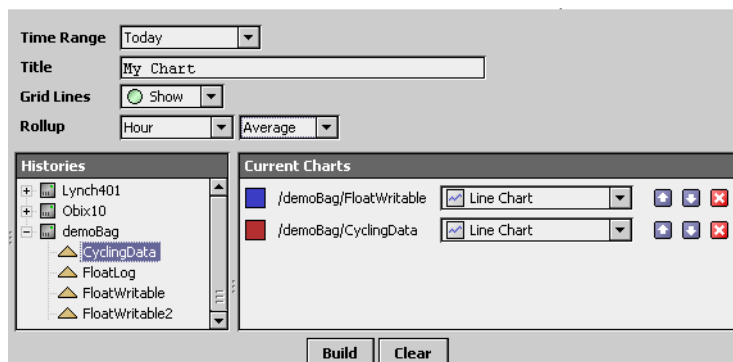


Figure 4-12 Chart builder view (NiagaraAX-3.5 and later using History Nav Shortcuts)

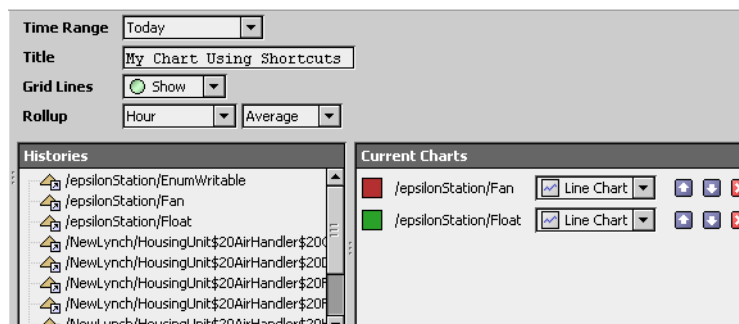
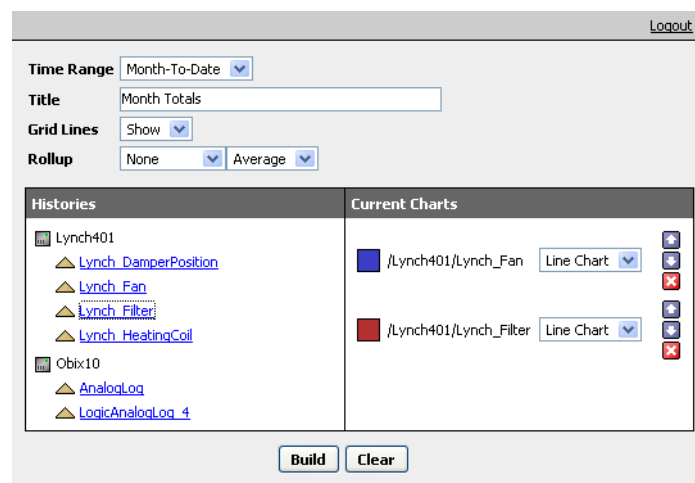


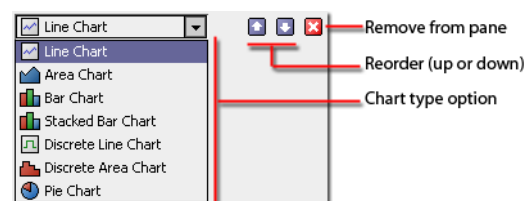
Figure 4-13 Chart builder view (NiagaraAX-3.2 and later Hx Web profile)



The display of the Chart Builder view is divided into three primary areas:

- **Display configuration fields**
 - Time Range
Select a time parameter option from the list, including an option that allows you to set a specific time range using the **Edit Time Range** dialog box.
 - Title
Type a title for your chart in this text field.
 - Grid Lines
Select Show or Hide to show or hide grid lines on the history chart.
 - Rollup
Rollup (or Rollup Interval) is an interval of time that is used to determine what (and how) data is presented in your chart. Each point displayed, using the rollup, represents a designated time interval before the specified plot time. A rollup value of 1 hour will present data at a granularity level of every one hour, while a rollup value of 15 minutes will show data for every 15 minutes of logged data. Rollup options are:
 - Average
This option plots the average value for the selected rollup period.
 - Min
This option plots the minimum value for the selected rollup period.
 - Max
This option plots the maximum value for the selected rollup period.
 - Sum
This option plots the total of the values in the selected rollup period.
- **Histories pane**
This is the lower left area of the Chart Builder view. It displays all histories that are available in your local station or any station histories that you import by means of the Niagara network or other network driver (for example, BacnetNetwork). Histories are grouped under the station by station name. Double-click (Wb Web profile) or click (Hx Web profile) on a history name to copy it to the Current Charts pane.
Note: The histories that are displayed in this pane are the same histories that are displayed under the history space node in the Workbench nav sidebar.
- **Current Charts pane**
This pane displays the histories that are selected to be plotted. For each history, you may select the type of chart to generate, using the chart type option list, as displayed in Figure 4-14.
Note: NiagaraAX-3.2 has more chart type options than earlier versions.
Adjacent to the histories in this pane are icon-type controls that allow you to reorder histories in the pane or remove histories from the pane.

Figure 4-14 Current chart pane options and controls



• **Control buttons**

The following control buttons are located at the bottom of the Chart Builder view:

- **Build** button
Click this button to build the chart using the histories that are in the selected Current Histories pane.
- **Clear** button
Click this button to remove all histories from the Current Histories pane.

Figure 4-15 shows an example of chart that displays two histories.

Figure 4-15 Two different history line charts using chart builder view (Wb and Hx Web profiles)

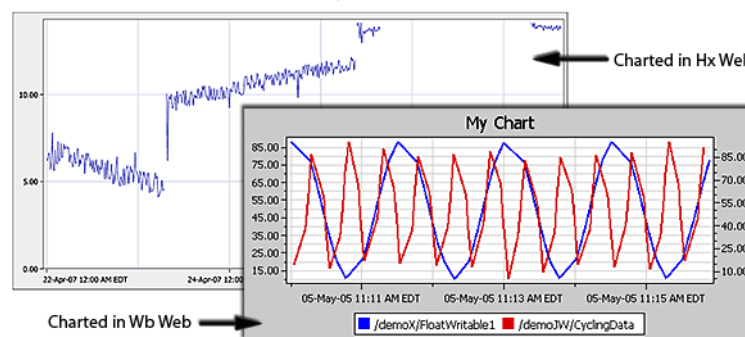
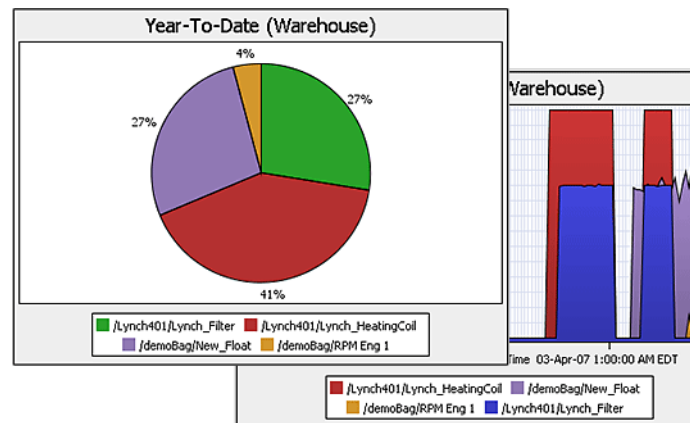


Figure 4-16 History pie chart and area chart using chart builder view



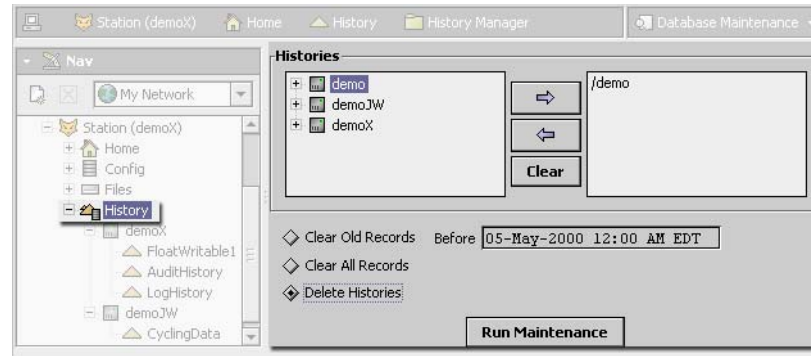
About Time Zones and the Chart Builder view

When charting multiple histories that include different timezones, the Chart Builder uses a “zoneless” time range configuration so that it can plot each history with reference to its own timezone. This means that the resulting charts are aligned by local time. For example, if you select a time range of 8:00 AM through 5:00 PM for two histories—one in EST and another in CST—then the values at 8:00 AM align so that the 8:00 AM values may be visually compared.

About the database maintenance view

The database maintenance view is shown in Figure 4-17. Using this view, you can clear records and delete complete histories from your history database. Select the database maintenance view by selecting the history space node in the nav sidebar and using the view selector menu or the popup menu to choose the database maintenance menu item.

Figure 4-17 Database maintenance view



The left side of the **histories** area contains the **available histories** window. This window displays all histories that are available in your local station or any station histories that you import by means of the Niagara network or other network driver (for example, BacnetNetwork). Histories are grouped under the station by station name.

Note: *The available histories are the same histories that are displayed under the history space node in the nav sidebar.*

The right side of the **histories** area contains the targeted histories window. This window displays the histories that are affected when you click the **Run Maintenance** button. Move the histories that you want manage into this window using the control buttons, as described below:

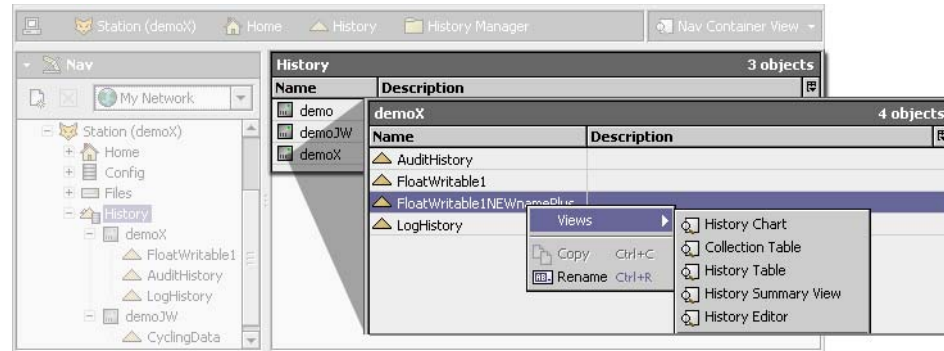
Controls and options for the database maintenance view are described in the following list:

- **Add history button (right arrow)**
Click this button to move histories that are selected in the **available histories** window to the targeted histories window.
- **Remove history button (left arrow)**
Click this button to move histories that are selected in the **targeted histories** window to the available histories window.
- **Clear Old Records option**
Select this option and use the **Before** date selector to remove records, based on date, from the histories that are in the targeted histories window.
- **Before date field**
Use this field with the **Clear old records** option to set the year, month, day, and time parameters that you want to use for removing old records.
- **Clear all records**
Select this option to delete all records from the selected history database.
- **Delete Histories**
Select this option to delete all histories that are in the targeted histories window.
- **Run Maintenance button**
Click this button to execute the option that you have selected on the histories in the targeted histories window.

About the nav container view

The nav container view is shown in Figure 4-18. This view displays a row for each station that is represented in the history space. You can double-click on any of the station icons to display the individual histories that are children to the station that is represented in the Name column of the nav container view. Double click on a history in this view to display the history in the History Chart view.

Figure 4-18 Nav container view

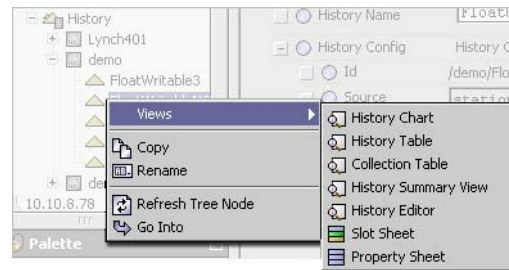


In this view you can select any history and switch to any other view of that history (see “Types of history views” on page 4-10) using the view selector or the popup menu. In this view you can also rename histories, using the popup menu.

Types of history views

You can view histories in different ways in Workbench. Figure 4-19 shows a menu of views that are available using either the Workbench view selector or a view popup menu. These views are listed and described in the following sections.

Figure 4-19 History views available from popup menu



- **History Chart**
This view shows a plotted chart of the history data. This is the default history view. Refer to “About the history chart view” on page 4-11 for more details.
- **Live History Chart**
Starting in NiagaraAX-3.3 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” on page 4-12 for more details.
- **History Table**
This table shows a view of history data that you can export and view in the following formats: PDF, CSV, Text. In addition, you can modify the tabular view of history data by choosing to show or hide columns and by filtering the data based on date and time. Use the **Table Options** menu in the top right corner of the history table to modify the history table view or to export the data in the view, as desired.
- **Collection Table**
This view shows a table of any data (in this case history data) that you can export and view in the following formats: PDF, CSV, Text. Use the **Table Options** menu in the top right corner of the collection table to modify the table view or to export the data in the view, as desired.
- **History Summary**
This view shows a summary of the history’s status and configuration properties.
- **History Editor**
This view allows you to actually edit data and filter histories.
- **History Slot Sheet**
Starting in AX-3.5, this view displays a standard slot sheet for a history. You can add slots, using this view and designate them as metadata slots, if desired.

- **History Property Sheet**

Starting in AX-3.5, this view displays a standard property sheet view for a history. In addition to the default history config properties, any slots that are added using the slot sheet view or the metadata browser display in this view.

Types of history data fields

The following types of data are common to several history table views and appear as columns that may be hidden or displayed using the **Table Options** menu. Refer to “[Table controls and options](#)” on page 2-18 for information about using the **Table Options** menu.

- **Timestamp**

This data field indicates the time that the recorded value occurred.

- **Trend Flags**

This data field displays trend flag information about the recorded data - for trend record types. These flags provide extra context information about the record data. For example: “Start”, “OutOfOrder”, “Hidden”, “Modified”, and “Interpolated” are possible trend flags.

- **Status**

This data field displays the status of the history’s parent component; for example, “OK” or “null”.

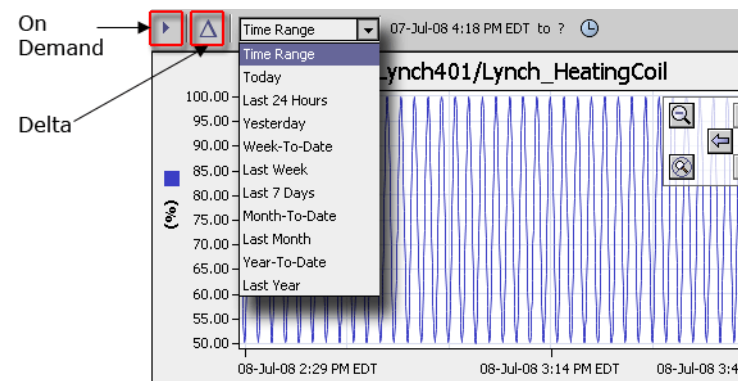
- **Value**

This data field displays the record value.

About the history chart view

The history chart view plots the data of the selected history log along x and y axes. This is the default view of the history, so if you double-click on a history in the nav sidebar, the history chart view will appear. An example of the history chart view is shown in [Figure 4-20](#).

Figure 4-20 History chart view



The history chart view contains the standard chart controls and options to help you customize and view the data, as described in “[Chart controls and options](#)” on page 2-19. In addition to these standard controls, the history chart view has the following additional buttons (also available on the toolbar):

- **On Demand button (available starting in NiagaraAX-3.4)**

This button initiates On Demand plotting of the history data. Also see “About On Demand history views” on page 20.

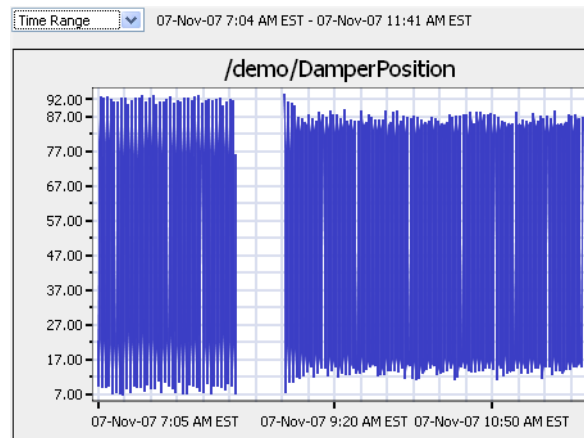
- **Delta button**

This button initiates the plotting of Delta values. Also see, “[About delta logging](#)” on page 4-23.

Charting in the browser

Starting in NiagaraAX-3.3, the Hx charting feature is redesigned to work in the browser without using any plug-ins. The following illustration shows a History Chart in the browser using Hx (no plugin).

Figure 4-21 Displaying a History Chart in the browser with Hx



Hx charts essentially look and work the same in all target media except for a the following differences:

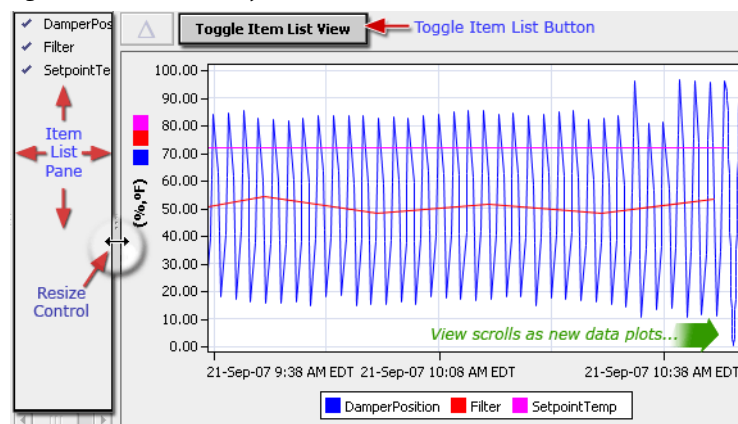
- The default time range for histories is now "Time Range" which is only the bounds of the history data. This is the default in workbench, so now history charts should look the same by default. You cannot however modify this time range. The existing options (Today, Last Month, etc.) are still available.
- Pie charts do not render in Microsoft® Internet Explorer® (IE)
- You may see slower render times in IE since the <canvas> tag has to be emulated using JavaScript and VML.

About the Live History Chart View

Background and live history charting provides a Live History Chart view that is similar to the static History Chart view. The key feature of the Live History Chart view is that it combines the historical plot of the History Chart with a continuing live plot that updates according to a configurable sample rate.

The live history chart view plots the data of the selected history log along x and y axes. This is the default view of the HistoryPointList component, so if you double-click on a HistoryPointList in the NiagaraAX-3.3 (or later) workbench nav sidebar, the Live History Chart view displays. An example of the Live History Chart view is shown in Live_History_Chart_view, below, which highlights the features that are only available in the Live History Chart view.

Figure 4-22 Live History Chart view



The Live History Chart view contains the standard chart controls and options to help you customize and view the data. However, in addition to these controls and options (described in “[Chart controls and options](#)” on page 2-19) this view has the following additional features:

- **Item List pane**
This pane displays all the points that are linked to the chart view. You can click to select or de-select each item to display or remove the associated item plot from the chart. You can always resize the pane by dragging the left border to widen or narrow the pane in the view.
- **Toggle Item List button**
Click this button to display or collapse the Item List pane.

- **Continuously updating view**
As new data is plotted the view continues to update. When the chart area is filled, the screen scrolls to keep showing the latest data.

Selecting the Live History Chart View

You can display the Live History Chart view as either a view of a history extension or as a view of the HistoryChartList component, as shown in the following two illustrations:

Figure 4-23 Selecting the Live History Chart view as a History Extension view

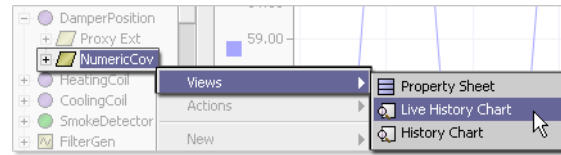


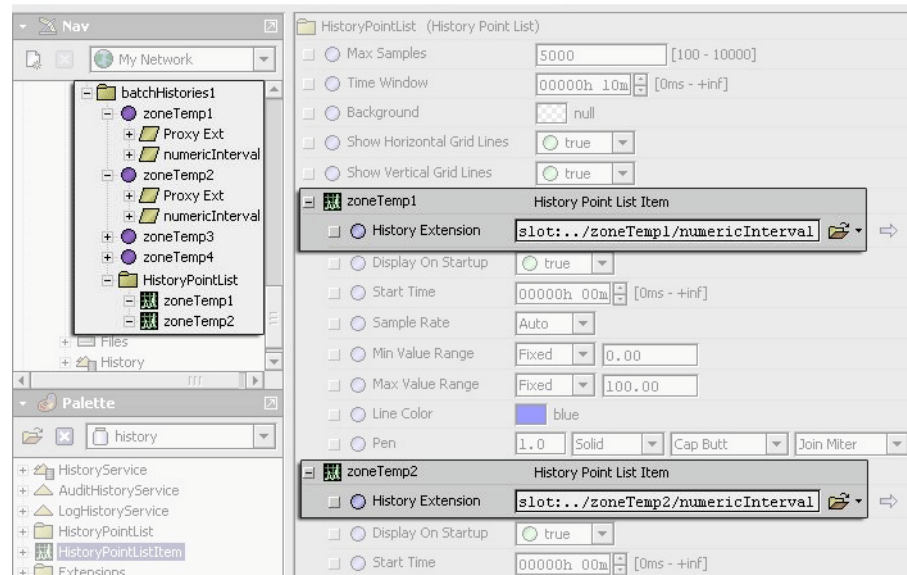
Figure 4-24 Selecting the Live History Chart view as a HistoryPointList view



Using relative history extension ords with HistoryPointList

You can use relative paths to point to history extension ords when using a HistoryPointListItem component. This allows you to create a HistoryPointList under a typical device (Air Handling Unit, VAV, or other) and then copy the device (with HistoryPointList) to other locations without having to edit the ords for each new location. In the example shown in Figure 4-25, the HistoryPointList and the proxy points (zoneTemp1, 2, 3, and so on) with their history extensions are placed at the same level (in the same folder). The HistoryPointList component takes care of resolving the ord path for each HistoryPointListItem, so the relative starting point is the HistoryPointList and not the HistoryPointListItem. In this example, the History Extension path needs to step up one level using the “..” then includes the folder name (“zoneTemp1/numericInterval”) to designate the history extension.

Figure 4-25 Relative ords to history extension in HistoryPointListItem



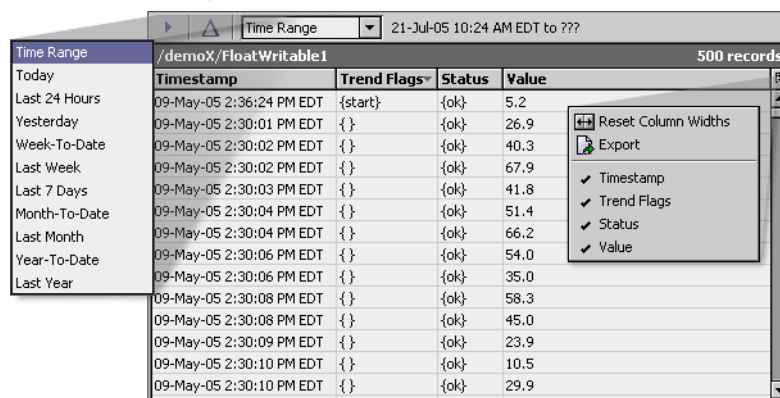
Note: You can also embed the History Chart view of a “relativized” history extension ord into a Px view. This allows you to use a single Px view for multiple HistoryPointList (and associated Live History Chart view) locations.

About the history table view

The history table view displays history records with columns of data that you can customize by displaying or hiding selected columns. An example of the history table view is shown in Figure 4-26.

In addition to these standard controls, the history table view has the following additional buttons (also available on the toolbar):

- **On Demand button**
This button initiates On Demand plotting of the history data. Also see “About On Demand history views” on page 20.
 - **Delta button**
This button initiates the plotting of Delta values. Also see “About delta logging” on page 23.
- Refer to “[Table controls and options](#)” on page 2-18 for details about table controls and options that are common to many table views.

Figure 4-26 History table view

Time Range	Timestamp	Trend Flags	Status	Value
Today	09-May-05 2:36:24 PM EDT	{start}	{ok}	5.2
Last 24 Hours	09-May-05 2:30:01 PM EDT	{}	{ok}	26.9
Yesterday	09-May-05 2:30:02 PM EDT	{}	{ok}	40.3
Week-To-Date	09-May-05 2:30:02 PM EDT	{}	{ok}	67.9
Last Week	09-May-05 2:30:03 PM EDT	{}	{ok}	41.8
Last 7 Days	09-May-05 2:30:04 PM EDT	{}	{ok}	51.4
Month-To-Date	09-May-05 2:30:04 PM EDT	{}	{ok}	66.2
Last Month	09-May-05 2:30:06 PM EDT	{}	{ok}	54.0
Year-To-Date	09-May-05 2:30:06 PM EDT	{}	{ok}	35.0
Last Year	09-May-05 2:30:08 PM EDT	{}	{ok}	58.3
	09-May-05 2:30:08 PM EDT	{}	{ok}	45.0
	09-May-05 2:30:09 PM EDT	{}	{ok}	23.9
	09-May-05 2:30:10 PM EDT	{}	{ok}	10.5
	09-May-05 2:30:10 PM EDT	{}	{ok}	29.9

In addition to a title bar that displays the history name and number of records in the table, the history table has the following four columns that are described in “[Types of history data fields](#)” on page 4-11.

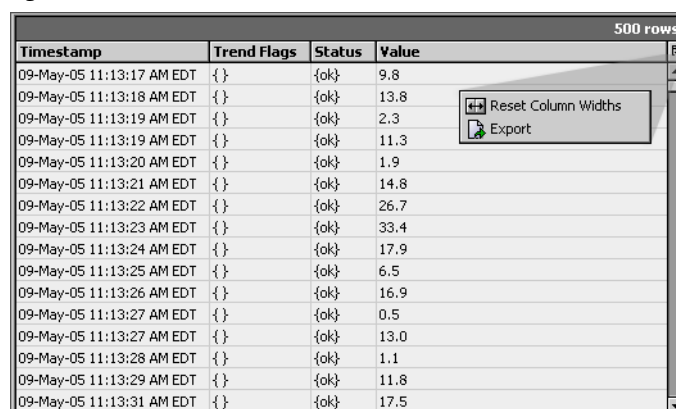
- **Timestamp**
- **Trend Flags**
- **Status**
- **Value**

Use the **Table Options** menu in the top right corner of the history table to modify the table view or to export the data in the view, as desired. Refer to “[Table controls and options](#)” on page 2-18 for a description of the **Table Options** menu.

About the collection table view

The collection table view displays records with columns of data that you can customize by displaying or hiding selected columns. Refer to “[Table controls and options](#)” on page 2-18 for details about table controls and options that are common to many table views.

An example of the collection table view is shown in [Figure 4-27](#).

Figure 4-27 Collection table view

Timestamp	Trend Flags	Status	Value
09-May-05 11:13:17 AM EDT	{}	{ok}	9.8
09-May-05 11:13:18 AM EDT	{}	{ok}	13.8
09-May-05 11:13:19 AM EDT	{}	{ok}	2.3
09-May-05 11:13:19 AM EDT	{}	{ok}	11.3
09-May-05 11:13:20 AM EDT	{}	{ok}	1.9
09-May-05 11:13:21 AM EDT	{}	{ok}	14.8
09-May-05 11:13:22 AM EDT	{}	{ok}	26.7
09-May-05 11:13:23 AM EDT	{}	{ok}	33.4
09-May-05 11:13:24 AM EDT	{}	{ok}	17.9
09-May-05 11:13:25 AM EDT	{}	{ok}	6.5
09-May-05 11:13:26 AM EDT	{}	{ok}	16.9
09-May-05 11:13:27 AM EDT	{}	{ok}	0.5
09-May-05 11:13:27 AM EDT	{}	{ok}	13.0
09-May-05 11:13:28 AM EDT	{}	{ok}	1.1
09-May-05 11:13:29 AM EDT	{}	{ok}	11.8
09-May-05 11:13:31 AM EDT	{}	{ok}	17.5

In addition to a title bar that displays the history name and number of records in the table, the history table has the following four columns that are described in “[Types of history data fields](#)” on page 4-11.

- **Timestamp**
- **Trend Flags**
- **Status**
- **Value**

Use the **Table Options** menu in the top right corner of the history table to modify the table view or to export the data in the view, as desired. Refer to “[Table controls and options](#)” on page 2-18 for a description of the **Table Options** menu.

About the history summary view

The history summary view is shown in [Figure 4-28](#). This view displays the details of the history’s status and configuration properties in two groups, as follows:

Figure 4-28 History summary view

The screenshot shows a window titled 'History Summary View' with two main sections: 'Status' and 'Configuration'.

Status Section:

- Record Count: 20
- First Timestamp: 02-May-2005 11:03 AM EDT
- Last Timestamp: 02-May-2005 03:10 PM EDT

Configuration Section:

- Id: /demo/FloatWritable1
- Source: station:/slot:/Sampler/FloatWritable1/NumericInter
- Time Zone: America/New_York (-5/-4)
- Record Type: history (selected), NumericTrendRecord
- Capacity: Record Count: 500 records
- Full Policy: Roll
- Interval: regular, +00000h 13m 00s
- valueFacets: units=null,precision=1,max=+inf,min=-inf >>

- **Status parameters**
these parameters display data that is updated as of the time you select the history summary view.
 - Record count
this is the current number of total records, as of the Last Timestamp.
 - First Timestamp
this is the date, time and timezone information for the initial history record.
 - Last Timestamp
this is the date, time and timezone information for the latest recorded history record.
- **Configuration parameters**
these parameters display data that identifies and characterizes the specific history. Refer to “[Configure history extensions](#)” on page 4-25 (see the *History Config* bullet item) for a description of the configuration parameters.

About the history editor view

An example of the history editor view is shown in [Figure 4-29](#). The history editor view allows you to edit data and filter histories. This view also allows batch editing (selecting multiple rows and using the popup menu or the Edit menu).

Figure 4-29 History editor view

The screenshot shows a window titled 'History Editor' with a table of history records. The table has columns: Timestamp, Trend Flags, Status, and Value. A 'Table Options Menu' is visible over the table.

Timestamp	Trend Flags	Status	Value
02-May-05 11:03 AM EDT	{start}	{null}	0.0
02-May-05 11:16 AM EDT	{}	{null}	0.0
02-May-05 11:29 AM EDT	{}	{null}	0.0
02-May-05 11:42 AM EDT	{}	{null}	0.0
02-May-05 11:55 AM EDT	{}	{null}	0.0
02-May-05 12:08 PM EDT	{}	{null}	0.0
02-May-05 12:21 PM EDT	{}	{null}	0.0
02-May-05 12:34 PM EDT	{}	{null}	0.0
02-May-05 12:47 PM EDT	{}	{null}	0.0
02-May-05 1:00 PM EDT	{}	{null}	0.0

The history editor view is comprised of the following main areas:

- **Title bar**
this area displays the history name and number of records in the history.

- **Toolbar icons**
these icons are available on the toolbar when the history editor view is displayed. Refer to [“About the history editor toolbar icons”](#) on page A-15 for descriptions of these toolbar icons.
- **Time range options**
this menu is located in the top left corner of the history editor view. You can select one of the pre-defined times or select the **Time Range** option that allows you to set a specific time range using the **Edit Time Range** dialog box.
- **Table options menu**
Use the **Table Options** menu in the top right corner of the history editor view to change which columns are displayed or to export the data in the view, as desired. Refer to [“Table controls and options”](#) on page 2-18 for more information about using the **Table Options** menu.
- **Table columns**
In addition to a title bar that displays the history name and number of records in the table, the history table has the following four columns that are described in [“Types of history data fields”](#) on page 4-11.
 - **Timestamp**
 - **Trend Flags**
 - **Status**
 - **Value**

Use the **Table Options** menu in the top right corner of the history table to modify the table view or to export the data in the view, as desired. Refer to [“Table controls and options”](#) on page 2-18 for a description of the **Table Options** menu.
- **Control buttons**
The following two control buttons are used to initiate record editing in the history editor view:
 - **Edit**
this button is available when one or more records are selected in the history editor table. When you click this button, the **Edit Records** dialog box appears.
 - **Select Outliers**
this button is available when the outlier configuration parameters are active (when the option box is selected in the **Configure Outliers** dialog box). When you click this button, the **Configure Outliers** dialog box appears.

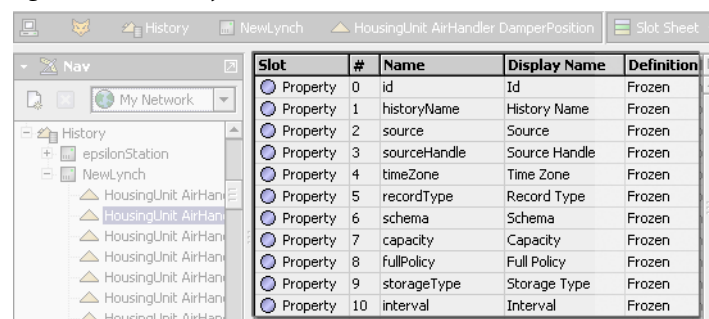
Use the **Table Options** menu in the top right corner of the history table to modify the table view or to export the data in the view, as desired. Refer to [“Table controls and options”](#) on page 2-18 for information about using the **Table Options** menu.

About the history slot sheet view

Starting with AX-3.5, a slot sheet view is available on histories. This view allows you to edit and add slots to histories, if desired. In addition, you can designate slots as “metadata slots” by assigning a metadata flag to a slot.

Note: *Properties that have the metadata flag are designated as metadata properties. You might use metadata property values to identify the location and function of a history's source. History sources may be filtered or organized by the values of the metadata properties. You can use the Metadata Browser view to add, edit, delete, or view metadata tags.*

Figure 4-30 History slot sheet view



Slot	#	Name	Display Name	Definition
Property 0	0	id	Id	Frozen
Property 1	1	historyName	History Name	Frozen
Property 2	2	source	Source	Frozen
Property 3	3	sourceHandle	Source Handle	Frozen
Property 4	4	timeZone	Time Zone	Frozen
Property 5	5	recordType	Record Type	Frozen
Property 6	6	schema	Schema	Frozen
Property 7	7	capacity	Capacity	Frozen
Property 8	8	fullPolicy	Full Policy	Frozen
Property 9	9	storageType	Storage Type	Frozen
Property 10	10	interval	Interval	Frozen

To add a new property to a history using the slot sheet view, open the **Add Slot** dialog box (Ctrl+A) from the history Slot Sheet view, as shown below.

Figure 4-31 Select a Metadata flag to create a metadata slot

See also: [“About the history property sheet view”](#) on page 4-17 and [“About the metadata browser view”](#) on page 4-18.

About the history property sheet view

Starting in AX-3.5, histories can have properties. Like components, histories have a property sheet view that allows you to view and edit property values. You can add properties to a history (and designate them as metadata, if desired) using the slot sheet view or the metadata browser view. An example of the history property sheet view is shown in [Figure 4-32](#).

Figure 4-32 History property sheet view

The history property sheet view is comprised of the default history properties, as described in [“Configure history extensions”](#) on page 4-25 and any additional dynamic properties that are added. If you change any of the editable properties in this view then the properties are changed at the actual source of the history, if the source is accessible. For example, in a history property sheet view, if you change a history Full Policy property from Stop to Roll, then, if the “source” is accessible, the history Full Policy property is changed on the appropriate history extension or History Import component (Config Overrides property).

Note: *If the history is imported from a NiagaraHistoryImport descriptor or NiagaraSystemHistoryImport descriptor that was made via *export tags* (HistoryImportTag, SystemHistoryImportTag) in the source station, the next export tag Join to/from that station will *overwrite* whatever fullPolicy or other property value changes that were initiated from the history property sheet.*

Consider the following examples:

Example 1: History imported from remote controller to Supervisor


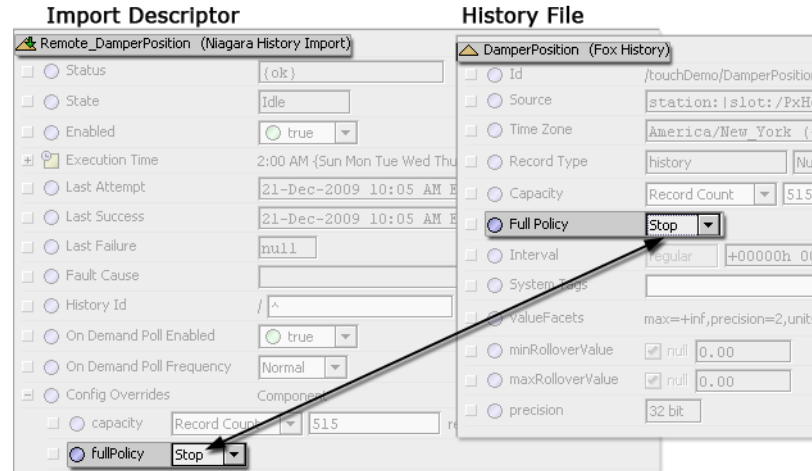
In a situation where histories are imported to a Supervisor from a remote controller, the source configuration for that history is available on the supervisor Import Descriptor  property sheet view. If you change the fullPolicy on the history property sheet view (as shown below), then the same property is changed on that history's Niagara History Import descriptor component (under the Config Overrides property).

Figure 4-33 Example 1: History property changes Import Descriptor property



Local histories behave in the same way, since local histories and history import descriptors should always be available to the supervisor station.

Example 2: History exported to Supervisor by remote controller

Histories that are exported from remote stations may create histories on the Supervisor that have an unreachable source. In this case:

- If the source of a history is unreachable, then the history configuration properties that are modified (under the history properties view on the supervisor) may be overwritten the next time that the history is exported to the supervisor.
- History properties that are flagged as metadata properties are not overwritten or dropped when histories are exported to the supervisor. Local metadata property values are persisted as long as they do not have the same name as properties that are being exported from the remote station.

Note: In most cases, it is a better practice to add metadata properties at the remote station if you are doing history exports so that they are exported up to the supervisor station.

See also: [“About the history slot sheet view”](#) on page 4-16 and [“About the metadata browser view”](#) on page 4-18.

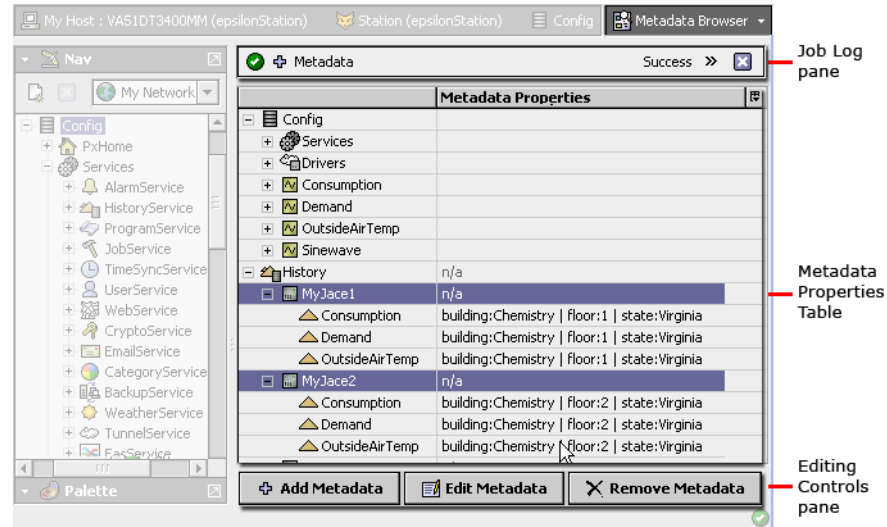
About the metadata browser view

Starting in AX-3.5, the metadata browser view is available for working exclusively with metadata properties on components and histories. The interface provides a convenient way to add metadata slots to one or more components or histories. Although you can add metadata slots directly to a component or a history from the slot view, the metadata browser has the ability to add metadata slots to thousands of objects in one batch job.

The interface (shown in the following image) is a view on the root station component. It allows you to filter and navigate through your histories or components so that you can selectively batch-add metadata to all the desired objects. The view runs a job for the metadata operation and reports the results in a job log. If a job runs and is unsuccessful at any point, the job log displays the reason for the failure.

Note: An example of a failure might be that you try to add a slot type to an object that does not support that particular slot type. This exception should be indicated in the job log.

Figure 4-34 Metadata browser view



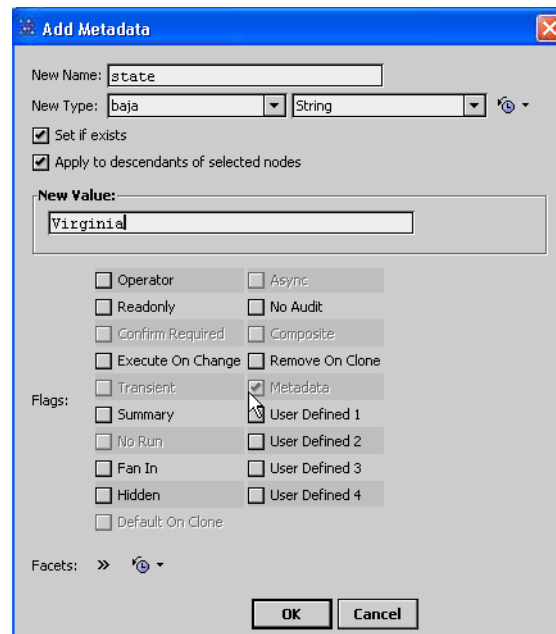
The following areas comprise the main parts of the metadata browser view:

- **Job Log pane**
This pane displays after you run any of the editing control jobs to add, edit, or remove metadata. Click the >> icon to view the job log or click the x icon to remove the most recent job log record.
- **Metadata Properties table**
This table has two adjustable-width columns as follows:
 - **Navigation**
This column provides a navigation tree with root nodes at the Config (station) level and at the History level. The navigation column displays an expandable tree structure representation of the selected station. Expanding a parent node displays additional rows that you can select for editing metadata.
 - **Metadata Properties**
This column displays a summary string of the properties. The metadata property name and its corresponding value are separated by a colon (:) and individual metadata properties are separated by a bar (|). For example: country:USA | state:Alaska | city:Fairbanks
Note: The history “device” object always displays an “n/a” in this column because it does not support the use of properties.
- **Editing controls pane**
This pane contains the following buttons that open dialog boxes that you can use to add, edit, and remove metadata, as desired:
 - **Add Metadata button**
Click this button to open the **Add Metadata** dialog box. Fields available in this dialog box vary according to the type of property that you select. The following fields are available for the default “string” type property:
 - **New Name:**
Enter the desired name of the property.
 - **New Type:**
Select the property type that you want to add. The default is string type.
 - **Set if exists**
This option box allows you to choose to edit an *existing* property value, if it already is present in or under the selected object(s). If this option is not selected, a new property may be added but an existing property is not edited when the Add Metadata job runs.
 - **Apply to descendants of selected nodes**
This option, when selected, allows you to run the Add Metadata job on the selected object and all of its descendant objects. If the option is not selected, only the selected object is edited.
Note: You can select multiple rows in the Metadata Browser table by using the Shift or Ctrl keys to select a range or a non-contiguous set of rows, respectively.
 - **New Value:**

This field allows you to set an appropriate value for the property type. For example, a string field displays for the default string type and allows you to enter the string value that is used for the property.

- **Flags:**
In addition to the default (required) Metadata flag option, other flags are available.
- **Facets**
Use this field to add facets to the slot, if needed, for the chosen property type and use.

Figure 4-35 Metadata dialog box

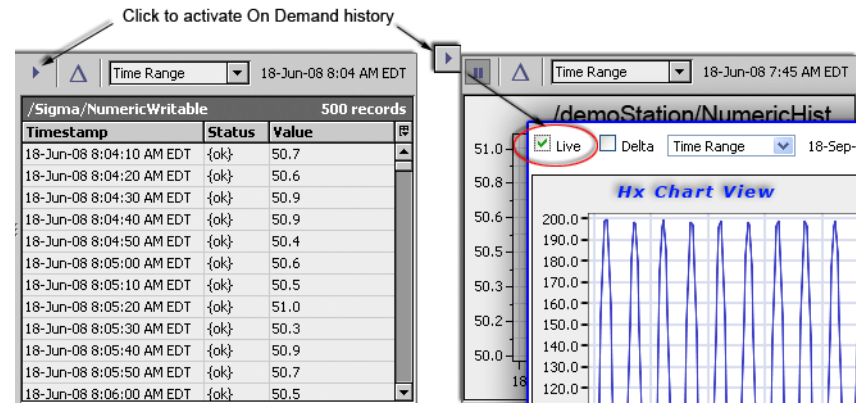


- **Edit Metadata** button
This button is available only when you select an object that already has one or more metadata properties. Clicking the **Edit Metadata** button opens the **Edit Metadata** dialog box with the following property options:
 - **Property option list**
This is an option list that displays all metadata properties assigned to the object and allows you to select the property that you want to edit.
 - **Apply to descendants of selected nodes**
Selecting this option applies any edits you make to this property to all descendant objects under the selected object.
 - **New Value:**
This field allows you to edit the selected metadata property by specifying a new value for it.
- **Remove Metadata** button
This button is available only when you select an object that already has one or more metadata properties. Clicking the **Remove Metadata** button opens the **Edit Metadata** dialog box with the following property options:
 - **Property option list**
This is an option list that displays all metadata properties assigned to the object and allows you to select the property that you want to remove.
 - **Apply to descendants of selected nodes**
Selecting this option removes this property from all descendant objects under the selected object.

About On Demand history views

Starting in NiagaraAX-3.4, the On Demand history feature enables polling of both local history sources and Niagara History imports for live data when displaying history chart or history table views. This feature works in addition to, and does not replace, a standard polling schedule. For example, you would typically still have your History Import descriptors scheduled to archive at some daily interval (such as every night), even though you might be displaying On Demand history views.

Figure 4-36 On Demand history example view



The On Demand feature provides a 'Live Updates' toggle button (checkbox in Hx view) on both the history chart and history table views. When you click the 'Live Updates' button, it initiates a history subscription that finds the source of the history and subscribes to that source component. If the source is a history import descriptor, then that descriptor starts polling at a frequency defined by its On Demand Poll Frequency property (described below). If the source is a local History Ext, the history chart or table updates whenever the history extension appends a new record to the history.

Note: When the On Demand History is active, a timestamp appears in the view's status bar (lower corner) to indicate the time of the last successful update. This does not apply to Hx views.

Since histories eligible for on demand support are sourced directly from a local history extension or at a history import descriptor, On Demand History *availability* and *update rate* are both affected by parameters described below and shown in Figure 4-38:

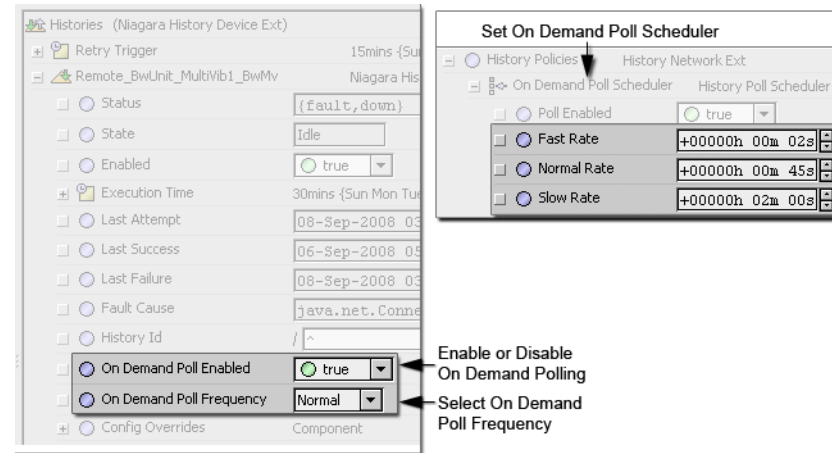
- **On Demand History availability is affected by the following factors:**
 - On Demand History is *not* available if the source component is not accessible. Since histories can be categorized (for permissions levels) independently from their source components, it may be possible for a user to have access to a history, but not have access to the source component. In this case, the On Demand History feature is disabled (button is dimmed on the History Chart or Table view). Also, if the source component is no longer in existence (has been deleted) the "Live Updates" toggle button is dimmed (inaccessible).
 - On Demand History is *not* available if the source history import descriptor's On Demand Poll Enabled property is set to `false` (see Figure 4-37). On Demand Polling may be disabled to limit bandwidth usage.

Figure 4-37 On Demand Poll setting in the History Import Manager view



- If the On Demand Poll Scheduler property is disabled (or the driver network does not have it included in a HistoryNetworkExt)
- **On Demand History rates are affected by the following property settings (see Figure 4-38):**
 - **On Demand Poll Scheduler (sources using import descriptor)**
This property is located under the NiagaraNetwork History Policies property. You can set standard polling rate values for a Fast Rate, Normal Rate, and a Slow Rate in the fields displayed in the property sheet view. These rates are available as selection options for individual Niagara History Import descriptors under the NiagaraNetwork History Device Extensions.
 - **On Demand Poll Frequency (sources using import descriptor)**
This property is located under the Niagara History Import property. Select one of the three options (defined in the On Demand Poll Scheduler): Fast, Normal, or Slow.

Figure 4-38 On Demand Poll frequency settings for import descriptor-sourced views



- History extension update interval**
 If the On Demand history is being sourced from a local history extension, then the On Demand chart or table view updates whenever the local history extension appends a new record to the history. This function reflects the presence of a (Last Record) property starting in NiagaraAX-3.4.

About the history process

There are essentially three steps in the “history logging” life cycle, as shown in [Figure 4-39](#):

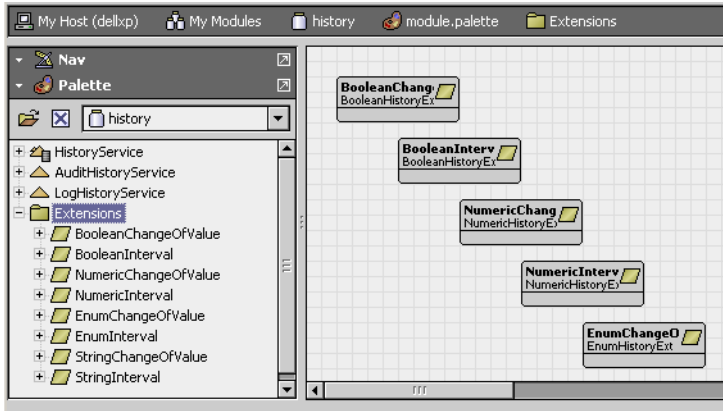
Figure 4-39 Simplified history process



- collect**
 involves defining the parameters that specify what data will be recorded and when (or how often) it will be recorded. For example, you can collect data whenever a change of value occurs - or at a regular time interval that you specify.
 To collect history information you need to:
 - Add history extensions to a component (refer to [“Add history extensions to a component”](#))
 - Configure the extensions (refer to [“Configure history extensions”](#) on page 4-25)
 - Use a valid history name (part of configuration, see, [“About history names”](#) on page 4-26)
 Refer to [“Add history extensions to a component”](#) on page 4-24 for more details.
- store**
 involves defining the parameters of the history database file. For example, you can customize the name of the database file, define the maximum number of records to save, and choose metadata to add to the records. Refer to [“Configure history extensions”](#) on page 4-25 for more details.
- archive (transfer)**
 includes importing and exporting records from one station to another station. For example, you can limit your local station records to a small number that you specify and archive all records to another station. Refer to [“Archiving”](#) on page 4-27 for more details.

You can add extensions to a component’s property sheet in order to *extend* the functionality of the component. By adding a history extension, the real-time value or status of the component’s output can be collected as a time-stamped entry in the associated history table. History extensions are available in the history palette, as shown in [Figure 4-40](#). The history table is not stored as part of the component’s data but is a separate collection of data simply referred to as the history.

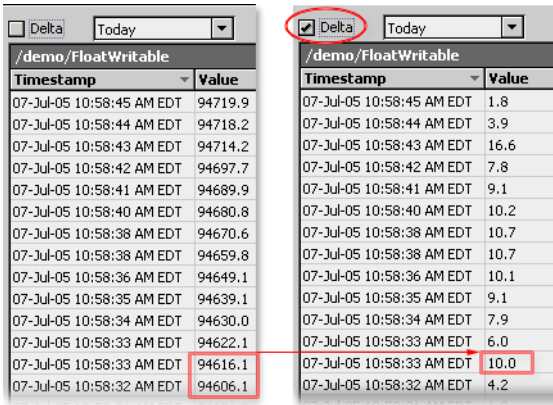
Figure 4-40 History extensions



About delta logging

When you are logging data, such as electric consumption (kWh) or other information that uses a running total, you may want to know the difference between consecutive timestamped values instead of the actual running total. The delta logging feature is provided for this type of calculation. For delta logging, data is logged (as normal) using the appropriate NumericChangeOfValue or NumericInterval extension. Then, in the history chart or history table view, you simply select the `Delta` option box (or use the toolbar icon) to display the delta values instead of the running total value, as shown in Figure 4-42. Delta values are computed by taking the difference between one numeric record and the next. The timestamp of the last record (of the two) is used as the timestamp for the delta value.

Figure 4-41 Delta logging history values



Two other parameters that apply to delta logging are related to the concept of **rollover**.

Rollover occurs when a running total reaches a defined maximum number and then resets to “zero” or another defined number. The defined maximum number is represented in the history extensions by the **Max Rollover Value** parameter. The reset value (which is often zero) is represented in the history extensions by the **Min Rollover Value** parameter. These parameters allow you to specify the behavior of the delta logging when the rollover occurs. If you do not know these values or if they are not specified, then select the `null` option for these parameters.

Figure 4-42 Using rollover value parameters with delta logging

Consider the following example. If you are logging energy consumption with the Max Rollover Value parameter set to 999,999 and the Min Rollover Value set to 100, then when a rollover is detected, the delta logging bases its delta calculations on a maximum value of 999,999 and a subsequent initial value of 100.

Refer to [“Configure history extensions”](#) on page 4-25 for a list and short description of all the history extension parameters.

Add history extensions to a component

Extensions come in different types to match the data type of the component and the collection method desired. You can add history extensions by dragging them onto your property sheet or into your nav side bar pane from the history palette (see [Figure 4-44](#)). Refer to [“To add an extension to a component property sheet view:”](#) on page 9-17 for steps for adding an extension.

Figure 4-43 History extensions in Workbench the history palette

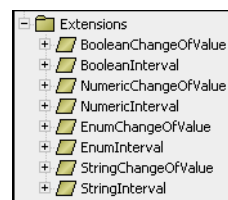


Figure 4-44 History extensions in the property sheet

Extension types include:

- **BooleanChangeOfValue**
used to collect boolean values whenever those values change.
- **BooleanInterval**
used to collect boolean values at specified time intervals.
- **NumericChangeOfValue**
used to collect numeric values whenever those values change.

- **NumericInterval**
used to collect numeric values at specified time intervals.
- **EnumChangeOfValue**
used to collect integer values (enumerations) whenever those values change.
- **EnumInterval**
used to collect integer values (enumerations) at specified time intervals.
- **StringChangeOfValue**
used to collect a string of characters whenever the characters change.
- **StringInterval**
used to collect a string of characters at specified time intervals.

For more details about extension types, refer to “[Types of history extensions](#)” on page 3-16.

Configure history extensions

When a history extension is first added to a component, it is set to be disabled by default. All that is really necessary to enable collection is to change the 'Enable' property to 'true.' Of course, you may desire some other property changes to better configure the collection to suit your actual needs.

Some configuration properties are common among all history extension types. A few properties are unique to the data type or the collection type. Following, is a list of all the properties available.

- **Status**
read-only field that tells the user the current status.
- **Fault Cause**
read-only field that displays a reason that the extension is not working.
- **Enabled**
set either 'true' or 'false' to disable or enable the history extension.
- **Active Period**
specifies the days of the week as well as start and end times for record collection.
- **Active**
read - only value that indicates whether or not (true or false) the history collection is active (as defined by the Active Period parameters).
- **History Name**
is a formatting convention used to provide a way to consistently name histories using a standardized formatting pattern.
The history name format field provides a way to automatically name histories any time a new history is created. The format `%parent.name%` is the default history name format and this will create a name based on the parent object. Refer to “[About history names](#)” on page 4-26 for more details about naming histories.
- **History Config**
These properties set up the attributes of the History record.
 - **Id**
a unique identification for each history. The id value is the name that was automatically created by the History Name Format or manually created in the name field or rename dialog box.

Figure 4-45 History Id



- **Source**
a read-only field that displays the ORD of the active history extension.
- **Time Zone**
a read-only field that displays the time zone of the active history extension.
- **Record Type**
displays the data that the record holds in terms of: extension type (history) and data type: (BooleanTrendRecord, NumericTrendRecord, and so on).
- **Capacity**
allows you to set a finite number of records to collect *or* to choose to collect an unlimited number of records. If you choose the **Record Count** option, an additional “records” field displays. In the “records” field, type in the maximum number of records that you want to save in the history database.
- **Full Policy - (Roll or Stop)**

allows you to choose what to do when the Capacity number is reached. The `Roll` option drops off the oldest record to make room for the newest record. The `Stop` option simply causes the history to stop recording.

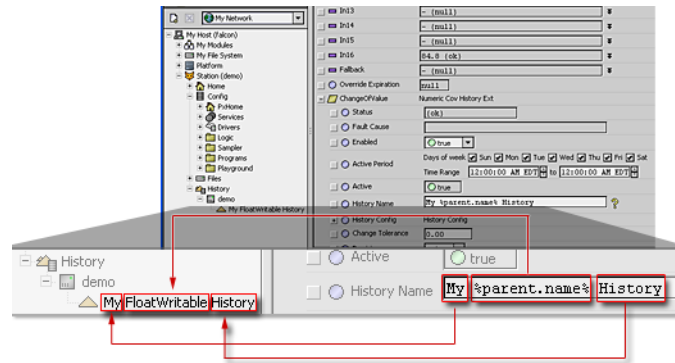
- **Interval**
a read-only field that displays the selected collection type: `regular` or `interval`.
- **System Tags**
Available starting in NiagaraAX-3.4, the System Tags property allows you to add an additional identifier (the System Tag) to a history extension. This identifier is then available for selective import or export of histories using the Niagara System History Import or Niagara System History Export option (using the System Tag Patterns). Each System Tag is separated by a semicolon. For example: `NorthAmerica;Region1;Cities`. Also see [“About System Tags”](#) on page 4-29.
- **valueFacets**
allows you to use the Edit Facets dialog box to choose how you want to display the logged data.
- **Change Tolerance**
for COV based collections, the monitored variable has to change at least this much to be logged as a changed value.
- **Interval**
for Interval-based collection, the cycle time, or how often the history parameters are checked. Any time you change this parameter, a new history is created (or “split-off”) from the original history because histories with different intervals are not compatible.
- **minRolloverValue**
this is a number that is used as the starting point for calculations for cumulative logging after a rollover. Rollover occurs after a running total maximum value is reached. Select the `null` option if a `minRolloverValue` is unknown. Refer to [“About delta logging”](#) on page 4-23 for more details about rollover and delta logging.
- **maxRolloverValue**
this is a number that is used as a maximum value for calculations when a rollover is detected by the history logging process. Using this parameter and the `minRolloverValue` parameter helps you avoid getting negative numbers when you are logging running total data, such as energy usage. Refer to [“About delta logging”](#) on page 4-23 for more details about rollover and delta logging.
- **precision**
this option allows you to select 32 bit or 64 bit options for the history data logging. 64 bit allows for higher level of precision but consumes more memory.

About history names

By default, when a history extension is added to a component, a history format default string is set to the following: `%parent.name%`. This string automatically names any histories with the name of the parent component and appends a sequential number to additional names, as necessary. For example, a history extension on a `NumericWritable` component will, by default, create a history name “`NumericWritable`” - then if that component is duplicated, the name for the history will be duplicated and incremented to “`NumericWritable1`”. You may add a literal name in front of, or after, the `%parent.name%` string, as shown in [Figure 4-46](#), to customize it without losing the automatic naming feature.

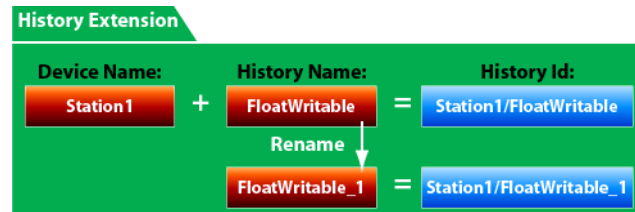
Note: *If you use a literal name and not a script to name a history, you lose the automatic incrementing feature that the script provides. Duplicating extensions that have a literal name duplicates the exact literal name and therefore creates an invalid (non-unique) name. In this case, you must rename each duplicated history manually to create a unique name and avoid a fault condition.*

Figure 4-46 History naming relationships



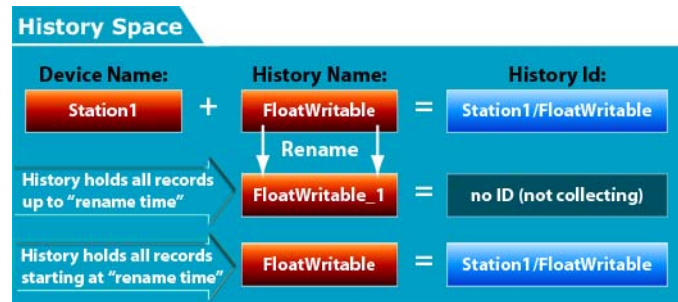
As described in “Configure history extensions” on page 4-25, and illustrated in Figure 4-45, history names are part of the unique history extension property “Id”. When you *rename* a history at the history extension, you are renaming the history at its “source”. Therefore, the history configuration and the history Id *both* change. This concept is illustrated in Figure 4-47.

Figure 4-47 Renaming a history in the history extension



If, however, you rename a history in a “history space” view, such as under the history space node in the nav sidebar, or in the nav container view, you are changing the name of the history as it has been saved in the history space—not at the configuration (or origination) level. Therefore, the history Id is not changed and the history extension continues to produce records under the original history name as long as that history extension is active and enabled. This results in a history “split”; the newly-named history is no longer updated, as of the time of the renaming, but it contains all the records up to that time. In this scenario, a history under the original name begins with the first record after the renaming and continues recording as configured. This concept is illustrated in Figure 4-48.

Figure 4-48 Renaming a history in the history space



Renaming Summary:

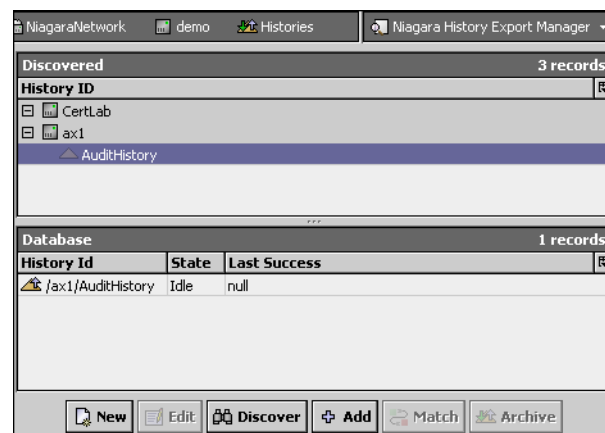
- **No history split**
If you rename a history in either the property sheet view or the history extension manager view, you are editing the actual history extension and therefore not forcing a history split.
- **History split**
If you rename a history in either the nav side bar view or the nav container view you are editing the name in the history space and not actually changing the history ID – the history is split.

Archiving

Archiving is the process of saving a history out to a different location (station) from where it was originated. There are two general methods (or directions) for archiving, as described below:

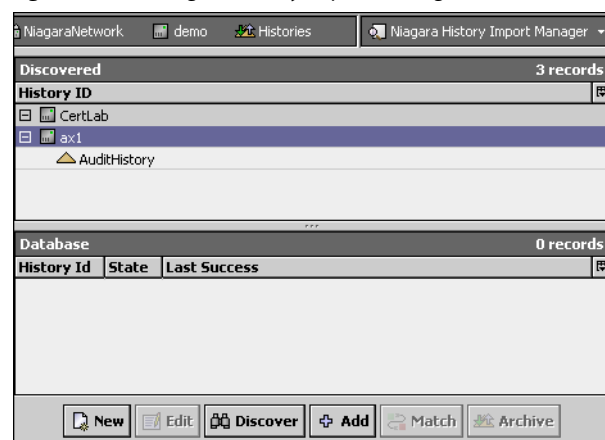
- **Pushing data (exporting histories)**
is the process of exporting history data and is done using the history export manager, shown in [Figure 4-49](#). For details about exporting histories, refer to the “Niagara History Export Manager” section in the *Drivers Guide*.

Figure 4-49 Niagara history export manager



- **Pulling data (importing histories)**
is the process of importing history data and is done using the history import manager, shown in [Figure 4-50](#). For details about importing histories, refer to the “History Import Manager” section in the *Drivers Guide*.

Figure 4-50 Niagara history import manager



About editing history data

The history editor view, as described in [“About the history editor view”](#) on page 4-15, provides the ability to review and modify the history data that you have collected.

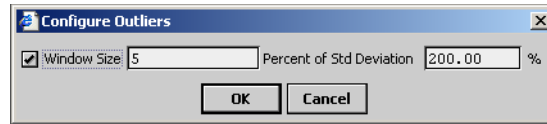
In some cases, you may want to search through the history data and look for unusual data or “outliers.” An “outlier” is a data value that is far apart from the rest of the data; an extreme value that is either much lower or much higher than the rest of the values in the data set. Outliers are known to skew means or averages, so it may be helpful to identify and edit or hide this data. This doesn’t mean that the data point is necessarily bad – but in most cases the information is more helpful without the inclusion of this “unusual” data.



Caution It is possible to alter good data and miss filtering some bad data points using the history editor view.

Workbench provides the ability to find and edit outliers based on parameters that you specify. The **Configure Outliers** dialog box, shown in [Figure 4-51](#), appears when you click the **Configure Outliers** icon in the toolbar menu. Refer to [“About the history editor toolbar icons”](#) on page A-15 for a list of toolbar icons specific to the history editor view.

Figure 4-51 Configure Outliers dialog box



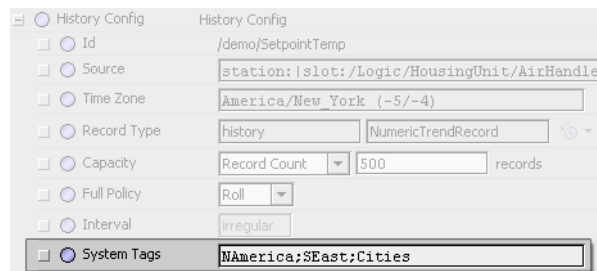
- **Configure Outliers check box**
Outlier filtering is disabled by default. Select the check box to enable the outlier filtering feature and use the parameters that are displayed in the dialog box. Clear the check box to disable outlier data filtering. When outlier parameters are enabled, the **Window** and the **Percent of Std Deviation** fields are available and allow you to specify the “intensity” of the search for outliers in the data.
- **Window**
Enter an integer in the Window field to define the number of surrounding data points to consider when determining whether a given point is an outlier. For example, if you use the default value of “4”, it will look at the two points before and after the point under investigation (PUI). This is a surrounding “window” of 4 points—from which a standard deviation will be calculated and used with the percentage parameter, as described, below.
- **Percentage**
Enter a value in this field to specify the percent of standard deviation (calculated from the window of points) to apply for identifying whether or not the PUI should be considered a valid value (not an outlier). If the PUI falls outside of this valid range, then it is considered to be an outlier and its value is replaced by the linear interpolation of the surrounding 2 valid points. If the PUI falls within the range, then the data point is used and considered valid.

About System Tags

Available starting in NiagaraAX-3.4, the System Tags property allows you to specify an additional identifier (the System Tag) to use with a history extension.

The System Tag identifier is specified as one or more string parameters under the History Config property of a History Extension (see [Figure 4-52](#)). Each string in a System Tag property must be separated by a semicolon. For example: NorthAmerica;Region1;Cities

Figure 4-52 Specifying system tags



This additional identifier is then available for selective import or export of histories using the Niagara System History Import or Niagara System History Export option from the Niagara History Import Manager or Niagara History Export Manager, respectively (see [Figure 4-53](#) and [Figure 4-54](#)).

Figure 4-53 Using a Niagara System History Export

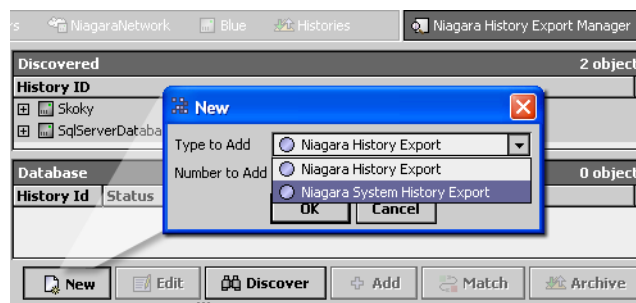


Figure 4-54 Specifying a System Tag pattern for history import

Name	History Id	Execution Time	Enabled
NiagaraSystemHistoryImport		2:00 AM {Sun Mon Tue Wed Thu Fri Sat }	true

☐ Name

NiagaraSystemHistoryImport

☐ History Id

Cannot edit

☐ Execution Time

Daily

Time Of Day 02:00:00 AM EDT

Randomization +000000h 00m 00s

Days Of Week ☒ Sun ☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat

☐ Enabled

true

☐ Capacity

Record Count

500

records

☐ Full Policy

Roll

☐ On Demand Poll Enabled

true

☐ On Demand Poll Frequency

Normal

☒ System Tag Patterns

NAmerica;SEast

For more information about History Import and History Export, refer to the NiagaraAX *Driver Guide* sections: “Using System Tags to import Niagara histories” and “Niagara History Export Manager”.

About history grouping

Available starting in AX-3.5, history grouping allows you to setup alternate navigation of your history space based on history properties.

Figure 4-55 History Groupings component comes as part of the HistoryService in AX-3.5

Nav

My Network

Services

AlarmService

SmsService

OnCallService

HistoryService

History Groupings

Geographic

Campus

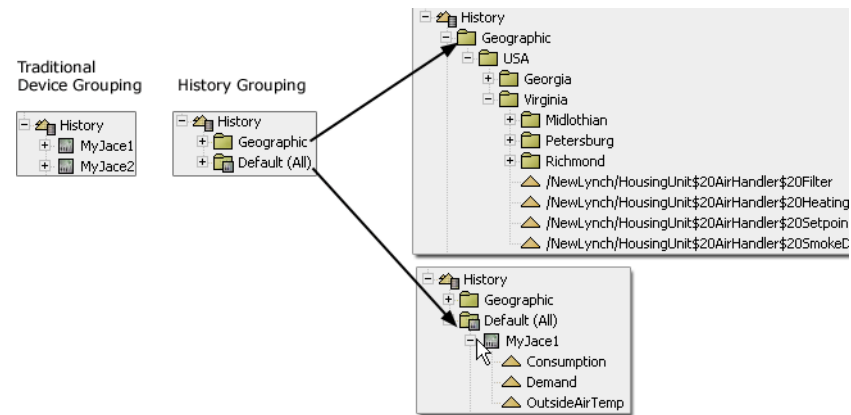
History Group Manager

Name	Enabled	History Properties To Group By
Geographic	true	state;city
Campus	true	building;floor

New Edit

The History Groupings component, located under the HistoryService, has properties that let you specify which history properties you want to filter on for organizing a station’s history space. These alternative navigation displays are based strictly on the history properties that you select and are a departure from the “device” or “driver” centered navigation model that builds a nav tree based around networks, devices, and points. With history properties, you can use one or more HistoryGroup components to set up "geographic", "functional", or any other desired types of organization of your history space. You can create multiple navigation schemes and use them, as needed, or use the default history navigation structure which always remains visible. History grouping allows you to simultaneously display histories in multiple nav paths, as shown below.

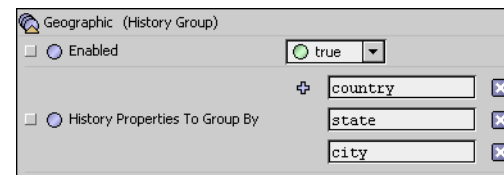
Figure 4-56 One custom history grouping with the default history navigation



About the History Group property sheet view

An example of the HistoryGroup component property sheet view is shown below.

Figure 4-57 HistoryGroup property sheet

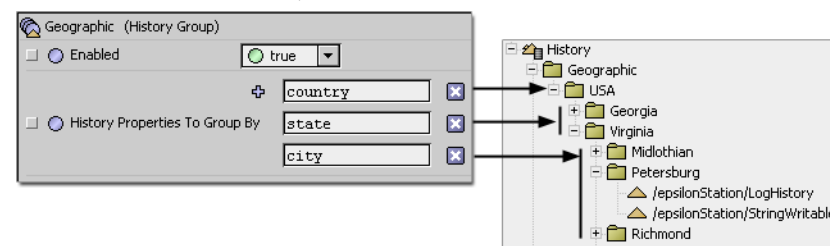


The HistoryGroup component has the following properties available in its property sheet view:

- **Enabled**
Select the `true` option to enable or the `false` option to disable the selected HistoryGroup.
- **History Properties To Group By**
This property provides fields that you can use for organizing your histories. Each completed field represents a specific property name that must match exactly to a history property name that belongs to one or more histories. Add or delete additional grouping parameters using the **+** button (to add) and the **x** button (to delete) fields. For example, to filter by the country, state, and city, history properties that exist in your histories, create three Group fields and type in the exact (case-sensitive) property names. When you click the **Save** button, the corresponding navigation tree is created under the History space node, as shown below:

Note: Notice in the following illustration that the order of the “history properties to group by” (in this case: “country”, “state”, “city”) determines the sub folder ordering.

Figure 4-58 Example history group



About the History Group Manager view

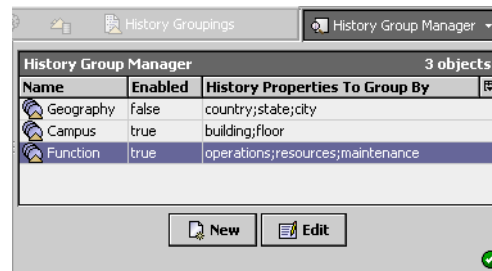
As the default view of the History Groupings component, the history group manager shows all history groups in the station and displays the following columns in a table view:

- **Name**
This is the identifier for the grouping. You can select a row and **Rename** or **Edit** the group using the control buttons at the bottom of the view.

- **Enabled**
This column indicates whether or not the group is actively filtering. If the Enabled value is set to `False`, then no custom grouping displays under the history space node for that group. If no history groups are present or enabled, the default device-oriented history organization displays.
- **History Properties To Group By**
This column displays a summary list of the history properties that are set to filter the presentation of the histories. Individual properties are separated by a semicolon (;).



You can double-click on any entry-row to open the **Edit** dialog box for editing the group. This table has the standard table features. Use the **Table Options** menu in the top right corner of the history table to modify the table view or to export the data in the view, as desired. Refer to “[Table controls and options](#)” on page 2-18 for information about using the tabular view, including the **Table Options** menu.

Figure 4-59 History Group manager view



Name	Enabled	History Properties To Group By
Geography	false	country;state;city
Campus	true	building;floor
Function	true	operations;resources;maintenance

About history nav shortcuts

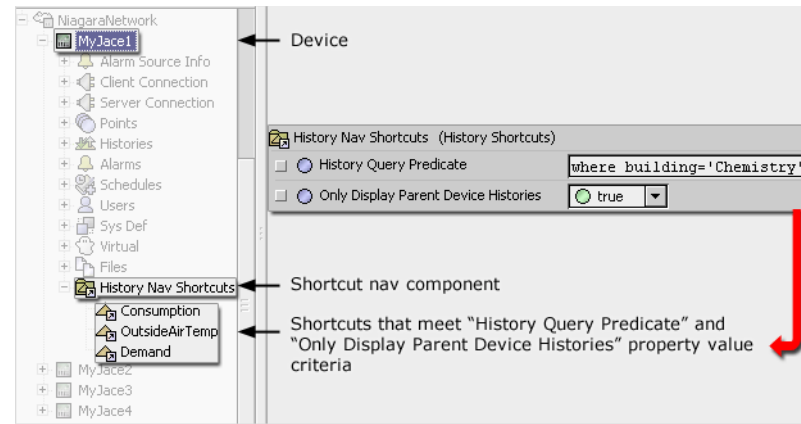
Starting in AX-3.5, history nav shortcuts  provide convenience navigation links to histories. You can place these shortcuts anywhere in a station to provide a filtered list of individual history shortcuts . The default property values of this component are set to display links to all histories that are under the nav shortcut ancestor device's name. You can edit properties in the property sheet view to display only the histories you want. History nav shortcuts have a Nav Container view and a History Chart Builder view.

Note: *The history shortcuts are not actual histories but links that provide convenient access to a history when you double-click on the shortcut. You cannot drag and drop the history shortcuts onto a Px page and relativize them.*

The history shortcut has the following properties:

- **History Query Predicate**
This property allows you to filter the histories by entering a value as a BQL query predicate value. For example, you could enter the following text string: `where state='Georgia'` (note the single quotes) to filter for all histories that have a state property with the value set to “Georgia”. This property works in conjunction with the Only Display Parent Device Histories property.
- **Only Display Parent Device Histories**
This property allows you to restrict the history shortcut filtering to those histories that match the ancestor device of the history nav shortcut component. If set to `true`, only shortcuts to histories that are identified with the history shortcuts parent device name are displayed. If set to `false`, shortcuts to all valid (those that match the “BQL query predicate value”) are displayed. In cases where the parent device name is not a valid history device name, you can add a dynamic “historyDeviceName” property, as described below. This property works in conjunction with the History Query Predicate property.

Figure 4-60 Example History Nav Shortcut property values



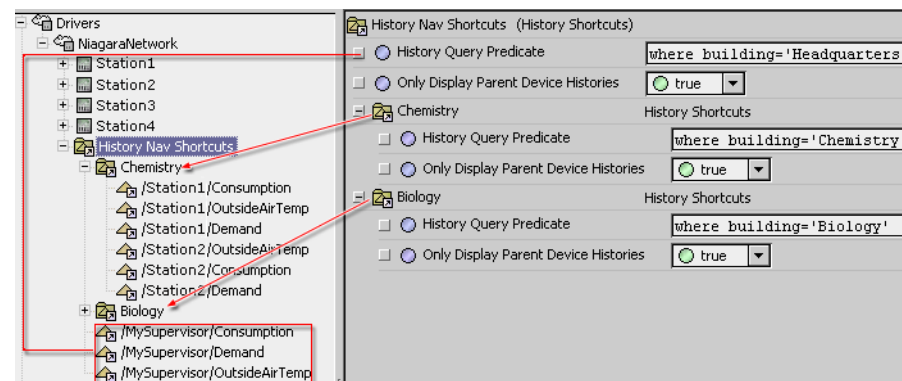
- historyDeviceName (optional)**
 Using the History Nav Shortcuts component slot sheet view, you can add a “historyDeviceName” string property to the History Nav Shortcuts property sheet. This additional property allows you to filter for an alternate history device name (in case the History Nav Shortcuts parent device’s name is not a valid history device).
 For example, if you place a History Nav Shortcut under a device that has a name “SqlServerDatabase” and a history ID from that RDBMS device are named “/myStation/currentTemp”, then a shortcut for this history is not displayed when the Only Display Parent Device Histories is set to `true`. However, if you add a historyDeviceName property to the component and set a value of “myStation” in that property, a shortcut for that history displays under the History Nav Shortcuts component.

About nesting history shortcuts

History Nav Shortcuts can be nested in order to build hierarchies of shortcut navigation. In the following illustration, note the following points:

- The History Query Predicate property filters to show the shortcuts in the nav tree. Since the Only Display Parent Device Histories value is set to `true` and the shortcut is not a child of any of the station devices (Station1, Station2, etc.) these shortcuts include histories that have a “building” property set to a value of “Headquarters” *and* are included anywhere in the history space. If the shortcut is placed under Station1 with these same property settings, then only the Station1 “Consumption”, “Outside-AirTemp”, and “Demand” histories would display.

Figure 4-61 Example nested History Nav Shortcuts














- The History Nav Shortcuts named “Chemistry” shows all shortcuts to histories with a building property set to “Chemistry”.
- The History Nav Shortcuts named “Biology” shows shortcuts to histories with a building property set to “Biology”. Since the Only Display Parent Device Histories value is set to `true` and the shortcut does not have a parent “device” these are histories that have a “Biology” property set to a value of “Headquarters” *and* are included anywhere in the history space.

CHAPTER 5

About Alarms

Figure 5-1 Typical alarm table display

	30-Apr-07 2:49:11 PM EDT	Normal	0 Acked / 128 Unacked	PreheatTemp	de
	30-Apr-07 2:49:11 PM EDT	Normal	0 Acked / 95 Unacked	MaTemp	de
	30-Apr-07 12:03:13 PM EDT	Normal	0 Acked / 127 Unacked	ColdDaTemp	de
	27-Apr-07 1:23:06 PM EDT	Offnormal		SuperNumericWriter	de
	27-Apr-07 12:33:13 PM EDT	Normal	0 Acked / 14 Unacked	HotDaTemp	de
	27-Apr-07 12:33:03 PM EDT	Normal	0 Acked / 7 Unacked	HotDuctSp	de
	27-Apr-07 12:33:03 PM EDT	Normal	0 Acked / 7 Unacked	ColdDuctSp	de
	27-Apr-07 12:09:23 PM EDT	Normal	0 Acked / 3 Unacked	PowerMonitorService	de
	27-Apr-07 12:04:09 PM EDT	Normal	0 Acked / 3 Unacked	PowerMonitorService	de
	26-Apr-07 12:26:59 PM EDT	Offnormal	0 Acked / 1 Unacked	LonNetwork DynamicDevice	de
	22-Apr-07 12:46:01 PM EDT	Normal	0 Acked / 3 Unacked	LonNetwork Cvahu	de

Alarms notify people that a predefined set of parameters has been met. Typically, these are “offNormal” or “Fault” parameters that are configured to notify specified recipients about the specified condition and to record the conditions that exist when the monitored point meets these parameters. The “normal” parameters for an individual point are properties that may be set and edited, as desired, by a user with proper access and privileges.

- **Alarm**

An alarm is used to indicate that some value is not within an appropriate or expected range. For example, the normal operating temperature range of a device may be 70 to 100 degrees F. You can set the “out of range” parameters to generate an “alarm” if the operating temperature exceeds the upper limit or goes below the lower limit of this range.

- **Alert**

This is an alarm that does not have a “normal” state. For example, a motor may require lubrication after every 400 hours of operation (this is not an “out of range” condition). Using the alarming function in Workbench, you can configure an extension to send an email *alert* to you when the 400 hours runtime has accumulated on the motor.

Alarm examples

The following are examples of possible ways that alarms are used:

- **Out of operating range notification (offNormal)**

An alarm is most commonly used to indicate that some value is not within an appropriate or expected range. For example, the normal operating temperature range of a device may be 70 to 100 degrees F. You can set the “out of range” parameters to generate an “alarm” if the operating temperature exceeds the upper limit or goes below the lower limit of this range.

- **Advisory notification (alert)**

You may use an alarm in situations to report on a parameter that does not really have a “normal” state. For example, a motor may require lubrication after every 400 hours of operation (this is not an “out of range” condition). Using the alert function, a system integrator can setup a control point that monitors accumulated device run-time and sends an email *alert* notification at or before the 400 hours run-time has occurred.

- **Device fault notifications (fault)**

Some devices may report values that are so far out of range that it is obvious that there is a device or system “fault” that needs attention. For example, if a device with a normal operating temperature of between 70 to 100 degrees reports a temperature of 0 degrees F or 1000 degrees F, then it is probable that there is a device or system fault and that the reported temperature is not the actual temperature at the device. The system engineer or supervisor can set parameters and enable alarms for a separate notification for values that are judged to be “faults” as opposed to simply “out of range”.

Overview of the alarming process

Each alarm is a single record in the alarm database that changes throughout its life cycle. An alarm has four general “life cycle” states that it may be in:

- **alarm**
this is the initial state of an alarm and this state remains in effect until the alarm is acknowledged.
- **acknowledged**
this is an alarm that has been acknowledged by the recipient. Acknowledged alarms may be normal alarms.
- **normal**
is an alarm that has a current status of “normal”.
- **acknowledged normal alarm**
is a normal alarm that has been acknowledged.

Note: An alarm is considered “open” when it is NOT (acknowledged and normal) or NOT (acknowledged and alert). Open alarms display in the alarm console. Cleared alarms do not display in the alarm console. The following table shows the conditions that result in an alarm being “open” or “cleared”.

Table 5-1 Alarm “Open” and “Cleared” conditions

Alarm State	Acknowledge State	Open or Cleared
Offnormal (or Fault)	Unacknowledged	Open
Offnormal (or Fault)	Acknowledged	Open
Normal	Unacknowledged	Open
Normal	Acknowledged	Cleared

Typically, an alarm provides some visual and audible indication that a limit or value is met or exceeded. Alarm notifications may be routed and displayed in a variety of ways, including the following:

NiagaraAX alarming is comprised of three primary actions:

- Creating alarms (refer to [“About alarm sources”](#) on page 5-3)
- Routing alarms (refer to [“About the alarm service”](#) on page 5-7)
- Managing alarms (refer to [“About the alarm database maintenance view”](#) on page 5-30)

Figure 5-2 Simplified alarm process



Under these three processes NiagaraAX provides highly specific and flexible alarming life cycle management, including the following alarming functions:

- **creating alarms**
Alarms are generated by components using an alarm extension (refer to [“About alarm extensions and components”](#) on page 5-3). The alarm extensions *create* the alarm whenever specified values are outside of “normal” range. Those alarms are then handled by the alarm service (refer to [“About the alarm service”](#) on page 5-7).
- **routing alarms**
Alarms are handled by the **alarm service** which, in addition to allowing you to specify routing destinations (including archiving destinations), provides notification and acknowledgement parameters.
 - notification
Alarms are routed to one or more recipients based on their alarm class. This includes notification by email, at the alarm console, a lineprinter, or at remote stations.
 - acknowledgment
Alarms may require a response from those who are notified. If a required acknowledgement is not received within an optionally-specified time, alarms can be “escalated” and re-routed to other designated alarm recipients.

- **managing alarms**
alarms are archived in records that are managed by the alarm database management interface.

The following list outlines the basic steps in the alarming process:

- **add alarm extensions to components**
Choose alarm extensions to match the component data type.
- **configure alarm extensions**
Configure the alarm parameters to define when a component is in an alarmed state.
- **route alarms**
Configure the alarm parameters to define where an alarm record is sent.
- **manage the alarm archive**
Use the alarm archive management tools to manage the *storage* of all alarms.

About alarm sources

The parameters that are set in a point's "alarm point extension" determine *when* that point is in an "alarmable" condition. You use the alarm extension to configure routing options, issue alarms to the alarm service, and update the alarm to reflect a status change when the parent point goes back to a normal state. The extension also notifies the point that an acknowledgement has been received. To set up the alarm source on an object you need to:

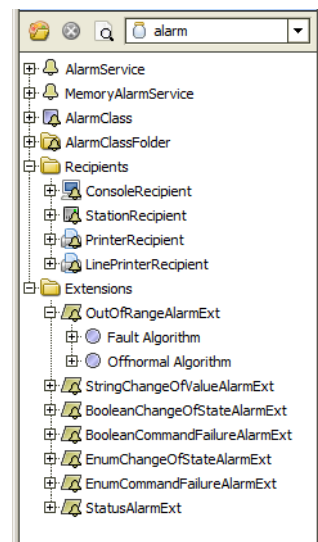
- add a proper alarm extension to the component (see [About alarm extensions and components](#))
- configure the alarm extension to meet your needs (see [About alarm extensions](#))

About alarm extensions and components

To set up alarming on a component you need to add an alarm extension to the component's property sheet. Refer to "[About alarm extensions](#)" on page 3-14 for more details about the alarm extensions. 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. For a list of alarm extensions types and their associated point types, refer to "[Types of alarm extensions](#)" on page 3-15.

Alarm extensions are contained in the alarm Palette, as shown in [Figure 5-3](#).

Figure 5-3 Alarm extensions in alarm palette



For details on adding extensions to a property sheet, see "[To add an extension to a component property sheet view:](#)" on page 9-17. Once the extension is added, configure the extension to meet your alarming requirements. Refer to "[About alarm extension properties](#)" on page 5-3 for more details.

About alarm extension properties

Each alarm extension has a set of configurable properties that allow you to specify the alarming conditions and certain routing options. [Figure 5-4](#) illustrates and the following list describes each alarm extension as it appears in a component property sheet.

Figure 5-4 Alarm extension properties

The screenshot shows the 'OutOfRangeAlarmExt' configuration window. It includes the following properties:

- Alarm Inhibit:** A dropdown menu set to 'false (ok)'.
- Inhibit Time:** A time field set to '00000h 00m 00s' with a range of '[0ms - +inf]'.
- Alarm State:** A dropdown menu set to 'Normal'.
- Time Delay:** A time field set to '00000h 00m 00s' with a range of '[0ms - +inf]'.
- Time Delay To Normal:** A time field set to '00000h 00m 00s' with a range of '[0ms - +inf]'.
- Alarm Enable:** Two checkboxes, 'toOffnormal' and 'toFault', both checked.
- Alarm Timestamps:** A section containing:
 - Alarm Time:** '27-Aug-2012 03:41 PM EDT'
 - Ack Time:** '27-Aug-2012 03:42 PM EDT'
 - Normal Time:** '27-Aug-2012 03:42 PM EDT'
 - Count:** '3'
- To Fault Times:** A section containing:
 - Time In Current State:** '+000006h 53m 18s'
- Source Name:** A text field containing '%parent.displayName%'.
- To Fault Text:** A text area containing '***FAULT***' and 'Value not used by system!'.
- To Offnormal Text:** An empty text area.
- To Normal Text:** A text area containing '%alarmData.sourceName% is at %alarmData' and 'which is inbetween the low limit of %alarmData.lowLimit% and alarmData.hig'.

Note: Starting in AX-3.7, all alarm message “text” properties are now “multi-line capable,”—for instance, properties “To Fault Text,” “To Offnormal Text,” and “To Normal Text” properties as shown in Figure 5-4, as well as possible “High Limit Text” and “Low Limit Text” properties under any “Offnormal Algorithm” and “Fault Algorithm” containers (not shown). As needed, you can now press “Enter” to start new lines in these text fields, making lines the desired length and/or providing spaces between lines.

Following, are the alarm extension properties descriptions:

- **Alarm Inhibit**

A `true` value in this field prevents alarm generation due to any transition or state change.

For example, if there is a `true` value in this field and an Offnormal value is reached, a “toOffNormal” status will *not* be communicated. When the state returns to Normal, a “toNormal” status will also *not* be communicated. A `false` value in this field will prevent alarms from being inhibited (even if an Inhibit Time is set).

Note: The purpose of the Alarm Inhibit property is to prevent unintended alarms, such as in after-hours situations where a piece of equipment is turned off. A difference between Alarm Inhibit and Alarm Delay is that the Alarm Inhibit is a boolean value (`true/false`) and may be controlled by another device (for example, an ON/OFF value of a fan).

If the Alarm Inhibit value is set to `true`, the [Inhibit Time](#) property qualifies the behavior.

- **Inhibit Time**

The value of this property affects the length of time that the current [Alarm Inhibit](#) state will remain in effect after an Alarm Inhibit state change. For example, when an Alarm Inhibit value changes from `true` to `false`, alarm generation continues to be inhibited for a time that is specified by the Inhibit Time property value. When an Alarm Inhibit value changes from `false` to `true`, alarm generation may continue to be inhibited for a time that is dependent on the point type, as shown in [Table 5-2](#).

Table 5-2 Time until Alarm Inhibit state change affects alarm generation

Alarm Inhibit transition	Point type	
	Discrete point	Numeric point
True to False	Inhibit Time property value	Inhibit Time property value
False to True	3 x Inhibit Time property value	0 (zero)

- **Alarm State**
This field displays the current alarm state of the component: Normal, Low Limit, High Limit, or Fault.
- **Time Delay**
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.
This is 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 Time Delay property value before an alarm is generated. The general purpose of the Time Delay property is to provide a means to prevent nuisance alarms that may be caused by a momentary change in a state value (Normal, Low Limit, High Limit).
Time Delay applies to properties that transition both *in* and *out* of alarm states. Therefore, an alarm status may continue to display as Offnormal (for example) for a time (equal to the Time Delay) after the value has come back into Normal parameters. The Time Delay is the minimum time period that a normal condition must exist before the object comes *out* of alarm.
Note: Typically, when both Alarm Delay and Alarm Inhibit properties are used, Time Delay is less (shorter) than Alarm Inhibit.
- **Time Delay to Normal**
This is the minimum time period that a normal condition must exist before the object may return to normal status.
- **Alarm Enable**
Select any of the options to individually enable the generation of alarms when the following transitions occur:
 - toOffnormal
This is the time that the offNormal event occurred.
 - toFault
This is the time that the Fault event occurred*Note:* No alarms will be generated unless an Alarm Enable option is selected.
- **To Offnormal Times**
This property displays four pieces of information that are related to the time that the component status changed to Offnormal. A “null” value means that the event has not occurred. See [Figure 5-5](#).
 - Alarm Time
This property displays the time that the To Offnormal event occurred.
 - Ack Time
This property displays the time that the alarm was acknowledged.
 - Normal Time
This property displays the time that the To Normal event occurred.
 - Count
This field displays the total number of Offnormal events.

Figure 5-5 To Offnormal times

Alarm Timestamps	
<input type="checkbox"/> Alarm Time	28-Jul-2005 03:46 PM EDT → Offnormal Event Timestamp
<input type="checkbox"/> Ack Time	28-Jul-2005 03:47 PM EDT → Acknowledgement Timestamp
<input type="checkbox"/> Normal Time	28-Jul-2005 03:48 PM EDT → Normal Event Timestamp
<input type="checkbox"/> Count	75 → Total Offnormal Events

- **To Fault Times**
This property displays four pieces of information that are related to the time that the component status changed to a Fault state. A “null” value means that the event has not occurred. See [Figure 5-6](#).
 - Alarm Time
This property displays the time that the To Fault event occurred.
 - Ack Time
This property displays the time that the alarm was acknowledged.

- **Normal Time**
This property displays the time that the To Normal event occurred.
- **Count**
This field displays the total number of To Fault events.

Figure 5-6 To Fault times

Alarm Timestamps	
<input type="radio"/> Alarm Time	28-Jul-2005 04:24 PM EDT
<input type="radio"/> Ack Time	null
<input type="radio"/> Normal Time	28-Jul-2005 04:25 PM EDT
<input type="radio"/> Count	255

Offnormal Event Timestamp
Acknowledgement Timestamp
Normal Event Timestamp
Total To Fault Events

- **To Normal Times**
This property displays the time that the component transitioned to a normal state.
- **Time in Current State**
This property displays the elapsed time since the component transitioned to the current state.
- **Source Name**
This property displays the name of the alarm source. If you use the default script setting %parent.displayName%, the source name field will show the “display name” of the alarm extension parent. You can edit this script or type in a literal string to display.
- **To Fault Text**
(multi-line in AX-3.7 and later, see [Note](#): on page 5-4) Enter text that you would like to display when the component transitions to a Fault status. This may be overridden if text is entered under the “Fault Algorithm” properties “High Limit Text” and/or “Low Limit Text”, if applicable.
- **To Offnormal Text**
(multi-line in AX-3.7 and later, see [Note](#): on page 5-4) Enter the text that you would like to display when the component transitions to an Offnormal (alarm) state.
***Note:** This text may be overridden if text is entered under the “Offnormal Algorithm” properties “High Limit Text” and/or “Low Limit Text”, if applicable.*
- **To Normal Text**
(multi-line in AX-3.7 and later, see [Note](#): on page 5-4) Enter the text that you would like to display when the component transitions to a Normal status.
- **Hyperlink Ord**
Use this property enter or choose an Ord, a BQL Query or a path to a component that you would like to associate with an alarm status on the component you are configuring. When an alarm is reported in the console, the Hyperlink button is active and uses this path. Use the folder icon to browse to the file that you want to link to. Click the arrow icon to the right of the folder icon to test the Ord that you enter.
- **Sound File**
Enter or choose the path to a sound file that will execute when the current component is in an alarm state. Use the folder icon to browse to the file that you want to use. Click the arrow icon to the right of the folder icon to test the path that you enter.
- **Alarm Icon**
Use this property to enter or choose the path to a graphic file that will added to the display in the “timestamp” column of the alarm table in the Console Recipient view. Use the folder icon to browse to the file that you want to use. Click the arrow icon to the right of the folder icon to test the path that you enter.
- **Fault Algorithm**
This property when available, displays fault parameters. Different options are available here, depending on the alarm extension being used. In the case of an **OutOfRangeAlarmExt** (for status numeric points), separate “High Limit”, “Low Limit”, and “Deadband” parameters apply, along with associated limit enable properties and “High Limit Text” and “Low Limit Text” properties.
- **Offnormal Algorithm**
Use this property to enter high and low limit parameters for numeric components and select the alarm value (on or off) to designate the alarm state for a boolean component. In the case of an **OutOfRangeAlarmExt** (for status numeric points), separate “High Limit”, “Low Limit”, and “Deadband” parameters apply, along with associated limit enable properties and “High Limit Text” and “Low Limit Text” properties.
- **Alarm Class**
Use this property to select an alarm class from the option list. The alarm class specifies the alarm routing options for this component. Refer to [“About alarm class”](#) on page 5-32 for details about alarm class.

- **Meta Data**
Use this property to enter new facets using the meta data property. See [“About point facets”](#) on page 3-7 for details about facets.

About alarm instructions

Each alarm can have “instructions” assigned to it so that any time an alarm is generated, the instructions are presented with the alarm notification (in the Alarm Record dialog box) to provide information that may be important or helpful to the user. Instructions are created, assigned, and edited from the Instructions view. Refer to [“About the Instructions Manager view”](#) on page 5-27 for details about the Instructions view.

About notes

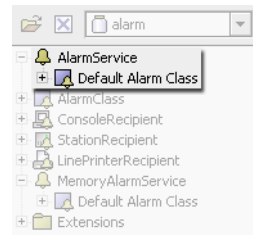
Notes are simple text entries that are associated with a particular alarm. It is possible to add a Note to one alarm or to multiple alarms. Alarm records that have notes are indicated by a “note” icon. Refer to [“Types of Alarm Details View dialog boxes”](#) on page 5-38 for more information about Notes.

About the alarm service

Note: In addition to the standard file-based alarm service, described in this section, there is a memory alarm service available. These two services should not run in the same station. Refer to [“About memory alarm service”](#) on page 5-8 for details about the memory alarm service.

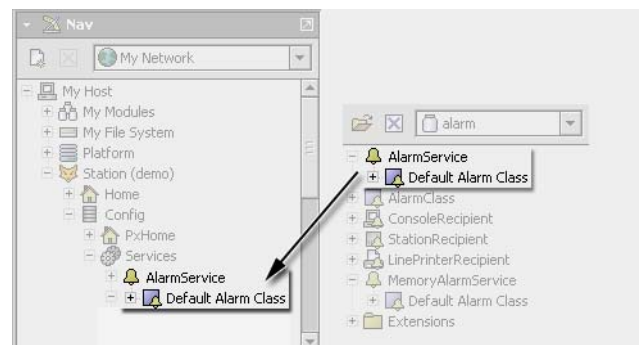
Each station may contain a single alarm service that coordinates the routing of alarms within the NiagaraAX framework and maintains the alarm database. The alarm service is available in the alarm palette, as shown in [Figure 5-7](#).

Figure 5-7 Alarm service in the alarm palette



If you do not have the alarm service in your active station, you can add it by dragging and dropping a copy from the alarm palette, as shown in [Figure 5-8](#).

Figure 5-8 Adding the alarm service



The alarm service may contain one or more **alarm classes** (refer to [“About alarm class”](#) for more information about alarm class). An alarm class may route alarms to one or more alarm recipient types (refer to [“Types of alarm recipients”](#) on page 5-36 for more information about alarm recipient).

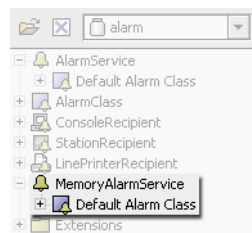
The routing process includes getting alarm acknowledgements from the recipients back to the source, and alarm notifications from the source to the recipients. The default view (wire sheet view) of the alarm service makes it easy to visualize the relationships between the alarm class and the alarm recipient. These relationships are created by linking the alarm class to the alarm recipient, as described in [“About alarm class”](#) on page 5-32. In addition to the wire sheet and property sheet views, there are several other views of the alarm service. See [“Types of alarm service views”](#) on page 5-26 for a description of the alarm service views.

About memory alarm service

Note: *MemoryAlarmService is an alarm service that provides an alternative to the standard file-based AlarmService. A station should have only one alarm service. Do not enable both the standard AlarmService and MemoryAlarmService on the same station.*

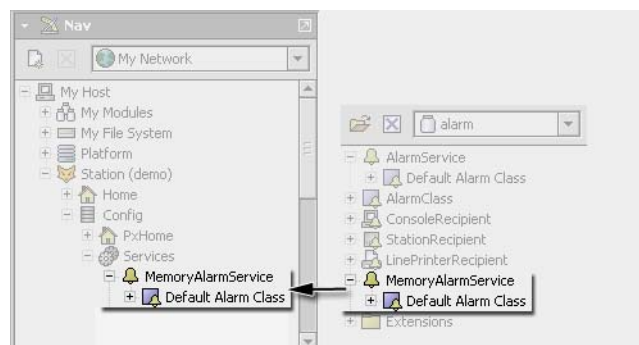
Each station may contain a single alarm service that coordinates the routing of alarms within the NiagaraAX framework. The memory alarm service is available in the alarm palette, as shown in [Figure 5-9](#).

Figure 5-9 Memory alarm service in the alarm palette



If you do not have the memory alarm service in your active station, you can add it by dragging and dropping a copy from the alarm palette, as shown in [Figure 5-10](#).

Figure 5-10 Adding the alarm service



Like the alarm service, the memory alarm service may contain one or more alarm classes. An alarm class may route alarms to one or more alarm recipient types.

The routing process for the memory alarm service is the same as the standard file-based alarm service and includes alarm acknowledgements from the recipients back to the source, and alarm notifications from the source to the recipients. The default view (wire sheet view) of the alarm service 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, as described in [“About alarm class”](#) on page 5-32. The memory alarm service has the same views available as the standard alarm service.

The main distinction of MemoryAlarmService is that when you use this service, alarms are *not* stored persistently on the station host (typically a JACE) as they are with the standard file-based alarm service. 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 notifications. Alarm records are stored in memory and therefore they are lost in the case of a power failure.

About the On Call Service

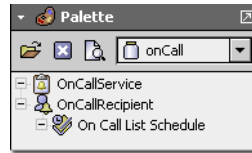
Available in NiagaraAX-3.5 and later, the On Call Service is a customization of the alarm service that allows you to send alarm notifications to users based on a priority list (“Contact List”). This service initiates an email (or text message) notification for designated alarms, sending the notification sequentially, as alarms escalate, to users based on their assigned priority.

As one or more users on the contact list are sent alarms, any user may acknowledge the alarm and halt the escalation process. However, after a specified time interval, if no user acknowledges the alarm then the alarm is escalated and an email or text message is sent to the next specified user. Alarms are continually escalated until they are acknowledged or until they reach the final escalation level.

Note: On Call Service alarm escalation is a similar concept to alarm escalation, as described in [“About alarm escalation”](#) on page 5-33. The primary functional difference is that the On Call Service allows you to set up a flexible user contact list and schedule that list using the standard scheduler interface.

The On Call Service is available in the onCall palette, as shown in [Figure 5-11](#).

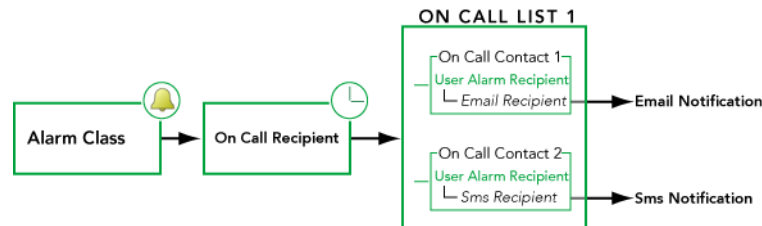
Figure 5-11 On Call Service in the onCall palette



You can add the On Call Service to a station by dragging and dropping a copy from the onCall palette.

The general process for the On Call Service is as follows:

Figure 5-12 On Call Service alarm routing



1. A designated alarm class receives an alarm notification and sends the alarm notification to an On Call Recipient.
2. The On Call Recipient sends the alarm notification to the *active* On Call List.

Note: The “Active” On Call List is designated by the On Call List Schedule.

3. The On Call List specifies which users (On Call Contacts) are notified.
4. The On Call Contact properties include a reference to a User Alarm Recipient which specifies how the alarm notification is sent. For example, User Alarm Recipient types may include:
 - EmailRecipient (sends alarm notification by email)
 - SmsRecipient (sends alarm notification by text message)
5. Unacknowledged alarms may be escalated and sent, sequentially to other members of the On Call List.

See the following sections for related On Call Service information:

- [“About the On Call List”](#) on page 5-10
- [“About the On Call List Manager view”](#) on page 5-10
- [“About the On Call Contact”](#) on page 5-11
- [“About the On Call Contact Manager view”](#) on page 5-12
- [“About the On Call User Report view”](#) on page 5-13
- [“Exporting the On Call User Report view”](#) on page 5-14
- [“Using Spy pages with the On Call Service”](#) on page 5-24

About On Call Service alarm data in the alarm console

In addition to the default alarm console alarm data columns, many On Call alarm data facets are available for optional viewing directly in the alarm console.

Figure 5-13 On Call Service alarm data facets viewed in the alarm console

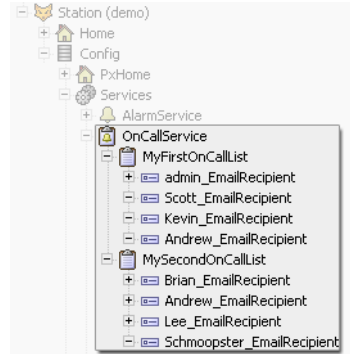
Open Alarm Sources				4 Sources / 168 Alarms		
Timestamp	Source State	On Call User	On Call Recipient	On Call List	On Call Start Time	
06-Jan-10 3:58:14 PM EST	Offnormal	User2	OnCallRecipient	OnCallList1	06-Jan-10 3:58 PM EST	
06-Jan-10 3:41:25 PM EST	Offnormal	User2	OnCallRecipient	OnCallList1	06-Jan-10 3:41 PM EST	
06-Jan-10 3:34:37 PM EST	Offnormal	User2	OnCallRecipient	OnCallList1	06-Jan-10 3:34 PM EST	
06-Jan-10 1:59:11 PM EST	Offnormal	User2	OnCallRecipient	OnCallList1	06-Jan-10 1:59 PM EST	

You can add On Call alarm data columns to the alarm console view by using the **Add Alarm Data Column** dialog box, as described in [“Hide, show, add, and remove alarm data columns in the alarm console”](#) on page 5-37.

About the On Call List

The On Call List component (AX-3.5 and later) is designed to contain a set of one or more On Call Contacts. You can create, edit, or delete unique On Call Lists from the On Call List Manager view. After creating an On Call List, it appears under the OnCallService node in the nav pane, as shown below. It is also available as an option on the Event Output property in the On Call List Schedule's Scheduler view, where you may select it as a scheduled event.

Figure 5-14 On Call Lists under the OnCallService node



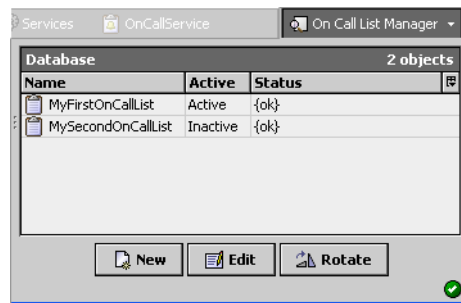
Like other scheduled outputs, you can assign more than one On Call List to a single day. The On Call List is active only during the scheduled day and time. The On Call List active status is displayed in the On Call List property sheet view as well as in the On Call List Manager view.

Note: When assigning On Call List events in the Scheduler view, make sure On Call List event times are contiguous.

The On Call List component has the following properties, visible in the property sheet view:

- **Status**
This display-only property indicates the condition of the component. For example: ok, down, disabled are possible status conditions.
- **Active**
This property displays either Active or Inactive, according to its current state, as defined by the On Call List Schedule. Only one On Call List is active at a time.
- **Last Fault Cause**
This read-only field displays the reason for the latest On Call List fault, if there has been one.
- **Ordinal**
This read-only field is a unique identifier for the particular On Call List. The OnCallService tracks the next free ordinal number.

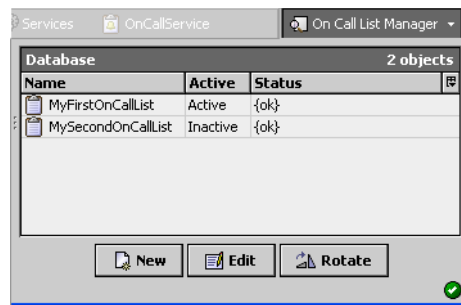
Figure 5-15 On Call List Manager view



About the On Call List Manager view

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.

Figure 5-16 On Call List Manager view



You can double click on any row to display the list in the On Call Contact Manager view. By default, the following columns display in the On Call List Manager view:

- **Name**
This column shows the unique name of the On Call List
- **Active**
This column displays the current current state of the list as either Active or Inactive. Only one On Call List is active at a time.
- **Status**
This column displays the current condition of the component. For example: ok, down, disabled are possible status conditions.

At the bottom of the view, the following control buttons are displayed:

- **New**
Click this button to open the **New** dialog box. Use the **New** dialog box for creating a new On Call list.
- **Edit**
Click this button to open the **Edit** dialog box. Use the **Edit** dialog box for editing (renaming) one or more selected On Call lists.
- **Rotate**
“Rotating” priorities means that the Priority value of each user in the contact list is reassigned to the next user in the list in a revolving manner. All selected lists are affected when you click the **Rotate** button.

Note: *If any OnCallContacts are added, removed, deleted, reordered, or has any of its properties modified, any alarms that the OnCallList is currently handling are resent to the contacts starting at the top of the list, by priority.*

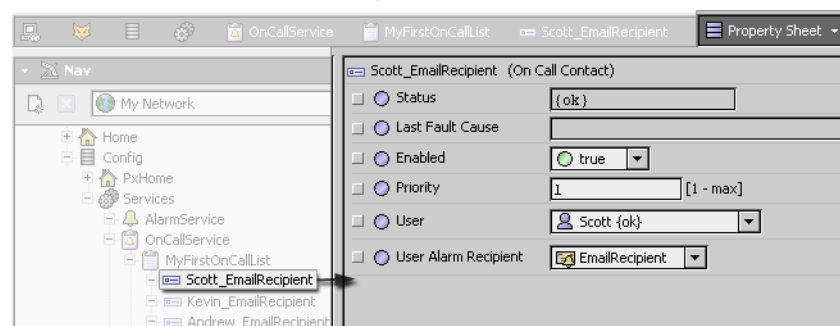
About the On Call Contact

The On Call Contact (AX-3.5 and later) is an entry item in an On Call Contact List and is comprised of a single User and additional configurable properties that are used for routing alarm notifications. Only users that exist as entries under the User Service are available for assignment to an On Call Contact List.

Note: *Any time an On Call Contact property is changed, the On Call List initiates a notification cycle.*

On Call Contact properties may be edited in the property sheet view or in the **Edit** dialog box that is available from the On Call Contact Manager view.

Figure 5-17 On Call Contact Property Sheet view



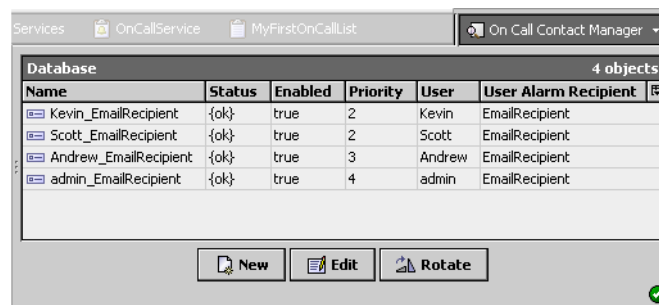
The On Call Contact has the following properties:

- **Status**
This display-only field shows the current condition of the On Call Contact. For example, the current Status may be `ok`, `disabled`, `fault`, or some other condition.
- **Last Fault Cause**
If there has been a fault, this field displays a text message indicating the possible reason for the last fault.
- **Enabled**
This field allows you to enable or disable an On Call Contact by selecting the `true` or `false` option, respectively.
- **Priority**
This field allows you to set an integer value for the current contact's priority. The Priority property specifies the order in which the On Call Service sends alarm notifications to the On Call Contact.
Note: One or more On Call Contacts may share the same Priority number value. The On Call Service sends an identical notification to all contacts that have the same priority number.
Highest priority On Call Contacts (alarm priority property value with least integer value) for their particular On Call Contact List receive the first alarm notification. If alarms are not acknowledged within a designated time, subsequent notifications are sent to contacts in order of increasing priority number. For example, contacts with priority 2 would be notified if a contact with priority value 1 does not acknowledge an alarm within a specified time. Contacts with priority 3 would be subsequently notified, if necessary.
- **User**
This option list displays the currently assigned User and provides options for selecting a new User. All Users that are under the User Service are presented as options to select. To view or edit User information, open the User Manager view (the default view of the User Service).
- **User Alarm Recipient**
This option list displays any alarm recipient options that are available for the On Call Service. The EmailRecipient or SmsRecipient are available only if they have been added under the AlarmService. The EmailRecipient component is available in the email palette and the SmsRecipient is available in the Sms palette. You must choose an alarm recipient to designate the way alarm notifications are routed to the selected On Call Contact.

About the On Call Contact Manager view

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.

Figure 5-18 On Call Contact Manager view



Name	Status	Enabled	Priority	User	User Alarm Recipient
Kevin_EmailRecipient	{ok}	true	2	Kevin	EmailRecipient
Scott_EmailRecipient	{ok}	true	2	Scott	EmailRecipient
Andrew_EmailRecipient	{ok}	true	3	Andrew	EmailRecipient
admin_EmailRecipient	{ok}	true	4	admin	EmailRecipient

You can double-click on any row to display the contact in the On Call User Report view. By default, the following columns display in the On Call Contact Manager view:

- **Name**
This column shows the unique name of the On Call Contact. The format for a contact name is always a combination of the assigned User name and the User Alarm Recipient. For example, a user “Scott” could have a Contact name of “Scott_EmailRecipient” or “Scott_SmsRecipient”.
- **Status**
This column shows the current condition of the On Call Contact. For example, the current Status may be `ok`, `disabled`, `fault`, or some other condition.
- **Enabled**
This column displays the current condition of each On Call Contact in the table — Enabled (`true`) or Disabled (`false`).

- **Priority**
This field displays the currently assigned priority value for each contact in the table. The Priority property specifies the order in which the On Call Service sends alarm notifications to the On Call Contact.
Note: One or more On Call Contacts may share the same Priority number value. In this case the On Call Service sends an identical notification to all contacts.
- **User**
This column displays the assigned User for each contact in the table.
Note: If a User is deleted, the contact displays as a {fault} state.

Figure 5-19 On Call Contact Manager with a contact in fault state

Name	Status	Enabled	Priority	User	User Alarm Recipient
Kevin_EmailRecipient	{ok}	true	2	Kevin	EmailRecipient
null_EmailRecipient	{fault}	true	2		EmailRecipient
Andrew_EmailRecipient	{ok}	true	3	Andrew	EmailRecipient
admin_EmailRecipient	{ok}	true	4	admin	EmailRecipient

- **User Alarm Recipient**
This column displays the assigned User Alarm Recipient for each contact in the table.

At the bottom of the view, the following control buttons are displayed:

- **New**
Click this button to open the **New** dialog box. Use the **New** dialog box for creating a new On Call Contact.
- **Edit**
Click this button to open the **Edit** dialog box. Use the **Edit** dialog box for editing the selected On Call Contact.
- **Rotate**
“Rotating” priorities means that the Priority value of each user in the contact list is reassigned to the next user in the list in a revolving manner. All contacts in the list are affected when you click the **Rotate** button.

About the On Call User Report view

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 the On Call Contact Lists that they are assigned to. As with other tables, you can show or hide columns and use other standard table controls and options that are provided in the **Table Options** menu. The **Table Options** menu is located in the top right corner of the table and is described in “[Table controls and options](#)” on page 2-18. Also, the export function is available using the Export icon on the toolbar or by selecting **File > Export** from the main menu.

An example of the alarm class summary view is shown in [Figure 5-20](#).

Figure 5-20 On Call User Report view

On Call Recipient	On Call List	Priority	Start	End
OnCallRecipient	MyFirstOnCallList	2	01-Nov-09 12:00 AM EDT	01-Nov-09 12:00 AM EDT
OnCallRecipient	MyFirstOnCallList	2	02-Nov-09 12:00 AM EST	02-Nov-09 4:00 PM EST
OnCallRecipient	MySecondOnCallList	1	02-Nov-09 5:00 PM EST	02-Nov-09 12:00 AM EST
OnCallRecipient	MyFirstOnCallList	2	03-Nov-09 12:00 AM EST	03-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	04-Nov-09 12:00 AM EST	04-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	05-Nov-09 12:00 AM EST	05-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	06-Nov-09 12:00 AM EST	06-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	07-Nov-09 12:00 AM EST	07-Nov-09 12:00 AM EST
OnCallRecipient	MyFirstOnCallList	2	08-Nov-09 12:00 AM EST	08-Nov-09 12:00 AM EST
OnCallRecipient	MyFirstOnCallList	2	09-Nov-09 12:00 AM EST	09-Nov-09 4:00 PM EST
OnCallRecipient	MySecondOnCallList	1	09-Nov-09 5:00 PM EST	09-Nov-09 12:00 AM EST
OnCallRecipient	MyFirstOnCallList	2	10-Nov-09 12:00 AM EST	10-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	11-Nov-09 12:00 AM EST	11-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	12-Nov-09 12:00 AM EST	12-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	13-Nov-09 12:00 AM EST	13-Nov-09 4:00 PM EST
OnCallRecipient	MyFirstOnCallList	2	14-Nov-09 12:00 AM EST	14-Nov-09 12:00 AM EST
OnCallRecipient	MyFirstOnCallList	2	15-Nov-09 12:00 AM EST	15-Nov-09 12:00 AM EST
OnCallRecipient	MyFirstOnCallList	2	16-Nov-09 12:00 AM EST	16-Nov-09 4:00 PM EST

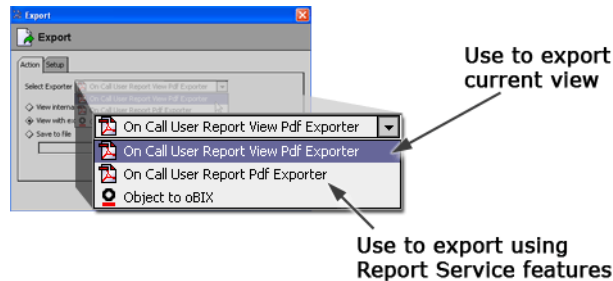
The On Call User Report view has the following columns:

- **On Call Recipient**
This is the name of the On Call Recipient being used to coordinate alarms.
- **On Call List**
This column displays the name of the On Call List that the On Call Contact is a member of. A contact may be a member of more than one contact list.
- **Priority**
This column displays the current priority assigned to the contact.
- **Start and End**
This column displays the scheduled active day and time for the specific On Call List.

Exporting the On Call User Report view

You can export On Call User Reports (AX-3.5 and later) and print them or email them. To initiate a report export, select the Export icon on the toolbar or by selecting **File > Export** from the main menu. There are two PDF output export options available:

Figure 5-21 Exporting On Call User Reports

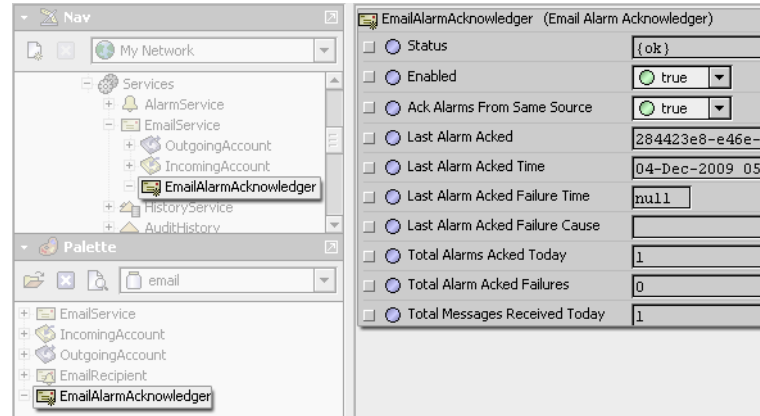


- **On Call User Report View Pdf Exporter**
You can use this option to immediately create a PDF file from the current view using the standard workbench export function. In this case you use the report view time range fields to set the range of the report.
- **On Call User Report Pdf Exporter**
Use this option to export reports using the Report Service. This allows you to schedule and email reports from stations on a periodic basis, if desired. In this case you need to add the Report View ord to the Source property in the ExportSource component (using the **Export Source Wizard**).

About the Email Alarm Acknowledger

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 alarm notifications that are sent out using the On Call Service or directly from the Email Service.

Figure 5-22 Email Acknowledger locations and properties



The EmailAcknowledger properties include:

- **Status**
This display-only field shows the current condition of the Email Acknowledger. For example, the current Status may be ok, disabled, fault, or some other condition.
- **Enabled**
This field allows you to enable or disable an EmailAlarmAcknowledger component by selecting the true or false option, respectively.
- **Ack Alarms From Same Source**
When you select the true option for this property, in addition to the current alarm, all previous alarms from this source are acknowledged with a single email reply. For example, a single control point may be going in and out of alarm repeatedly and have 23 unacked alarms reported and displayed in the alarm console. If you set this option to true all 23 alarms are acknowledged when the EmailAlarmAcknowledger receives a single email acknowledgement. If the false option is selected, a single alarm is acknowledged and 22 unacknowledged alarms remain.
- **Last Alarm Aced**
This field displays the identity of the last alarm that was acknowledged.
- **Last Alarm Aced Time**
This field displays the time that the last alarm was acknowledged by the EmailAlarmAcknowledger component.
- **Last Alarm Aced Failure Time**
If there has been a failure in the attempt to acknowledge an alarm, this field displays a message indicating the time of the last failure.
- **Last Alarm Aced Failure Cause**
If there has been a failure in the attempt to acknowledge an alarm, this field displays a message indicating the possible reason for the last failure.
- **Total Alarms Aced Today**
This field displays the current number of alarms that have been acknowledged for the day. This number is reset to zero at midnight.
- **Total Alarm Aced Failures**
This field displays the current number of alarm acknowledgement attempts that have failed for the day. This number is reset to zero at midnight.
- **Total Messages Received Today**
This field displays the current number of email messages that have been received for the day. This number is reset to zero at midnight.

In order to use the EmailAlarmAcknowledger component, you need to do the following:

- **Setup the EmailService**
See [“About the EmailService”](#) on page 5-16.

- **Setup the IncomingAccount**
See [“About the IncomingAccount”](#) on page 5-19.
- **Setup the EmailAcknowledger**
See [“About the Email Alarm Acknowledger”](#) on page 5-15.
- **Setup the EmailRecipient**
See [“About the Email Recipient \(email-EmailRecipient\)”](#) on page 5-43.

In addition, note the following requirements for configuring the EmailRecipient when using the EmailAlarmAcknowledger.

- **A User is required**
When properly configured, the EmailAlarmAcknowledger looks at incoming emails and checks to see if the the sender’s email address matches the email address of a User that is currently in the local UserService database. No alarm is acknowledged unless it is first determined that the acknowledgment email is sent from a matching user and email address or phone number. This validated User name is also assigned to the alarm database as the person who has acknowledged the alarm.
- **A UUID required (NiagaraAX-3.5)**
The UUID is the unique identifier that is assigned to every alarm and is used for keeping track of the alarm. The UUID must be in the subject of the acknowledging email. To ensure this, you must enter the following text in the EmailRecipient Subject property: UUID: %uuid%.

Figure 5-23 Adding UUID to the EmailRecipient Subject property (NiagaraAX-3.5)



See, [“About the Email Recipient \(email-EmailRecipient\)”](#) on page 5-43 for more information about the emailRecipient.

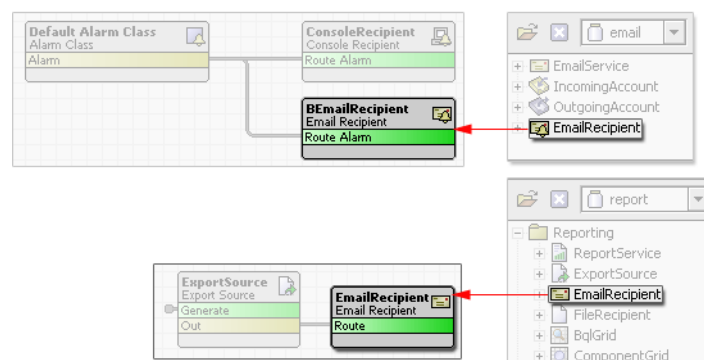
Note: In NiagaraAX-3.5, the EmailService only acknowledges alarms that contain the UUID of the alarm in the subject field of the EmailRecipient property view. Use the following syntax for configuring the EmailRecipient Subject property for use with the EmailAlarmAcknowledger component: UUID: %uuid%
Starting in NiagaraAX-3.6 the message-id and reply-to email headers are used to track emails associated with an alarm. No alarm identification is required in the email subject or content.

About the EmailService

The Email service is required whenever you specify an "EmailRecipient" in your station or application. For example, you can route both alarms and reports via email using the EmailRecipient, as shown below.

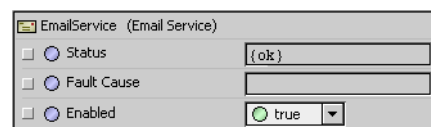
Note: The EmailRecipient component for alarms is located in the email palette and is different from the EmailRecipient that is used for reports (located in the ReportService palette). It is important to use the appropriate EmailRecipient component, as indicated in the illustration, below.

Figure 5-24 Alarm EmailRecipient and Report EmailRecipient require a configured EmailService



The EmailService property sheet view is shown below and includes the following properties:

Figure 5-25 EmailService property sheet view



- **Status**
This read-only view shows the current state of the EmailService ({ok}, {disabled}, {fault}, or other.)
- **Fault Cause**
This read-only field displays an error message if the EmailService is in a fault state.
- **Status**
This field allows you to enable (`true`) or disable (`false`) the EmailService.

The default view of the EmailService, the EmailAccountManager view, is shown below. This view displays a table of all the OutgoingAccounts and IncomingAccounts. Double-click on an account in the table to display the (default) property sheet view of the account.

Figure 5-26 Email Account Manager view

Config

Services

EmailService

Email Account Manager

Email Account Manager

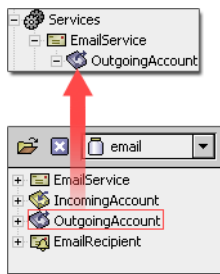
2 objects

Name	Hostname	Account	Pollrate	Status	
OutgoingAccount	mail.tridium.com	Softtest4	10sec	{ok}	
IncomingAccount	mail.tridium.net	Softtest4	20mins	{ok}	

About the OutgoingAccount

The email OutgoingAccount component is required for sending emails from a station. This component is located in the email palette and should be added under the EmailService component, as shown below.

Figure 5-27 OutgoingAccount is located in the email palette



The OutgoingAccount has the following properties:



Figure 5-28 OutgoingAccount property sheet view

OutgoingAccount (Outgoing Account)

Hostname	mail.acme.com
Port	25 [-1 - max]
Account	joejones
Password	*****
Pollrate	00000h 00m 10s [1sec - +inf]
Enabled	true
Status	{ok}
Last Poll Success	07-Jan-2010 03:40 PM EST
Last Poll Failure	null
Last Poll Failure Cause	
Debug	false
Use Ssl	false
Transport	Smtp
Connection Timeout	+00000h 01m 10s
Use Authentication	false
Reply To	joejones@acme.com
Persistent	false
Persistence Directory	file:^email
Allow Disabled Queueing	false
Queue Size	0
Max Queue Size	100 [1 - max]
Number Sent	2
Max Sendable Per Day	100 [1 - max]
Number Discarded	1
Last Discard	07-Jan-2010 09:21 AM EST
Last Discard Cause	BOutgoingAccount queue disabled, discard

- **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 "joejones@acme.com" on the host described above, the account name is simply "joejones". 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.

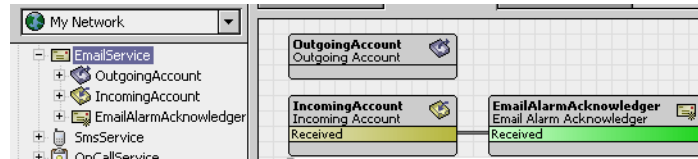
- **Use Ssl**
The Ssl (Secure Socket Layer) option is available on the OutgoingAccount starting in NiagaraAX-3.5. The option list allows you to disable Ssl (false) or enable it (true). Enable Ssl for communication with a host email server that requires it.
- **Transport**
This field allows you to select 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 (NiagaraAX-3.5 and later) displays a value to indicate how many emails did not successfully send.
 - **Last Discard**
This read-only property (NiagaraAX-3.5 and later) displays a date and time value to indicate when the last email did failed to send.
 - **Last Discard Cause**
This read-only property (NiagaraAX-3.5 and later) displays an error message that indicates the cause of the last email send failure.

About the IncomingAccount

The email IncomingAccount component is required for receiving emails in a station. One use of this component is for email alarm acknowledgement with the EmailAcknowledger component.

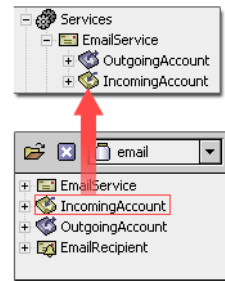
Note: *In order to use the EmailAcknowledger, you must set up the IncomingAccount under the EmailService and connect it's "received" slot to the EmailAcknowledger "received" slot, as shown below. More than one IncomingAccount may be connected to the EmailAcknowledger, if desired.*

Figure 5-29 Using the IncomingAccount with the EmailAcknowledger



This IncomingAccount component is located in the email palette and should be added under the EmailService component, as shown below.

Figure 5-30 IncomingAccount is located in the email palette



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: (1) change the Delivery Policy setting from Delete to Mark As Read or Mark as Unread OR (2) 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

The IncomingAccount has the following properties:

Figure 5-31 IncomingAccount property sheet view

- **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.
- **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 Host-name, 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 starting in NiagaraAX-3.5. 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.
- **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
 - Pop3
- **Delivery Policy**
This property (available starting in NiagaraAX-3.5) 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.

About the Email Recipient (report-EmailRecipient)

The report-EmailRecipient component is found in the Report palette and contains properties that allow you to specify where the report is to be emailed.

Note: *The EmailRecipient component for reports is located in the reports palette and is different from the Email Recipient that is used for alarms (located in the email palette). It is important to use the appropriate EmailRecipient component.*

The property sheet view, shown in [Figure 5-32](#), displays the properties that you configure for this function.

Figure 5-32 Email Recipient component property sheet view

EmailRecipient component properties include the following:

- **To**
This is the email address of the email recipient (person who is to receive the emailed report).
- **Cc**
This is the email address of anyone that is to receive a copy of the email (and attached report).
- **Bcc**
This is the email address of anyone that should receive a copy of the email (and attached report) but whose name you do not want to display in the delivered email.
- **Language**
This is the ISO 639 language code as two lower-case letters. For a list of codes, see the following:
<http://www.loc.gov/standards/iso639-2/langcodes.html>
- **Email Account**
This is an option list that allows you to choose the email account that you want to use for sending the report email.
- **Subject**
This is a text field that allows you to manually add text to the subject line of the email or use the following default text:
%reportName% from %sys().station.stationName% %time()%
- **Body**
This is a text field that allows you manually add text to the body of the email or use the following default text:
Attached is the %reportName% report from %sys().station.stationName% generated on %time()%.

About the SmsService

The SmsService is used for sending text messages to other cellular phones. The SMS driver allows retransmitting alarm information and sending general text messages.

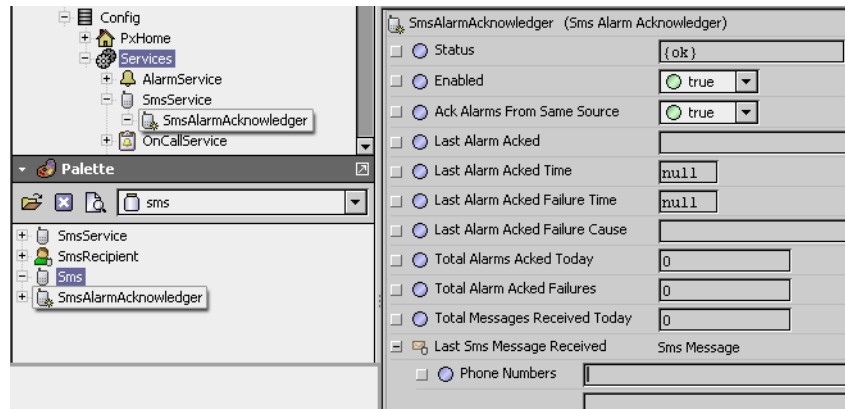
Note: The term 'SMS' is used to refer to the Simple Message Service technology available on GSM devices. All SMS Niagara components are referred to as 'Sms' components. This follows the style and conventions used in naming Niagara components, similar to the 'Bacnet' driver for BACnet.

For more details, refer to the *Sms Users Guide*.

About the Sms Alarm Acknowledger

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.

Figure 5-33 *SmsAlarmAcknowledger locations and properties*

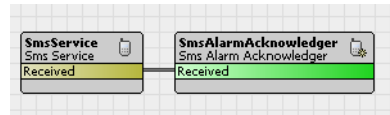


In order to use the `SmsAlarmAcknowledger` component, you need to add and configure the component. The `SmsAlarmAcknowledger` properties include:

- **Status**
This display-only field shows the current condition of the `SmsAlarmAcknowledger`. For example, the current Status may be `ok`, `disabled`, `fault`, or some other condition.
- **Enabled**
This field allows you to enable or disable an `SmsAlarmAcknowledger` component by selecting the `true` or `false` option, respectively.
- **Ack Alarms From Same Source**
When you select the `true` option for this property, in addition to the current alarm, all previous alarms from this source are acknowledged with a single text reply. For example, a single control point may be going in and out of alarm repeatedly and have 23 unacked alarms reported and displayed in the alarm console. If you set this option to `true` all 23 alarms are acknowledged when the `SmsAlarmAcknowledger` receives a single text acknowledgement. If the `false` option is selected, a single alarm is acknowledged and 22 unacknowledged alarms remain.
- **Last Alarm Aced**
This field displays the identity of the last alarm that was acknowledged by the `SmsAlarmAcknowledger` component.
- **Last Alarm Aced Time**
This field displays the time that the last alarm was acknowledged by the `SmsAlarmAcknowledger` component.
- **Last Alarm Aced Failure Time**
If there has been a failure in the attempt to acknowledge an alarm, this field displays a message indicating the time of the last failure.
- **Last Alarm Aced Failure Cause**
If there has been a failure in the attempt to acknowledge an alarm, this field displays a message indicating the possible reason for the last failure.
- **Total Alarms Aced Today**
This field displays the current number of alarms that have been acknowledged for the day. This number is reset to zero at midnight.
- **Total Alarm Aced Failures**
This field displays the current number of alarm acknowledgement attempts that have failed for the day. This number is reset to zero at midnight.
- **Total Messages Received Today**
This field displays the current number of `Sms` messages that have been received for the day. This number is reset to zero at midnight.
- **Last Sms Message Received**
This field displays the message and phone numbers associated with the last `Sms` message received.

In order to use the `SmsAlarmAcknowledger` component, you need to do the following:

- **Setup the `SmsService`**
See [“About the `SmsService`”](#) on page 5-22.
- **Setup the `SmsAlarmAcknowledger`**
See [“About the `Sms Alarm Acknowledger`”](#) on page 5-22. This includes linking the ‘Received’ topic on the `SmsService` to the ‘Received’ slot of the `SmsAlarmAcknowledger`, as shown, below.

Figure 5-34 SmsService “Received” topic linked to the SmsAlarmAcknowledger “Received” slot

- **Setup the SmsRecipient**

See [“About the Sms recipient”](#) on page 5-44.

In addition, note the following requirements for configuring the SmsRecipient:

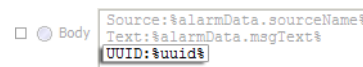
- **A User is required**

When properly configured, the SmsAlarmAcknowledger checks incoming text messages to see if the sender’s phone number matches the phone number of a User that is currently in the local UserService database. No alarm is acknowledged unless it is first determined that the acknowledgment message is sent from a matching user and phone number. When validated, the User name is assigned to the alarm database as the person who has acknowledged the alarm.

- **A UUID is required**

The UUID is the unique identifier that is assigned to every alarm and is used for keeping track of the alarm. For the SmsRecipient, the UUID must be in the body of the Sms message, so that it is present in the “reply to” acknowledging message. Also, this requires that the cell phone device is configured to “reply to sender with original text” (or the user may manually enter the UUID).

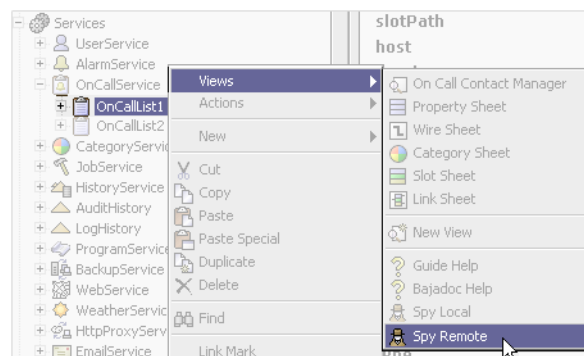
To ensure that the original text contains the required UUID, you must enter the following text in the SmsRecipient Body property: `UUID: %uuid%`.

Figure 5-35 Adding UUID to the SmsRecipient Body property

For related information, refer to the *Sms Users Guide*.

Using Spy pages with the On Call Service

Spy Pages can provide helpful information for tracking alarms when using the On Call Service or for On Call “debugging”. Spy page views are available for the On Call Lists and the On Call Recipient. Open a Spy Page by right-clicking on the On Call List or Recipient in the nav tree and selecting **View > Spy Remote** from the popup menu, as shown below.

Figure 5-36 Displaying an On Call List Spy Page

Some of the important information provided by Spy Pages include: current On Call Priority assignment (“On Call Priority”), current On Call Contacts (“On Call Contact”), and current time remaining (“Remaining”) until a notification is sent to the next On Call Contact if an alarm is not acknowledged.

Note: To update Spy Page information, you must reload a Spy Page by clicking the **Refresh** button.

Figure 5-37 Example Spy Page information










On Call List	On Call Contact	On Call Priority	Start Time	End Time	Remaining	Source
OnCallList1	User2_EmailRecipient	1	9:02 AM	9:32 AM	24mins 4.073sec	local:station: slot:/

Common alarm controls and indicators

The following alarm controls and indicators are common to several alarm views:









- **Alarm control buttons**

The following buttons appear in one or more alarm views:

-  **Acknowledge**
Use the **Acknowledge** button to acknowledge *all selected* alarms.
-  **Hyperlink**
Use the **Hyperlink** button to change the current view to the hyperlinked target associated with the selected alarm. If no hyperlink is associated with the alarm, the Hyperlink button is not available.
-  **Notes**
Click the **Notes** button to display the **Notes** dialog box and add a note to the selected alarm or alarms.
-  **Silence**  **Filter**  **Filter**, stop the audible notification associated with the selected alarm.

-  **Filter**
Click the **Filter** button to display the **Filters** dialog box for setting the filter parameters. Starting in NiagaraAX-3.4, the “Filter” text displays in the color red, as a reminder, to indicate that alarms are being filtered.
-  **Close**
Click the **Close** button to cancel the current interface actions without saving changes.

- **Alarm icons**

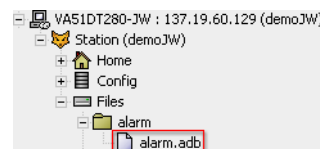
Alarm icons appear with color coding and symbolic images:

-  A red alarm icon in the table indicates that the current state of the alarm source is “offnormal” and “not acknowledged”.
-  An orange alarm icon in the table indicates that the current state of the alarm source is “alert” and is “not acknowledged”.
-  A yellow alarm (gold) icon in the table indicates that the current state of the alarm source is “offnormal” but is “acknowledged”.
-  A green alarm icon in the table indicates that the current state of the alarm source is “normal” and “not acknowledged”.
-  A white alarm icon in the table indicates that the current state of the alarm source is “normal” and “acknowledged”.
-  A note alarm icon (it may be *any* color) in the table indicates that there is a note associated with the alarm
-  A link icon in the table indicates that the alarm has a link associated with it. When an alarm displays this icon, the **Hyperlink** button is also active.
-  An optional icon may display if it is setup in the alarm properties. If included, this graphic appears at the left end of the alarm record row.

About the alarm database

The alarm database typically resides in a station file system, under the station’s “alarm” folder, as shown in [Figure 5-38](#).

Figure 5-38 Alarm database location



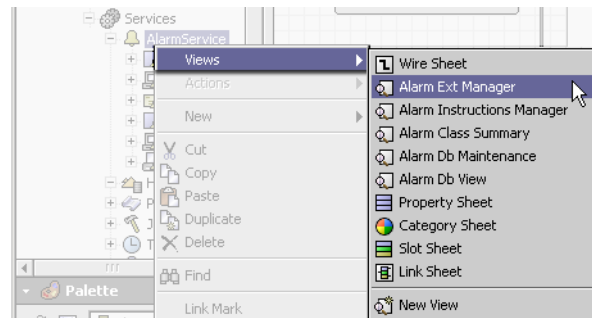
Starting in NiagaraAX-3.2, it is possible to use the `rdAlarmService` and configure a relational database (RDBMS) for storage of alarm records. This alternative to the standard alarm database requires an `RdbmsNetwork` connection to a supported database. See the *NiagaraAX-3.x RdbmsNetwork Driver Guide* for more details.

Note: Use of an RDBMS to store NiagaraAX alarm records is different than exporting histories to an RDBMS. Exported histories also still reside in the station's default history database (`^\\history\\segN\\(etc)`), whereas, configuring for RDBMS storage of alarms replaces use of the station's default alarm database (`^\\alarm\\alarm.adb`).

Types of alarm service views

You can view the alarm service in different ways in Workbench. [Figure 5-39](#) shows a menu of views that are available using either the Workbench view selector or a popup menu.

Figure 5-39 Alarm service views available from popup menu



In addition to the standard Workbench views (Wire Sheet, Category Sheet, Slot Sheet, Link Sheet) there are several views specific to the alarm service, including the following:

- **Alarm Extension Manager view**
The alarm extension manager is a view of the alarm service that provides a record of information about each alarm extension. See [“About alarm extension manager”](#) on page 5-26 for details.
- **Instructions Manager view**
The Instructions Manager view provides a way to view, assign, and edit the instructions that can be assigned to alarms. See [“About the Instructions Manager view”](#) for details.
- **Alarm Class Summary**
The alarm class summary view provides a concise list of information relative to each alarm class. Refer to [“About the alarm class summary view”](#) on page 5-29
- **Alarm Database Maintenance**
[“About the alarm database maintenance view”](#) on page 5-30
- **Alarm Db view**
This view provides a “read-only” display of alarm data. See [“About the alarm database view”](#) on page 5-31 for details.
- **Alarm Service Property Sheet**
The alarm service property sheet is a typical Workbench property sheet view that shows all the alarm service slots. See [“About alarm service property sheet”](#) for details.

About alarm extension manager

The alarm extension manager view presents alarm extensions in a table format to make it easy for you to see all the alarm extensions that are associated with points in your station. As with other tables, you can show or hide columns and use other standard table controls and options that are provided in the **Table Options** menu. The **Table Options** menu is located in the top right corner of the table and is described in [“Table controls and options”](#) on page 2-18. Also, the export toolbar icon is available on the toolbar when the alarm extension manager view is displayed.

An example of the alarm extension manager is shown in [Figure 5-40](#).

Figure 5-40 Alarm extension manager

Alarm Source Exts						6 Extensions
Point	Extension	Alarm State	toOffnormal Enabled	toFault Enabled	Alarm Cla	
/Sampler/Float	OutOfRangeAlarmExt	Normal	true	true		Reset Column Widths
/Sampler/Boolean	BooleanChangeOfStateAlarmExt	Offnormal	true	true		Export
/Sampler/Boolean	BooleanCommandFailureAlarmExt	Normal	false	false		
/Sampler/MidPoint	OutOfRangeAlarmExt	Normal	true	false		✓ Point
/Sampler/FloatStatic	OutOfRangeAlarmExt	Normal	false	false		✓ Extension
/Sampler/AlarmDemo	OutOfRangeAlarmExt	Normal	true	false		✓ Alarm State
/Sampler/AlarmMin	OutOfRangeAlarmExt	Normal	true	false		✓ toOffnormal Enabled
/Sampler/AlarmMax	OutOfRangeAlarmExt	Normal	true	false		✓ toFault Enabled
/Sampler/EnumWritable	EnumChangeOfStateAlarmExt	Normal	false	false		✓ Alarm Class
/Sampler/EnumWritable	EnumCommandFailureAlarmExt	Normal	false	false		defaultAlarmC

The alarm extension manager view provides the following data fields:

- **Point**
This field indicates the point that is the parent of the listed alarm extension.

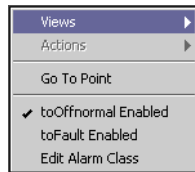
- **Extension**
This field identifies the type of extension that is listed (for example: “OutOfRangeAlarmExt”, StatusAlarmExt”, or others).
- **Alarm State**
This field displays the status of the listed alarm extension; for example, “High Limit” or “Normal”.
- **toOffnormal Enabled**
This field indicates (as True or False) if the toOffnormal parameter of the extension is enabled (true) or not (false).
- **toFault Enabled**
This field indicates (as True or False) if the toFault parameter of the extension is enabled (true) or not (false).
- **Alarm Class**
This field identifies the name of the alarm class that the extension is assigned to (defaultAlarmClass or other class).

Types of extension manager display features

Extension manager display features include:

- **Color coding**
A colored background on a row in the alarm extension manager table indicates that the alarm state of the parent component is *not* Normal.
- **Hyperlinking**
Double-click on any row in the Alarm Ext Manager view and the view will change to the property sheet view of the alarm extension that you clicked on.
- **Alarm extension editing (and batch editing)**
Select one or more (for batch editing) alarm extensions and use the popup menu (shown in [Figure 5-41](#)) to edit the extension(s).

Figure 5-41 Alarm extension manager popup menu



Edit alarm extensions from the alarm extension manager in the following ways:

- **toOffnormal Enabled** and **toFault Enabled** settings
You can toggle ON and OFF the **toOffnormal Enabled** and **toFault Enabled** parameters of each selected alarm extension using the popup menu.
- **Edit Alarm Class**
You can change the alarm class assignment of the alarm extensions. Select single or multiple (for batch editing) and from the popup menu select the **Edit Alarm Class** menu item. Choose the desired alarm class from the option menu and click the **OK** button to change the alarm class setting for the selected alarm extension(s).

About the Instructions Manager view

The Instructions Manager view displays a standard table-type report that provides a way to view, assign, and edit alarm instructions. The view is comprised of three primary panes:

- **Points**
This pane is located in the left half of the Master Instructions view and displays all points that are currently available for instruction assignment or editing. These points may have instructions assigned to them or they may have no instructions—they are simply the points that are available. The “Point” column contains the name of the control point source associated with the alarm and the “Conditions” column provides the name of property that holds the alarm instructions.
Note: You can select more than one point at a time by using the **Shift** key or **Ctrl** key. If the alarm instructions for all selected points are identical, then all associated instructions display in the **Point Instructions** pane. If there are any differences in alarm instructions for the selected points, no instructions display.

- **Point Instructions**

This pane is located in the top right portion of the **Assign Instructions** dialog box and it includes a list of instructions that are associated with the point or points currently selected in the Points pane.

You can add, remove, reorder, or edit instructions in this pane using the following controls:

Note: *Always remember to click the Save button immediately after making any changes.*

- **Add**

Clicking the **Add** button produces the **Add** dialog box that allows you to type in the text for an instruction. Once you add the instructions text to an alarm, the instructions appear in the Alarm Record dialog box (accessible from the **Open Alarm Sources** dialog box) for any new alarms associated with that point.

- **Remove**

Select an instruction then click the **Remove** button to delete the instruction.

- **Edit**

Select an instruction and then click the **Edit** button to open a dialog box that allows editing of the selected instruction.

- **Save**

Clicking the **Save** button commits any changes made to the point instructions. The Save action applies to all instructions and all points that are selected when the **Save** button is clicked. Changes are lost if the screen or if just the pane is refreshed before saving.

- **Move Up and Move Down**

Select an instruction in the point instruction window, then click either the **Move Up** or **Move Down** button to reorder the instructions in the window.

- **Master Instructions List**

This pane is located in the lower right portion of the Instructions Manager view and it displays all master instructions that are available for adding to the Point Instructions pane. Master Instructions allow you to choose and assign a pre-listed set of instructions to one or more points.

Note: *You can select more than one Master Instruction at a time using the Shift key or Ctrl key. If the alarm instructions for all selected points are identical, then all associated instructions display in the Point Instructions pane. If there are any differences in alarm instructions for the selected points, no instructions display.*

To add or remove one or more Master Instruction to one or more *selected points*, first select the Master Instruction(s) in the Master Instructions list and then click the **Add From Master List** button. Be sure to click the **Save** button immediately after making any edits.

Figure 5-42 shows an example of the Instructions Manager view with one point and one master instruction selected. You can Add, Remove and Edit Master Instructions using the following controls:

- **Add**

Clicking the **Add** button produces the **Add** dialog box which allows entering a text instruction to the Master Instructions list. The master instructions list allows you to enter instructions that are available to be assigned to any alarms. After entering instructions in the Master Instruction list, the instructions are available to be added to individual selected point instructions by first selecting the instruction in the list and then clicking the **Add From Master List** button.

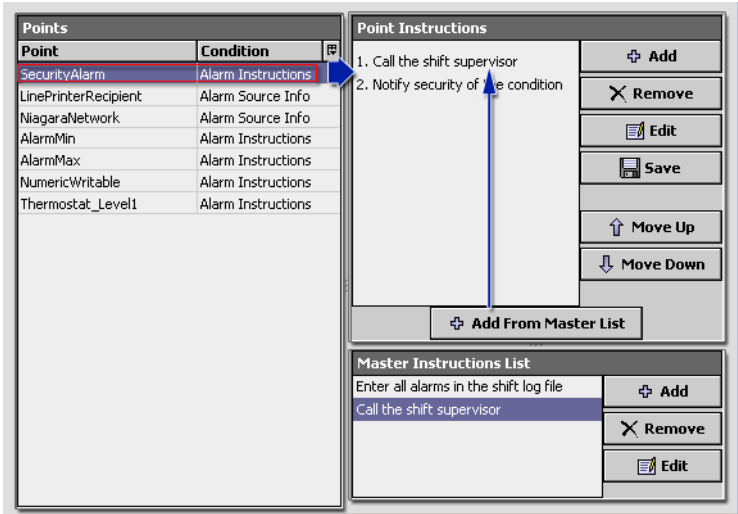
- **Remove**

Select an instruction in master instructions list and then click the **Remove** button to delete the instruction.

- **Edit**

Select an instruction and then click the **Edit** button to open a dialog box allowing the instruction to be modified.

Figure 5-42 Instructions Manager view with one point and one master instruction selected



About the alarm class summary view

The alarm class summary view provides a tabular presentation of data associated with all alarm classes that are assigned to the alarm service. As with other tables, you can show or hide columns and use other standard table controls and options that are provided in the **Table Options** menu. The **Table Options** menu is located in the top right corner of the table and is described in “Table controls and options” on page 2-18. Also, the export toolbar icon is available on the toolbar when the alarm class summary view is displayed.

An example of the alarm class summary view is shown in Figure 5-43.

Figure 5-43 Alarm class summary view

Alarm Class Summary						4 Alarm Classes
Name	Total	Open	In Alarm	Unacked	Last Alarm	
Default Alarm Class	194	117	1	116	25-May-05 9:48 AM EDT	
defaultAlarmClass1	49	24	0	24	25-May-05 9:28 AM EDT	
Total:	243	141	1	140		

The alarm class summary view has the following data fields:

- **Name**
This data field displays the alarm class name.
- **Open**
This data field displays the total number of open alarms associated with the alarm class.
- **In Alarm**
This data field displays the total number of alarms (points currently in alarm) associated with the alarm class.
- **UnAcked**
This data field displays the total number of unacknowledged alarms associated with the alarm class.
- **Last Alarm**
This data field displays the timestamp of the last alarm associated with the alarm class.
- **alarm.Class.priority**
This data field lists the priority of each alarm transition type. For example, “toOffnormal=1”, “to-Fault=60”, “toNormal=220”, and so on.
- **To Path String**
This data field identifies (as a string value) the path to the alarm class.

Alarm class summary view display features include:

Note: You can double-click on any row in the view and the view will change to the property sheet view of the alarm class that you clicked on. You can also use the popup menu to select a view.

About the alarm database maintenance view

The alarm database maintenance view presents alarm data in a table view to make it easy to view and edit the alarm database. As with other tables, you can show or hide columns and use other standard table controls and options that are provided in the **Table Options** menu. The **Table Options** menu is located in the top right corner of the table and is described in “Table controls and options” on page 2-18. Also, the filter icon and the notes icon appear on the workbench toolbar when the alarm database maintenance view is displayed.

An example of the alarm database maintenance view is shown in Figure 5-44.

Figure 5-44 Alarm database maintenance view

Alarm History						40 Alarms
Timestamp	Source State	Ack State	Ack Required	Source	Alarm Class	Priority
25-May-05 10:15:00 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:15:30 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:16:01 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:16:30 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:17:01 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:17:31 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:18:00 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:18:31 AM EDT	Normal	Unacked	true	AlarmDem		
25-May-05 10:19:00 AM EDT	Normal	Unacked	true	AlarmDem		

Context Menu Options:

- Reset Column Widths
- Export
- Timestamp
- Uuid
- Source State
- Ack State
- Ack Required
- Source
- Alarm Class
- Priority
- Normal Time
- Ack Time
- User
- Alarm Data
- Alarm Transition
- Last Update

Controls:

- Clear Old Records Before: 25-May-2000 10:00 AM EDT
- Clear All Before Selected Record
- Clear All Records
- Run Maintenance

The Alarm Database Maintenance view has two main sections:

- **Alarm History table**
This table fills the upper part of the view and contains a table of alarm history records.
- **Maintenance controls pane**
These controls are located below the table pane and contain the following controls and parameter options:
 - **Clear Old Records option**
allows you to clear alarm records before a certain date and time. The **Before** field is provided to allow you to set the date and time for removing old records.
 - **Clear All Before Selected Record option**
allows you to delete all records that have a timestamp earlier than the timestamp of the record that is currently highlighted in the table.
 - **Clear All Records options**
allows you to delete all records, regardless of the date.
 - **Run Maintenance button**
This button initiates the maintenance action.

The alarm database maintenance view has the following properties:

- **UUID**
This data field displays the unique universal identifier (UUID) of the alarm record.
- **Source State**
This data field displays the status of the listed alarm source; for example, “High Limit” or “Normal”.
- **Ack State**
This data field displays either “Acked” or “Unacked” to indicate whether the alarm has been acknowledged or not.
- **Ack Required**
This data field displays either “True” or “False” to indicate whether or not an acknowledgement is required for this alarm.
- **Source**
This data field displays the alarm source name.
- **Alarm Class**
This data field identifies the name of the alarm class that the extension is assigned to (defaultAlarm-Class or other class).
- **Priority**
This data field displays the priority number of the alarm.

- **Normal Time**
This data field displays the time that the alarm went to normal (if applicable)
- **Ack Time**
This data field displays the time that the alarm was acknowledged (if applicable)
- **User**
This data field identifies the name of the user that acknowledged the alarm. An unacknowledged alarm will display “unknown” in this field.
- **Alarm Data**
This data field presents a detailed list of alarm data.
- **Alarm Transition**
This data field displays the last transition type of the alarm.
- **Last Update**
This data field displays the time of the last alarm update.

About the Alarm details dialog box

You can double-click on any row in the alarm database maintenance view table and the **Alarm Details** dialog box appears, as shown in [Figure 5-45](#).

Figure 5-45 Alarm details

The screenshot shows a dialog box titled "Alarm Details" with a close button in the top right corner. The dialog contains a list of alarm parameters and their values. The parameters are grouped into sections: general alarm info, alarm data, alarm transition, and last update. The values are as follows:

timestamp	27-May-05 2:24 PM EDT
uuid	9623a249-3d89-4d08-b1d9-6f106cc1307a
sourceState	Offnormal
ackState	Unacked
ackRequired	true
source	local:/station:/slot:/Sampler/AlarmDemo/OutOfRangeAlarmExt
alarmClass	defaultAlarmClass
priority	50
normalTime	null
ackTime	null
user	Unknown User
sourceName	AlarmDemo
highLimit	95.0
presentValue	99.0
offnormalValue	99.0
deadband	0.0
alarmData	status: {alarm, overridden} @ 8
	toState: highLimit
	msgText:
	Count: 194
	fromState: normal
	TimeZone: America/New_York (-5/-4)
alarmTransition	Offnormal
lastUpdate	27-May-05 2:24 PM EDT

An "OK" button is located at the bottom right of the dialog.

The alarm details dialog box displays read-only details about the alarm that is displayed in the database maintenance view.

Types of alarm database maintenance

The database maintenance view allows you to perform the following database maintenance actions on the alarms, using the controls located at the bottom of the view:

- **Clear Old Records**
allows you to clear alarm records before a certain date and time. The **Before** field is provided to allow you to set the date and time for removing old records.
- **Clear All Before Selected Record**
allows you to delete all records that have a timestamp earlier than the timestamp of the record that is currently highlighted in the table.
- **Clear All Records**
allows you to delete all records, regardless of the date.

Refer to “[Managing alarms](#)” on page 9-22 for procedures about alarm database management.

About the alarm database view

Starting with NiagaraAX-3.2 the alarm Db view is available, as shown in [Figure 5-46](#). This view is similar to the alarm database maintenance view (see “[About the alarm database maintenance view](#)”), providing a table of history records but not allowing the operator to delete records.

The alarm database view only requires “read” access but allows operator-level personnel to view alarms in the alarm console. When a point is no longer in alarm, it is removed from the console. The primary purpose of this view is to provide operators a way to view alarms without having “admin” access to delete alarms from the alarm database.

Figure 5-46 Alarm database view

Alarm History							7 Alarms
Timestamp	Source	Ack State	Ack Req	Source	Alarm Class	Priori	
09-Nov-07 4:28:32 PM EST	Normal	Acked	false	AlarmTest	defaultAlarmClass	255	
09-Nov-07 4:33:26 PM EST	Normal	Acked	false	AlarmTest	defaultAlarmClass	255	
09-Nov-07 4:39:45 PM EST	Normal	Acked	false	AlarmTest	defaultAlarmClass	255	
09-Nov-07 4:40:15 PM EST	Normal	Unacked	true	AlarmTest	defaultAlarmClass	255	
19-Nov-07 9:41:43 AM EST	Normal	Unacked	true	AlarmTest	defaultAlarmClass	255	
19-Nov-07 9:42:00 AM EST	Normal	Unacked	true	AlarmTest	defaultAlarmClass	255	
19-Nov-07 9:42:07 AM EST	Normal	Unacked	true	AlarmTest	defaultAlarmClass	255	

You can double-click on any row in the alarm database maintenance view table and the **Alarm Details** dialog box appears, as shown in [Figure 5-45](#).

About alarm service property sheet

An example of the alarm service property sheet is shown in [Figure 5-47](#). The property sheet view of the alarm service displays all the recipients that are part of the alarm service and also has a capacity property that is visible and editable. The alarm service capacity property limits the number of records that are stored in the alarm service database. When the capacity is reached, the alarm records “roll”. This means that the newer records overwrite the oldest records.

Figure 5-47 Alarm service property sheet

AlarmService (Alarm Service)

Capacity: 10000 records [1 - 250000]

- + Default Alarm Class: Alarm Class
- + ConsoleRecipient: Console Recipient
- + BEmailRecipient: Email Recipient
- + defaultAlarmClass1: Alarm Class
- + ConsoleRecipient1: Console Recipient
- + LinePrinterRecipient1: Line Printer Recipient
- + StationRecipient: Station Recipient

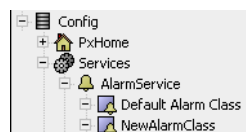
Note: For details about the alarm database manager views, refer to [“About the alarm database maintenance view”](#) on page 5-30. For details about the alarm class summary view, refer to [“About the alarm class summary view”](#) on page 5-29. For details about the alarm extension manager, refer to [“About alarm extension manager”](#) on page 5-26.

About alarm class

The alarm class is maintained by the alarm extension and all alarm classes available to the system are maintained as slots on the alarm service. The alarm class routes alarms with some similar set of properties along common routes and they serve as channels for like data. Alarm class manages the persistence of the alarms as needed via the alarm archive, but otherwise merely chains alarms from the alarm source via a topic. The alarm class also manages which alarms require acknowledgement and is the basis for visual grouping in the alarm console.

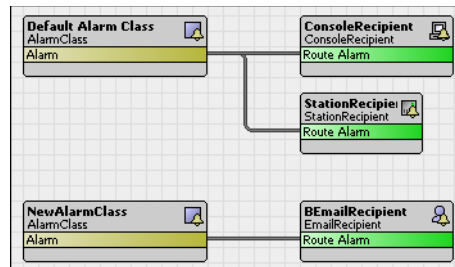
The alarm classes are located under the alarm service node in the nav side bar, as shown in [Figure 5-48](#).

Figure 5-48 Alarm class in nav side bar



You can set up multiple alarm classes in order to have a variety of alarming and routing options available for selection from the alarm extension properties. For example, you may set up an alarm class that routes to the console recipient and station recipient, while you may use another alarm class that routes alarms only to an email recipient, as shown in [Figure 5-49](#).

Figure 5-49 Alarm classes and linked alarm recipients



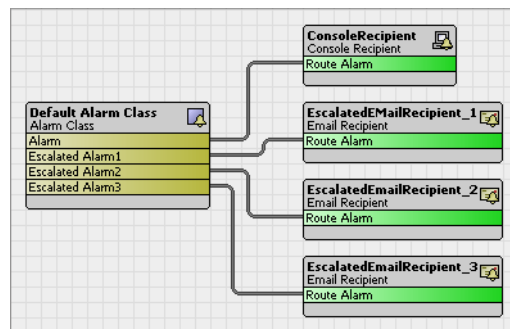
For more information about alarm recipients, refer to “[Types of alarm recipients](#)” on page 5-36. For information about configuring alarm classes, refer to “[About alarm class properties](#)” on page 5-34.

About alarm escalation

Alarm escalation is a feature that allows you to configure an alarm class so that unacknowledged alarms that are assigned to that class can be re-routed if they are not acknowledged within a specified amount of time. The following three levels of escalation are provided so that you can enable up to three opportunities to re-route an alarm notification as the alarm remains unacknowledged:

- Escalation Level1
- Escalation Level2
- Escalation Level3

Figure 5-50 Escalation levels routed to three email recipients



Each level may be routed to a different alarm recipient, as shown in [Figure 5-50](#), so that if an alarm remains unacknowledged long enough, it may be sent to as many as four different recipients (including the original recipient). However, if an alarm is acknowledged at any level, then it is not escalated to the next Level.

In addition to having an Enable property, each alarm escalation level has a Delay property that allows you to set the amount of time that you want to allow an alarm to remain at any level before it is moved to the next escalation level. Refer to “[About alarm class properties](#)” on page 5-34 for more information about Alarm Class properties.

About alarm class properties

Figure 5-51 Alarm class properties

The alarm class property sheet is shown in [Figure 5-51](#) and described in the following list:

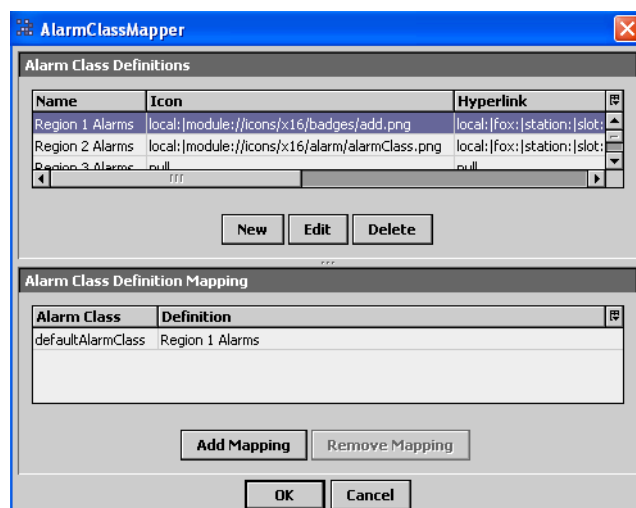
Note: Remember that these alarm class properties are specific to their parent alarm class and therefore, the properties do not contain values that are assigned to other alarm classes.

- **Ack Required**
Any alarm assigned to this alarm class is required to have the selected alarm transitions acknowledged when the associated option is selected (toOffnormal, toFault, toNormal, toAlert). If no option is selected, then no ack is required.
- **Priority**
These fields allow for custom entry of priority levels for each alarm transition type. The priority levels are associated with the alarms and are indicated graphically by colors that you can set in the alarm options dialog box. Refer to [“Alarm Console options”](#) on page 2-27 for details about setting priority colors.
- **Total Alarm Count**
This property displays the total number of alarms that have been assigned to the alarm class from all sources.
- **Open Alarm Count**
This property displays the current total number of open alarms.
Note: An alarm is considered open when it is not (acknowledged and normal) or not (acknowledged and alert).
- **In Alarm Count**
This property displays the total number of alarm conditions.
- **Unacked Alarm Count**
This property displays the total number of unacknowledged alarms.
- **Time of Last Alarm**
This property displays the time that the last alarm (assigned to this alarm class) was generated.
- **Escalation Level(*n*) Enable (where *n* is 1, 2, or 3)**
Select the true or false option of this property to enable or disable escalation at this level.
- **Escalation Level(*n*) Delay (where *n* is 1, 2, or 3)**
This property is the time since the alarm was generated until it is escalated. This is not the time between escalation levels. Set a time (in hours and minutes) that you want to allow an unacknowledged alarm to remain unacknowledged before you escalate it to the next level. One minute is the smallest increment that you can set for escalation level delay.

About alarm class mapping

Alarm class mapping is available using the **alarm class mapper** dialog box. This dialog box, shown in [Figure 5-52](#), provides a way for you to assign one or more alarm classes to a common set of alarm-handling parameters. One benefit of this type of mapping is that you can import alarms from a variety of alarm classes and have them display, link, or sound in a common manner. You do this by creating a **alarm class definitions** and then “mapping” existing alarm classes to these definitions. Open the alarm class mapper dialog box by choosing **Alarm Class Mapping...** from the **Alarm** menu. You can also open this dialog box in the **Alarm Console** window inside the **Workbench Options** dialog box.

Figure 5-52 Alarm class mapper

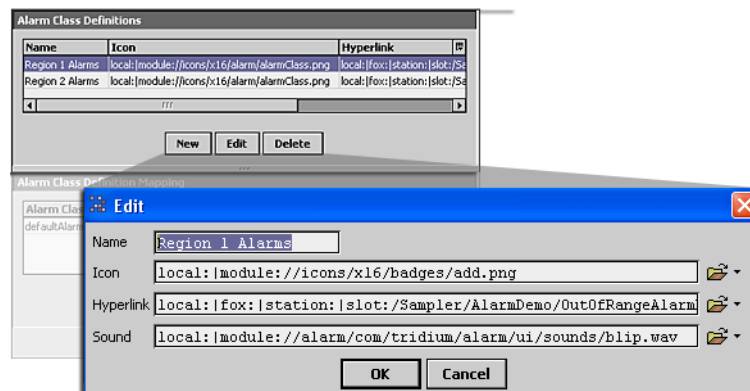


Note: The **Alarm** menu is available when you are in the alarm console view.

The alarm class mapper dialog box has two primary windows, as shown in Figure 5-54 and described below:

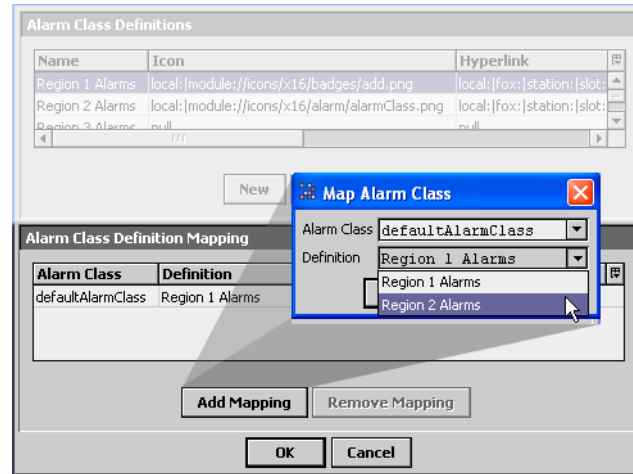
- **Alarm Class Definitions window**
This window is located at the top of the alarm class mapper dialog box. You can create, delete, or edit alarm mapper definitions in this window and then assign any existing alarm classes to these definitions, as shown in Figure 5-54. Click on the **Edit** button to display the **Edit** dialog box (shown in Figure 5-53) where you can configure the mapper alarm definition.

Figure 5-53 Editing a mapper alarm class



Note: The icons, hyperlinks, and sounds in the alarm mapper definition do not override those that are assigned in the alarm extension itself. If these parameters are assigned in both the alarm extension and in the alarm class mapper definition, the alarm class mapper definition parameters are ignored.

- **Alarm Class Definitions Mapping window**
This window is located at the bottom of the alarm class mapper dialog box. You can map any existing alarm classes to alarm mapper definitions. To do this, click the **Add Mapping** button to display the **Map Alarm Class** dialog box, as shown in Figure 5-54. Choose an available alarm class in the upper option box and assign it to the mapper alarm class definition in the lower option box. Click the **OK** button to complete the assignment. The mapping appears in the lower window. Remove mapping assignments by selecting the mapping in the lower window and clicking the **Remove Mapping** button.

Figure 5-54 Alarm class definitions mapping

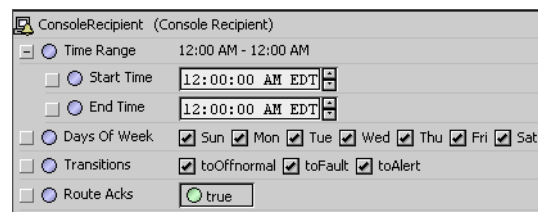
Types of alarm recipients

Alarm recipients are linked to an alarm class (from the alarm topic on the alarm class to the routeAlarm action on AlarmRecipient), as shown in [Figure 5-49](#). 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, as listed below:

- Console recipient (refer to [“About the console recipient”](#) on page 5-36)
- Station recipient (refer to [“About the station recipient”](#) on page 5-41)
- On Call recipient (refer to [“About the On Call Recipient”](#) on page 5-42)
- Email recipient (refer to [“About the Email Recipient \(email-EmailRecipient\)”](#) on page 5-43)
- Lineprinter recipient (refer to [“About the lineprinter recipient”](#) on page 5-45)
- Printer recipient (refer to [“About the printer recipient”](#) on page 5-46)

About the console recipient

The console recipient manages the transfer of alarms between the alarm history and the alarm console. For example, the console recipient gets unacknowledged alarms from the alarm history and updates the history when they are actually acknowledged. Console recipient properties are displayed and edited in the console recipient property sheet, as shown in [Figure 5-55](#).

Figure 5-55 Console recipient properties

Console recipient properties are described in the following list:

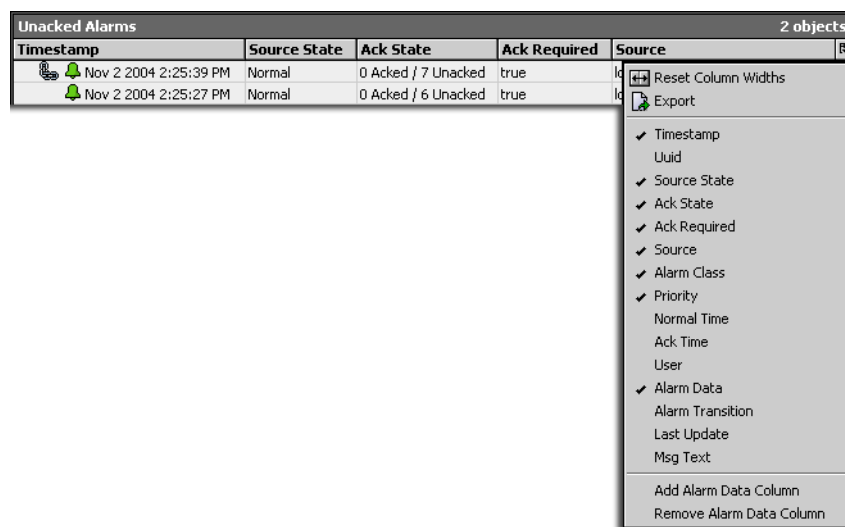
- **Time Range**
allows you to set a limited period of time during a day for collection of alarms, using the following parameters:
 - **Start Time**
a time of day to begin alarm collection, defined by hour, minute and second.
 - **End Time**
a time of day to end alarm collection, defined by hour, minute and second.
- **Days Of Week**
option boxes allow selection of specific days to collect alarms.
- **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.


The default view of the console recipient is the alarm console view (for details about the console view, see [About the alarm console](#)).

About the alarm console

The alarm console is a view of the console recipient. It allows you to see all the alarms that have been routed to the alarm console. Alarms are presented in a tabular view with columns that can be viewed or hidden using the drop-down menu in the top right corner of the column title bar.

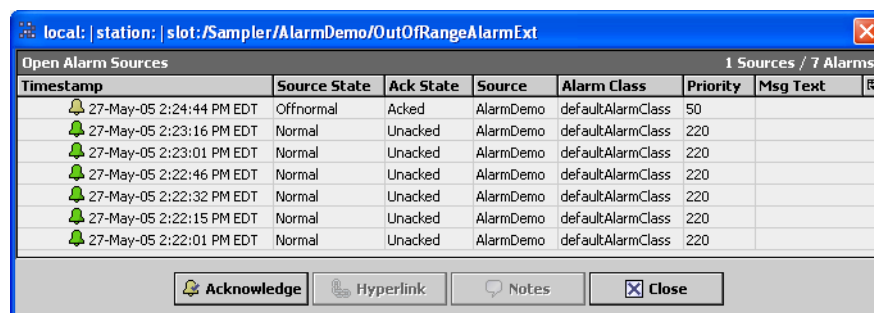
Figure 5-56 Alarm console



The alarm console manages alarms on a per-point basis. Each row in the alarm console is the most recent alarm from a point. To view all the current alarms or to get more details about a particular alarm from that point, use the alarm sources view and the associated dialog box displays. The export toolbar icon  is available on the toolbar when the alarm extension manager view is displayed.

To view all the current alarms from a particular point in the open alarm sources view, double click the desired row. The alarm details view appears, as shown in [Figure 5-57](#).

Figure 5-57 Open alarm sources view



To acknowledge an alarm, select the desired alarm and click the **Acknowledge** button. An alarm is cleared from the alarm console when both of the following conditions exist:


- alarm is acknowledged
- the point is in a “normal” state

As with other tables, you can show or hide columns using the **Table Options** menu in the top right corner of the table (refer to “[Table controls and options](#)” on page 2-18).

Hide, show, add, and remove alarm data columns in the alarm console

Default alarm data types appear in each of the columns of the alarm console view. You can change the visibility of alarm data columns in the following ways:

- Hide alarm data column**

To hide an alarm data column, click on the desired data column menu item in the Table Options Menu  (located in the top right corner of the table view). Selecting the menu item toggles the visibility, as indicated by the check mark to the left of data column name in the menu.



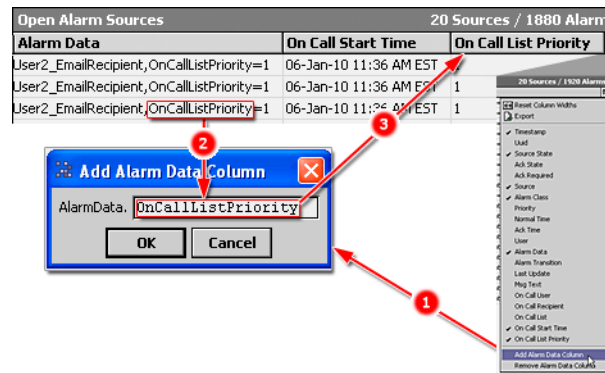
- **Show alarm data column**
To show an alarm data column, click on the desired data column menu item in the Table Options Menu  (located in the top right corner of the table view). Selecting the menu item toggles the visibility, as indicated by the check mark to the left of data column name in the menu.
- **Add alarm data column**
Add alarm data columns to the alarm console view using the **Add Alarm Data Column** dialog box. To open the Add Alarm Data Column dialog box, select the **Add Alarm Data Column** menu item in the **Table Options Menu**  (located in the top right corner of the table view). This opens the dialog box and allows you to enter the exact alarm data text string, as shown below.

Figure 5-58 Adding an alarm data column



Note: Alarm data string text displays in the “Alarm Data” column when that column is visible. When the exact case-sensitive string is entered in the **Add Alarm Data Column** dialog box, the new column appears in the console as a column that you can hide, show, or remove, as desired.


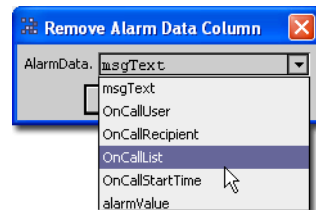
- **Remove alarm data column**
Remove alarm data columns from the alarm console view using the **Remove Alarm Data Column** dialog box. To open the Remove Alarm Data Column dialog box, select the **Remove Alarm Data Column** menu item in the **Table Options Menu**  (located in the top right corner of the table view). This opens the dialog box and allows you to select the alarm data column to remove, as shown below.

Figure 5-59 Removing an alarm data column



Note: Alarm data string text displays in the “Alarm Data” column when that column is visible. When the exact case-sensitive string is entered in the **Add Alarm Data Column** dialog box, the new column appears in the console as a column that you can hide, show, or remove, as desired.

Types of Alarm Details View dialog boxes

- **alarm record dialog box**
This dialog box shows additional detailed information about a specific point alarm, as shown in [Figure 5-60](#). It also allows you to:
 - acknowledge an alarm using the **Acknowledge** button.
 - jump directly to a point using the **Hyperlink** button.
 - add notes to the alarm record using the notes dialog box (use the **Notes** button).

Figure 5-60 Alarm record details dialog box

The dialog box displays the following information:

Timestamp	28-Aug-12 5:27:35 PM EDT																														
Uuid	032e4986-7b7d-4852-b736-10deabe26228																														
Source State	Offnormal																														
Ack State	Unacked																														
Ack Required	true																														
Source	Room_101 local: station: slot:/Logic/Room_101/OutOfRangeAlarmExt																														
Alarm Class	Default Alarm Class																														
Priority	255																														
Normal Time	null																														
Ack Time	null																														
User	Unknown User																														
Alarm Data	<table border="1"> <tr><td>Alarm Value</td><td>78.10</td></tr> <tr><td>Count</td><td>12</td></tr> <tr><td>Deadband</td><td>1.0 °F</td></tr> <tr><td>Escalated</td><td></td></tr> <tr><td>From State</td><td>normal</td></tr> <tr><td>High Limit</td><td>78.0 °F</td></tr> <tr><td>Hyperlink Ord</td><td>local: fox: station: slot:/Logic</td></tr> <tr><td>Low Limit</td><td>64.0 °F</td></tr> <tr><td>Message Text</td><td>Room_101 is at 78.1 °F which is above the high limit of 78.0 °F!</td></tr> <tr><td>Offnormal Value</td><td>78.1 °F</td></tr> <tr><td>Present Value</td><td>78.1 °F</td></tr> <tr><td>Source Name</td><td>Room 101</td></tr> <tr><td>Status</td><td>{ok} @ 16</td></tr> <tr><td>Time Zone</td><td>America/New York (-5/-4)</td></tr> <tr><td>To State</td><td>highLimit</td></tr> </table>	Alarm Value	78.10	Count	12	Deadband	1.0 °F	Escalated		From State	normal	High Limit	78.0 °F	Hyperlink Ord	local: fox: station: slot:/Logic	Low Limit	64.0 °F	Message Text	Room_101 is at 78.1 °F which is above the high limit of 78.0 °F!	Offnormal Value	78.1 °F	Present Value	78.1 °F	Source Name	Room 101	Status	{ok} @ 16	Time Zone	America/New York (-5/-4)	To State	highLimit
Alarm Value	78.10																														
Count	12																														
Deadband	1.0 °F																														
Escalated																															
From State	normal																														
High Limit	78.0 °F																														
Hyperlink Ord	local: fox: station: slot:/Logic																														
Low Limit	64.0 °F																														
Message Text	Room_101 is at 78.1 °F which is above the high limit of 78.0 °F!																														
Offnormal Value	78.1 °F																														
Present Value	78.1 °F																														
Source Name	Room 101																														
Status	{ok} @ 16																														
Time Zone	America/New York (-5/-4)																														
To State	highLimit																														
Alarm Transition	Offnormal																														
Last Update	28-Aug-12 5:27:35 PM EDT																														

Buttons at the bottom: Acknowledge, Hyperlink, Notes, Close, and navigation arrows.

Note: Starting in AX-3.7, if multiple records exist for an alarm, which can happen if repeated alarm transitions occur unacknowledged, arrow buttons may be enabled at the bottom corners:

- Go to previous alarm record (back in time)
- Go to next alarm record (forward in time)

Previously, to access details on a related alarm instance (previous or next), you had to close the alarm record popup dialog, and then select details on another record instance.

• Filters dialog box

The **Filters** dialog box displays parameters, as shown in [Figure 5-61](#), that you can use to include or exclude alarms from the alarm console by selecting or deselecting parameters. This filter action only affects which alarms display in the alarm console, it does not edit any alarm record data or perform any alarm “maintenance”.

Figure 5-61 Filters dialog box

The dialog box contains the following filter sections:

- Source State:** Normal
- Ack State:**
- Ack Required:** false
- Source:** %Temp% (Must Be Like, Case Sensitive)
- Alarm Class:** % (Must Be Like, Case Sensitive)
- Priority:** min 0 (Must Be Like, Case Sensitive)
- Normal Time:** Time Range (min 31-Dec-1969 07:00 PM EST, max 31-Dec-1969 07:00 PM EST)
- Ack Time:** Time Range (min 31-Dec-1969 07:00 PM EST, max 31-Dec-1969 07:00 PM EST)
- User:** % (Must Be Like, Case Sensitive)
- Alarm Data:** No facets
- Alarm Transition:**
- Last Update:** Time Range (min 31-Dec-1969 07:00 PM EST, max 31-Dec-1969 07:00 PM EST)

Buttons at the bottom: OK, Cancel

Note: Starting in AX-3.7, the **Filters** dialog's field editor changed for parameters "**Source**" (shown enabled in Figure 5-61) as well as "**Alarm Class**" and "**User**". These filter parameters now have an editor that when enabled (checked) include:

- Text string field, with wildcard "%" character by default. Enter text to filter on here.
- Drop-down operator, as either "Must Be Like" (default), "Must Not Be Like", or else "Must Equal" or "Must Not Equal". Note the last two (equal) ones require that the text string does *not* contain any "%" wildcard character.
- Checkbox for case-sensitive (set by default).

*Note if filtering on **Source** and using a wildcard, make sure to put the "%" character at both ends of the text string, as shown in Figure 5-61. Otherwise, filter matches cannot occur.*

You can choose, for example, to filter out any alarms in the alarm console that are currently in a "Normal" state by selecting the "Source State" check box and then selecting all states except "Normal" and clicking the **OK** button. This action filters out all alarm records that have "Normal" current Source States. If the source state changes or if you change the settings in the Filters dialog box, the alarm console table will update to change the display, as indicated.

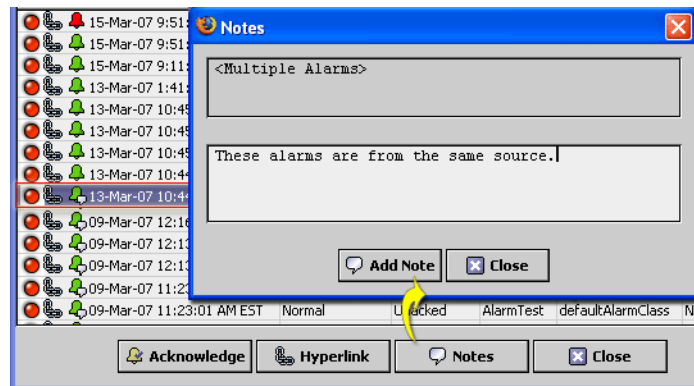


Caution It is important to remember that filter settings do not reset automatically—you must remove any filters that you set in order to view all alarm records. Note that the **Filter** button displays the "Filter" text in red to indicate that alarms are being filtered.

• Notes dialog box

Use the **Notes** dialog box to add a note to one or more alarms. To add a note to all the alarms from a selected source, open the **Notes** dialog box directly from the alarm console view, using the **Notes** button. If the selected alarm record represents a source with multiple alarms, any note that you add is added to all the alarms associated with that alarm source. When there is more than one alarm associated with an alarm record, the **Notes** dialog box displays a <Multiple Alarms> message, as shown in Figure 5-62.

Figure 5-62 Adding a single Note to more than alarm

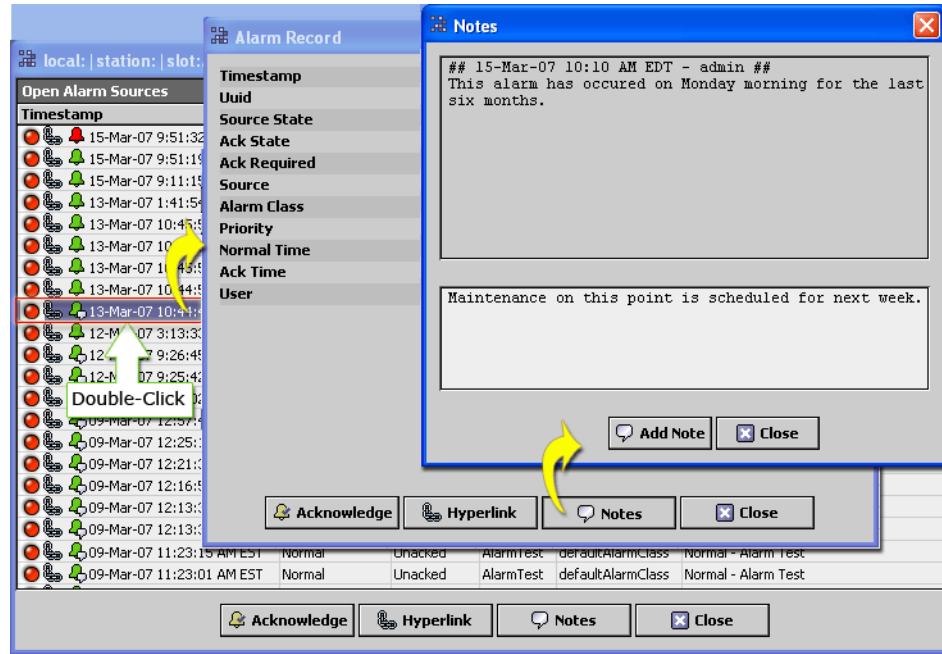


The **Notes** dialog box is comprised of the following:

- **Message pane**
The message pane is located in the upper half of the dialog box. It displays the text of any notes that are already associated with the selected alarm record. If multiple alarms are associated with the selected alarm record, the message pane also displays a "<Multiple Alarms>" notice to alert you to the fact that adding a note adds the note to all the alarms associated with the selected alarm source.
- **Editor pane**
The Editor pane is located in the lower half of the dialog box and is a text field that allows you to type the text for any note that you are adding.
- **Add Note** button
This button saves the note to the alarm record and dismisses the Notes dialog box.
- **Close** button
This button closes the **Notes** dialog box without saving any information.

You can also open the Notes dialog box from the **Alarm Record** dialog box. Since the **Alarm Record** dialog box displays single alarm records, notes are added to only one alarm at a time using this method.

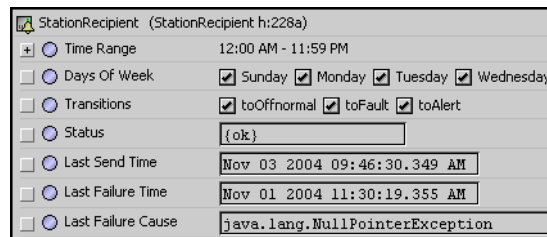
Figure 5-63 Opening the Notes dialog box from the Alarm Record dialog box



About the station recipient

The station recipient manages the transfer of alarms between the alarm service and a remote station. For example, a station may send alarm notifications to a “supervisor” station – or any other remote station that is defined under the NiagaraNetwork. The station recipient component provides a place to specify the location and other details about that remote station. Station recipient properties are shown in Figure 5-64. The properties on a station recipient include a field for selecting the remote station, as well as alarm collection parameters.

Figure 5-64 Station recipient property sheet



Station recipient properties are described in the following list:

- **Time Range**
allows you to set a limited period of time during a day for collection of alarms, using the following parameters:
 - Start Time
a time of day to begin alarm collection, defined by hour, minute and second.
 - End Time
a time of day to end alarm collection, defined by hour, minute and second.
- **Days Of Week**
option boxes allow selection of specific days to collect alarms.
- **Transitions**
option boxes allow selection of specific alarm transitions to display. Only those transitions that are selected will be displayed – even though the alarms are still saved into the alarm history.
- **Route Acks**
when this parameter is set to true, Acks are routed to this recipient; when set to false, Acks are not routed to the recipient.

- **Remote Station**
option selection displays eligible remote stations. Stations appear in the option list if there is a valid network connection to the station. Alarms that are sent to a remote station are routed according to the parameters defined in the Alarm class that is selected in the Alarm component under the remote station's NiagaraNetwork. Refer to the *Drivers Guide* section “NiagaraStation Alarms notes” for more information about remote station alarm routing.

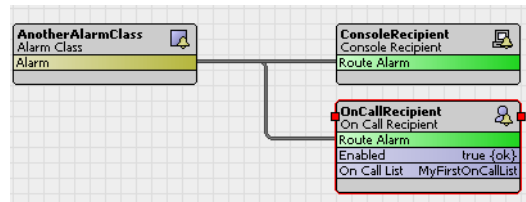
About the On Call Recipient

The On Call Recipient (AX-3.5 and later) manages the transfer of alarms between the alarm service and On Call Contacts. For example, a station may send alarm notifications by email or text message to one or more contacts if they are specified in the currently assigned On Call Contact List.

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

The On Call Recipient component is linked to an Alarm Class component and provides a place to specify the scheduling details and other routing options.

Figure 5-65 On Call Recipient linked to an Alarm Class component



On Call Recipient properties are shown in Figure 5-66 and described in the following list:

Figure 5-66 On Call Recipient property sheet

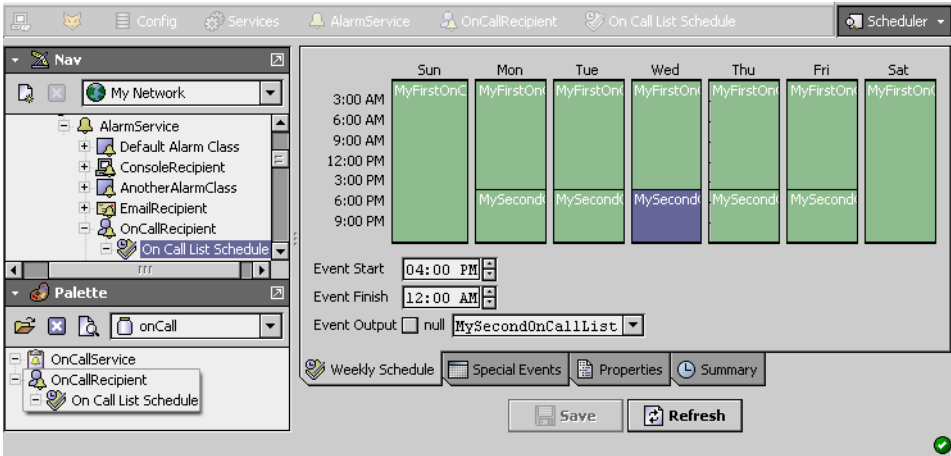
The screenshot shows the 'OnCallRecipient' (On Call Recipient) property sheet. The 'Time Range' is set to '12:00 AM - 12:00 AM'. The 'Days Of Week' are set to 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', and 'Sat'. The 'Transitions' are set to 'toOffnormal', 'toFault', 'toNormal', and 'toAlert'. The 'Route Acks' are set to 'true'. The 'Enabled' property is set to 'true {ok}'. The 'On Call List' is set to 'MyFirstOnCallList'. The 'On Call List Schedule' is set to 'MyFirstOnCallList {ok}'. The 'Escalation Delay' is set to '+00004h 30m 00s'.

- **Time Range**
This property allows you to set a limited period of time during a day for collection of alarms, using the following parameters:
 - **Start Time**
a time of day to begin alarm collection, defined by hour, minute and second.
 - **End Time**
a time of day to end alarm collection, defined by hour, minute and second.
- **Days Of Week**
These option boxes allow selection of specific days to collect alarms.
- **Transitions**
These option boxes allow selection of specific alarm transitions to display. Only those transitions that are selected will be displayed – even though the alarms are still saved into the alarm history.
- **Route Acks**
When this parameter is set to `true`, Acks are routed to this recipient; when set to `false`, Acks are not routed to the recipient.
- **Enabled**
When this parameter is set to `true`, the OnCallRecipient component is operational.
- **On Call List**
This field displays the currently active On Call List and its status.
- **On Call List Schedule**
This is a special On Call scheduling component that is contained in the On Call Recipient component. The On Call List Schedule provides a standard scheduling view that is similar to other schedule

views. The unique feature of this scheduling view is that the Event Output property value is assigned either a null value or an On Call List value. If a null value is selected, no alarms are forwarded. If an On Call List is assigned, then alarms are forwarded to contacts, as specified in the On Call List.

- **Escalation Delay**
This property allows you to set the amount of time to wait for an alarm to be acknowledged by an On Call Contact before escalating it to the next contact in the On Call List. The default delay time is 30 minutes.

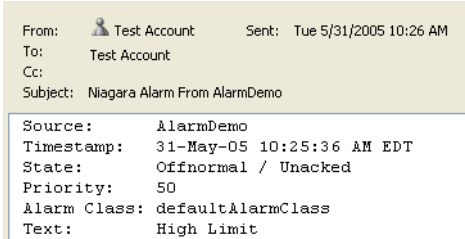
Figure 5-67 On Call List Scheduler view



About the Email Recipient (email-EmailRecipient)

The Email Recipient is like other alarm recipients except that the alarm may be formatted into an email message and delivered to another destination. An example of the email message format is shown in [Figure 5-68](#).

Figure 5-68 Email alarm message



The properties on an email message recipient (shown in [Figure 5-69](#)) include the data for the email message fields as well as alarm collection parameters.

Figure 5-69 Email recipient property sheet

Figure 5-70 Configuring the EmailRecipient for the AlarmAcknowledger

About the Sms recipient

The SmsRecipient component (AX-3.5 and later) acts as an alarm recipient for transmitting alarm notifications by text message. Its use is similar to the EmailRecipient component. You can specify the text for the SMS message within the body field of the property sheet view.

The properties on an Sms recipient (shown in [Figure 5-71](#)) include the data for targeted cell phone numbers, the Sms message source and text fields, and alarm collection scheduling parameters.

For including the alarm UUID, include the string: UUID: %uuid% in the Body property text field, as shown below.

Figure 5-71 Sms recipient property sheet

Note: There is a limit of 140 characters for an SMS message. Exceeding this limit causes the body text to be truncated.

For related information, refer to the *Sms Users Guide*.

About the lineprinter recipient

The **LinePrinterRecipient** allows you to print alarms to *line printers*. The printer must be “known” to the Windows OS of the host platform, and selected from the printers drop down-list.

- In AX-3.7 and later, a **PrinterRecipient** component is also available. It provides more formatting options, applicable to most modern printers. See “[About the printer recipient](#)” on page 5-46.
- The station must have permission to print on any printer chosen (which is typical).

Alerts may be generated if the printing of an alarm fails, but the lineprinter recipient does not print alarms that it generates itself. An example lineprinter recipient property sheet is shown in [Figure 5-72](#).

Figure 5-72 Lineprinter recipient properties

LinePrinterRecipient (Line Printer Recipient)

☒ Time Range 12:00 AM - 12:00 AM

☐ Start Time 12:00:00 AM EDT

☐ End Time 12:00:00 AM EDT

☐ Days Of Week ☒ Sun ☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat

☐ Transitions ☒ toOffnormal ☒ toFault ☒ toNormal ☒ toAlert

☐ Route Acks true

☐ Printer Epson LQ-1170 Scalable Font

☐ Language

☐ Print Format

Source: \$alarmData.sourceName%
Timestamp: \$timestamp%
State: \$sourceState% / \$ackState%
Priority: \$priority%
Alarm Class: \$alarmClass%
Text: \$alarmData.msgText%


☐ Alert On Failure true

☒ Alarm Source Info Alarm Source Info

Lineprinter recipient properties are described in the following list:

- **Time Range**
allows you to set a limited period of time during a day for collection of alarms, using the following parameters:
 - Start Time
a time of day to begin alarm collection, defined by hour, minute and second.
 - End Time
a time of day to end alarm collection, defined by hour, minute and second.
- **Days Of Week**
option boxes allow selection of specific days to collect alarms.
- **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**
when this parameter is set to `true`, Acks are routed to this recipient; when set to `false`, only alarms (not Acks) are routed to the recipient.
- **Printer**
a drop-down list shows printers available for selection. Select a printer from the list, which reflects printers (both locally attached and remote networked) that are “known” to the host platform’s Windows operating system. These are the only valid printer values (example in [Figure 5-73](#) on page 5-46).
- **Language**
this field provides a place to enter the ISO 639 language code for the language associated with the printer. This is a two letter code (lower-case preferred). Refer to the following link for the complete list of codes: http://www.loc.gov/standards/iso639-2/php/code_list.php.
- **Print Format**
This field has the following editable default parameters.
 - Source: %alarmData.sourceName%

- this parameter sends the source name of the alarm to print on the first line.
- Timestamp: %timestamp%
this parameter sends the timestamp of the alarm to print on the second line
- State: %sourceState% / %ackState%
this parameter sends the alarm state and the acknowledged state to print on the third line
- Priority: %priority%
this parameter sends the alarm priority to print on the fourth line
- Alarm Class: %alarmClass%
- Text: %alarmData.msgText%

Note: For more information about how to format data in this field, click on the help icon  to the right of the field.

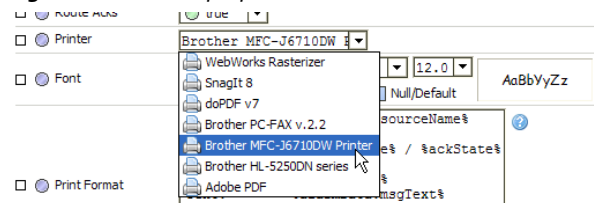
- **Alert on Failure**
when this parameter is set to `true`, an alert is generated if the printer fails to print the alarm.
- **Alarm Source Info**
contains the usual set of properties for configuring and routing alarms sourced from this component—in this case “alerts” for any failed print attempt (provided that “Alert on Failure” is true).

About the printer recipient

Starting in AX-3.7, a PrinterRecipient is available in the **Recipients** folder of the `alarm` module palette. It is similar to the prior `LinePrinterRecipient`, but is more appropriate for typical modern printers (such as laser printers) that print out each alarm on a separate page.

Like the `LinePrinterRecipient`, it applies to *Windows* hosted stations only. The printer must be “known” to the Windows OS of the host platform, selected from the printer drop down-list (Figure 5-73).

Figure 5-73 Example printers known to Windows OS of hosted station, in PrinterRecipient property sheet

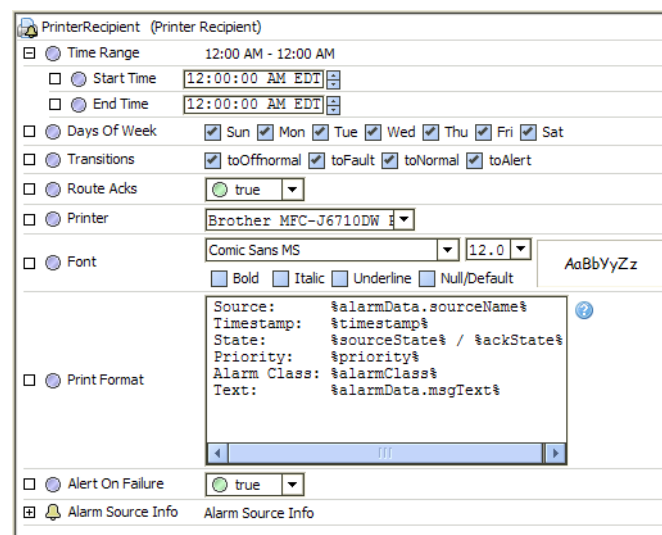


Note: The station must have permission to print on any printer chosen (which is typical).

The main differences from the `LinePrinterRecipient` are the additional “Font” property settings, which allow selection of font type, size, and various style overrides. Combined with the “multi-line” capable alarm message “text” properties in various alarm extensions (starting in AX-3.7), this provides more flexibility for alarm printing (see “About alarm extension properties” on page 5-3).

An example of the printer recipient property sheet is shown in Figure 5-72.


Figure 5-74 Example printer recipient properties



Printer recipient properties are described in the following list:

- **Time Range**
allows you to set a limited period of time during a day for printing, using the following parameters:
 - Start Time
a time of day to begin alarm collection, defined by hour, minute and second.
 - End Time
a time of day to end alarm collection, defined by hour, minute and second.
- **Days Of Week**
option boxes allow selection of specific days to print.
- **Transitions**
option boxes allow selection of specific alarm transitions to print. Only those transitions that are selected will be printed – even though the alarms are still saved into the alarm history.
- **Route Acks**
when this parameter is set to `true`, Acks are routed to this recipient; when set to `false`, only alarms (not Acks) are routed to the recipient.
- **Printer**
a drop-down list shows printers available for selection. Select a printer from the list, which reflects printers (both locally attached and remote networked) that are “known” to the host platform’s Windows operating system. These are the only valid printer values (see [Figure 5-73](#) on page 5-46).
- **Font**
Provides a number of controls to select font formatting for printed alarms, including:
 - (Type) — Lists the font type (default “Courier New”) with a drop-down ▾ control for selection.
 - (Size) — Point size of font (default 12.0) with a drop-down ▾ control for selection.
 - Style(s) — Checkboxes for Bold, Italic, and Underline, selectable separately or in combination, or else “Null/Default”. If null/default is chosen, this clears all other font settings, and specifies to use the printer’s native default font.

In the property sheet, a small area to the right of the Font settings displays a “preview” of selected font settings using a text string: AaBbYyZz.
- **Language**
this field provides a place to enter the ISO 639 language code for the language associated with the printer. This is a two letter code (lower-case preferred). Refer to the following link for the complete list of codes: http://www.loc.gov/standards/iso639-2/php/code_list.php.
- **Print Format**
This field has the following editable default parameters.
 - Source: %alarmData.sourceName%
this parameter sends the source name of the alarm to print on the first line.
 - Timestamp: %timestamp%
this parameter sends the timestamp of the alarm to print on the second line
 - State: %sourceState% / %ackState%
this parameter sends the alarm state and the acknowledged state to print on the third line
 - Priority: %priority%
this parameter sends the alarm priority to print on the fourth line
 - Alarm Class: %alarmClass%
 - Text: %alarmData.msgText%

Note: For more information about how to format data in this field, click on the help icon  to the right of the field.
- **Alert on Failure**
when this parameter is set to `true`, an alert is generated if the printer fails to print the alarm.
- **Alarm Source Info**
contains the usual set of properties for configuring and routing alarms sourced from this component—in this case “alerts” for any failed print attempt (provided that “Alert on Failure” is true).

CHAPTER 6

About Security

This section explains security for any NiagaraAX *station*. Station security means access by a person using either Workbench or a web browser, or station access by another station. Security determines if a station connection is allowed, what can be accessed, and what operations can be performed. In the case of individual users, it also typically determines navigation options and overall appearance.

Note: *Niagara platform security is a different topic altogether. For more details, see the sections “About a platform connection” and “Update Authentication” in the Platform Guide.*

A station’s security is based upon *users* and object *categories*, with each user’s *permissions* within any category defined with specific permission levels. All configuration is stored in the station database, using services, components, and component views.

A station’s security also depends upon the *authentication* used, as well as *password strength*.

The following sections provide more station security details:

- [“Security changes and notes”](#) on page 6-2 (Revised in this AX-3.8 update)
- [“Security model overview”](#) on page 6-2
 - [“Categories and security”](#) on page 6-3
 - [“Users and security”](#) on page 6-4
 - [“About built-in users”](#) on page 6-6
 - [“About authentication”](#) on page 6-6
 - [“Network users”](#) on page 6-7
 - [“Permissions and security”](#) on page 6-8
 - [“About permission levels”](#) on page 6-10
 - [“UserService security notes”](#) on page 6-11
 - [“About user audits”](#) on page 6-12
 - [“Multi-station security notes”](#) on page 6-15
 - [“Alarm ack permission notes”](#) on page 6-18
 - [“About password expiration and reset”](#) on page 6-18 (AX-3.7 and later)
 - [“Password history \(unique passwords\)”](#) on page 6-23 (AX-3.7 and later)
- [“UserService”](#) on page 6-24
 - [“UserService properties”](#) on page 6-24
 - [“Strong password notes”](#) on page 6-25
 - [“Password Strength \(AX-3.8\)”](#) on page 6-25
 - [“Require Strong Passwords \(AX-3.7\)”](#) on page 6-27
 - [“Stronger passwords”](#) on page 6-28
 - [“User Manager”](#) on page 6-28
 - [“Lockout notes”](#) on page 6-29
 - [“Network user notes”](#) on page 6-29
 - [“User”](#) on page 6-30
 - [“Network user related properties”](#) on page 6-34
 - [“Permissions Browser”](#) on page 6-34
 - [“Show Configured”](#) on page 6-34
 - [“Permissions abbreviations”](#) on page 6-35
 - [“Access permissions”](#) on page 6-35
 - [“About User prototypes”](#) on page 6-36
 - [“Properties of User Prototypes”](#) on page 6-36
 - [“Default Prototype”](#) on page 6-37
 - [“Additional \(non-default\) User Prototypes”](#) on page 6-37
- [“CategoryService”](#) on page 6-38

- “Category Manager” on page 6-39
- “Category Browser” on page 6-40
- “Category Sheet” on page 6-41

Security changes and notes

In AX-3.8, station security continues with improvements over prior releases, including the “update 1” release for AX-3.7 (in this document “AX-3.7u1”).

- In AX-3.8, the definition of “strong passwords” is now configurable for any station. A new “Password Strength” container under the station’s UserService has separate properties to specify the minimum numbers of characters in a valid password, including total number (length), upper case, lower case, digits, and special characters. This container effectively replaces the previous “Require Strong Password” (boolean property) of the UserService, where minimum requirements for strong passwords was fixed—and not especially strong. For details, see “[Strong password notes](#)” on page 6-25.
- AX-3.8 uses the same improved station password storage first introduced in AX-3.7u1 (and also “update 4” releases for AX-3.6 and AX-3.5). However, AX-3.8 Workbench platform tools to copy/transfer stations and make backups were improved to make station archives more “portable” among multiple hosts. An update to the standalone *NiagaraAX 2013 Security Updates* document describes these AX-3.8 changes, along with continuing considerations that apply when upgrading a system running a release prior to AX-3.7u1 (or AX-3.6u4 or AX-3.5u4) to AX-3.8.
- In AX-3.8, required user permissions to acknowledge alarms were *reduced*. For details, see “[Alarm ack permission notes](#)” on page 6-18.

Station security features introduced in the original AX-3.7 release continue in the AX-3.8 release. You can require users to reset their password upon next logon, define periodic password expirations (prompting users to change their passwords), and specify a “password history” limit to prevent reuse of old passwords. By default, “strong passwords” in a UserService are in effect, with default minimum values of password length 10, upper case 1, lower case 1, digits (numerals) 1, and special characters 0 (none). The Workbench “New Station” wizard uses these particular defaults; however, once a station is created you can change its definition of strong passwords if needed.

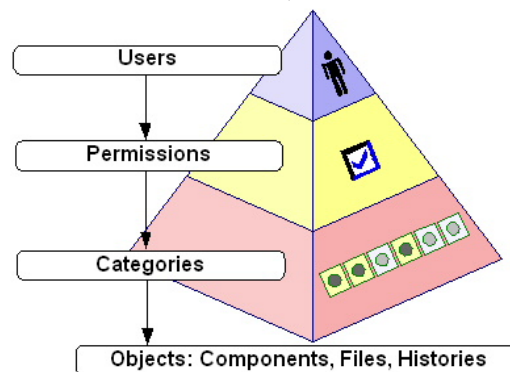
Finally, as in AX-3.7, most NiagaraAX hosts can be configured for secure, encrypted (SSL or TLS) connections for all types of Niagara access, including station (Fox) connections, browser access, or platform connections. Note this PKI certificate-based server authentication occurs *before* station login, and is SSL encrypted. In an SSL connection attempt, if a user “rejects” an untrusted or unknown certificate at the warning, the Niagara login dialog is never reached. For complete details, refer to the *NiagaraAX SSL Connectivity Guide*.

Security model overview

Along with [authentication](#) methods and password strengths, a station’s security is determined by the combination of [categories](#), [users](#), and [permissions](#), as shown in shown in [Figure 6-1](#).

Note: *In general, you create needed categories first, then assign objects to them. Then create users, to which you assign permissions. However, you can engineer security in any order you want.*

Figure 6-1 Station security includes categories, users, permissions



The following sections explain the various aspects of station security:

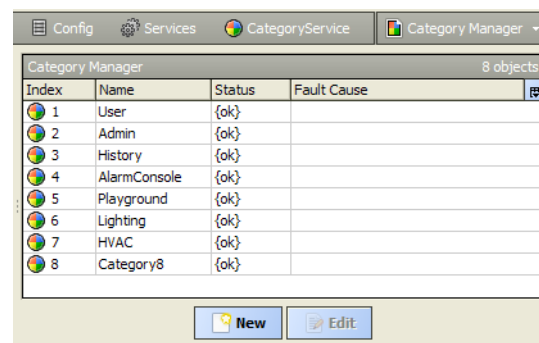
- “[Categories and security](#)” on page 6-3

- “Users and security” on page 6-4
- “About built-in users” on page 6-6
- “About authentication” on page 6-6
- “Network users” on page 6-7
- “Permissions and security” on page 6-8
- “About permission levels” on page 6-10
- “UserService security notes” on page 6-11
- “About user audits” on page 6-12
- “Multi-station security notes” on page 6-15
- “Alarm ack permission notes” on page 6-18
- “About password expiration and reset” on page 6-18 (AX-3.7 and later)
- “Password history (unique passwords)” on page 6-23 (AX-3.7 and later)

Categories and security

Categories are integral to station security. A station’s [CategoryService](#) (under Services container) provides a [Category Manager](#) view for you to add, delete, and edit categories, as shown in [Figure 6-2](#).

Figure 6-2 CategoryService and Category Manager (secondary) view



Each [Category](#) is really just an index (Category 1, Category 2, and so on), however, you typically *name* them to reflect some logical grouping. For example, you could have “Lighting” and “Floor 3” categories, and assign some objects to both, and others to just one category (or neither of these).

Note: By default, starting in AX-3.7, a new station (created using the New Station wizard) has two default categories: “User” (category 1) and “Admin” (category 2). Formerly, no default categories were created. However (as before) category slots (1—8) still exist.

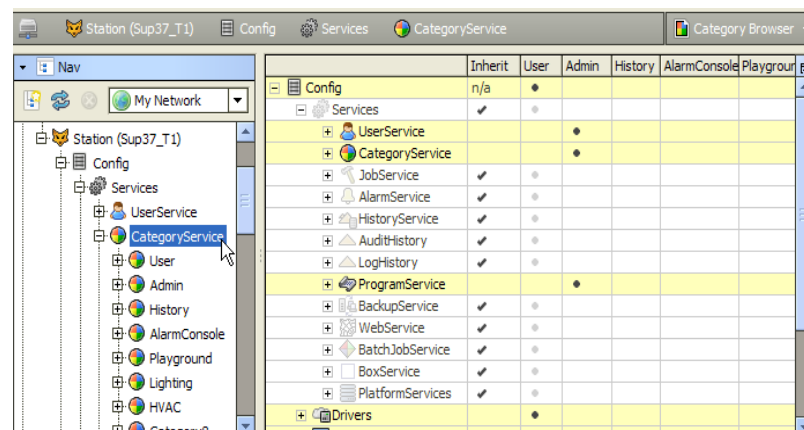
Before, all objects defaulted to category 1. However, starting in AX-3.7, to encourage stronger security, selected objects now default to category 2 (Admin), including the following:

- Config: Services: UserService, CategoryService, and ProgramService
- Files: (all, meaning entire file space)

You can simply add categories using the Category Manager, shown in [Figure 6-2](#), in order to rename indexes in a more meaningful way.

As shown in [Figure 6-3](#), you typically use the [Category Browser](#) (default view of the CategoryService) to assign objects in the station to different categories. Every object must be assigned to at least one category.

Figure 6-3 CategoryBrowser (CategoryService default view)



In the CategoryBrowser, you use an expandable tree to navigate to objects of interest, and click in table cells to add or remove category assignments. As an alternative to explicitly assigning one or more categories to an object, you can have it “inherit” the assigned categories of its parent. This affects appearance in the CategoryBrowser, where:

- Yellow rows are objects explicitly assigned into one or more categories.
- Dimmed rows represent objects *inheriting* their parent’s category or categories.

Note: You can also review and assign components to categories individually at the component level, using any component’s CategorySheet view. Included is a convenience link back to the CategoryBrowser (Category-Service). See “Category Sheet” on page 6-41.

By themselves, categories have little security meaning. Only when you set a user’s [permissions](#) do you reference categories. For more details on categories, see “CategoryService” on page 6-38 and “Category caveats” on page 6-40.

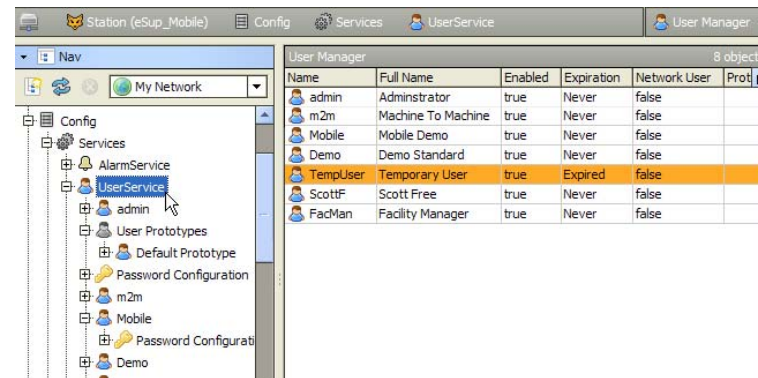
Users and security

Note: This section describes Users under the `baja` **UserService**, the standard service container for station users. By default, the UserService is included in the **Services** container of any new station created using the **New Station** tool in Workbench. Note that some installations may use LDAP or Active Directory integration, where another user service from the `ldap` module replaces the standard UserService.

In those stations, creation of “local” station users is effectively the same; however, LDAP users are more typically used. For related details, refer to the NiagaraAX LDAP / Active Directory Configuration Guide.

Users define possible connections to the station. Under the station’s Service container, a UserService provides a default User Manager view (Figure 6-4) for you to add, delete, and edit users.

Figure 6-4 UserService and default UserManager view



Name	Full Name	Enabled	Expiration	Network User	Prot
admin	Administrator	true	Never	false	
m2m	Machine To Machine	true	Never	false	
Mobile	Mobile Demo	true	Never	false	
Demo	Demo Standard	true	Never	false	
TempUser	Temporary User	true	Expired	false	
ScottF	Scott Free	true	Never	false	
FacMan	Facility Manager	true	Never	false	

Typically, each user represents a specific *person* who uses Workbench or a web browser (or both) to connect to the station. Properties for any user include various settings, including several affecting mostly browser access, including Facets, Nav File, Web Profile, (and in AX-3.7) Mobile Profile, as shown in Figure 6-5 on page 6-5.

However, a user can be for one or more remote NiagaraAX stations to use (for client connection), in order to share data and import/export histories and schedules. If a “station-type” user (“service account”), in any remote NiagaraAX station, you reference this user when adding that NiagaraStation device (under its Fox Client Connection properties). For related details, see “Multi-station security notes” on page 6-15.

Figure 6-5 Example User properties (Edit dialog from User Manager)

Name	Full Name	Enabled	Expiration	Permissions	Network User	Prototype Name	Language	Password	Email
AnitaL	Anita Life	true	Never	1=rwIR;2=rwl	false			--password--	alife@newmetro

Name: AnitaL

Full Name: Anita Life

Enabled: ☒ true

Expiration: ☒ Never Expires ☐ Expires On 05-Aug-2012 11:59 PM EDT

Permissions: ☐ Super User (access entire station, file system) 1=rwIR;2=rwl;3=rwlR;4=rwlRN;6=rwlRN

Network User: ☒ false

Prototype Name:

Language:

Password: Password: Confirm:

Email: alife@newmetropolis.net

Cell Phone Number: 8005551236

Facets: Time Format: (default) Unit Conversion: None

Nav File: files:~/nav/Home.nav

Default Web Profile: Auto Logoff Enabled: ☒ true Auto Logoff Period: 00000h 15m [5mins - +inf] Type: Default Wb Web Profile

Applet Reload On Hyperlink: ☒ true Workbench Theme: Lucid Show Back And Forwards Buttons: ☒ false

Force Password Reset: ☒ false

Password Expiration: ☒ Never Expires ☐ Expires On 05-Aug-2012 11:59 PM EDT Auto Logoff Enabled: ☒ true Auto Logoff Period: 00000h 15m [5mins - +inf] Mobile Nav File: null Type: Default Mobile Web Profile

Mobile Web Profile: Theme: mobile DefaultQueryMobileTheme: Show Header: ☒ Show Show Header Back Button: ☒ Show Show Select Views: ☒ Show Show Home: ☒ Show

Default Web Profile properties for Applet Reload, Theme, and Show Back and Forward Buttons are available only if Type is a "Wb Web" selection.

Force Password Reset and Password Expiration are properties in the PasswordConfiguration container for each User (new starting in 3.7).

Mobile Web Profile properties appear only if a 3.7 or later host licensed with the "mobile" feature.

Configure station users in the User Manager by using the **New** button to create users, or else click (select) existing users and then click the **Edit** button (or simply double-click existing users). Figure 6-5 shows an example **Edit** dialog for an existing station user.

Two important security aspects are "built-in users," and the authentication method used by a station.

- [About built-in users](#)
- [About authentication](#)

For further details on the UserService, including descriptions of service properties and views, as well as properties of child (User) components, see "[UserService](#)" on page 6-24 and "[User](#)" on page 6-30.

Note: The concept of "network users" is also important for any multi-station job. For an overview, see "[Network users](#)" on page 6-7.

About built-in users

Any station has two *built-in users* that you can neither delete nor rename: [admin](#) and [guest](#).

User admin User “admin” is unique from all other users in that you cannot set it to expire, or assign permissions (it is always “super user”, meaning *all* permissions). This ensures every station always has at least one super user.



Caution

Prior to AX-3.7, you could not disable the admin user—however, this changed such that “commonly known accounts” could not be a security issue. In AX-3.7 and later you can disable the station’s admin user. However, be careful before doing this—first, ensure that you have at least one other configured super user in the station.

Because super user accounts are so powerful, only assign super user permissions when absolutely necessary!

When using the **New Station** wizard tool to create a station, the wizard prompts for the password for the admin user. Note a new station is created enabled for “strong passwords”, so you must enter one to proceed. Requiring strong passwords is the recommended configuration for all stations. For related details, see “[New Station wizard](#)” on page 8-10.

Note: *Any host upgraded to AX-3.7 or later from older builds may have changed behavior regarding passwords. The station’s UserService now defaults to “strong passwords”, which will initially affect any new users created, as well as all subsequent password change for existing users. For related details, see “[UserService properties](#)” on page 6-24 and “[Strong password notes](#)” on page 6-25.*

Finally, password storage in NiagaraAX is much more secure starting in AX-3.7u1. For related details, see the NiagaraAX 2013 Security Updates engineering notes document.

In summary (unless you disable the admin user), guard its password from almost everyone, as well as use it infrequently. Each person should have their own (unique) user account, which also makes audit information more valuable. See “[About user audits](#)” on page 6-12 for more details.

User guest User “guest”, if enabled, provides station access from a browser with *no login required* (no [authentication](#)).

Note: *By default in AX-3.8, user “guest” is not only disabled, but its slot in the user service is automatically hidden upon the first startup of the station. This is an extra precaution to help prevent inappropriate usage. Only hosts licensed as “demo hosts” in AX-3.8 can enable and use the guest user—meaning it is not available on any host with a “non-expiring” license.*



Caution

By default, the guest user is disabled (and it is recommended to leave disabled). While guest is disabled, all station access requires [authentication](#), meaning a browser user is always prompted for user name and password. In this case, station login as “guest” is not available (using either a browser or Workbench).

If user “guest” is enabled, any browser pointed to the host (<http://<hostIpAddress>>) sees “home” in the Nav file assigned to guest. The browser operator can then navigate in the normal fashion, providing that guest has read permissions for objects. Access of an object lacking guest permissions produces the login prompt, whereby the operator must login (as a non-guest user) to proceed.

About authentication

For a station connection attempt, the user’s login *credentials* (user name, password) are checked against those users under the station’s UserService. This process is called *authentication*. The actual process (and authentication *mechanism* used) can differ, where the three authentication points are:

- **Workbench-to-station (FoxService)**

Typically, the Workbench user is prompted for user name and password, such as whenever opening a station from the File menu in Workbench. Fox authentication is used, which is configurable via the FoxService (under the NiagaraNetwork) as either Basic or Digest. Digest (the default) is the typical preferred mechanism, since the password is never passed in clear text. However, if using LDAP, then Basic authentication must be configured.

Note: Starting in AX-3.7u1, “Digest MD5” authentication is no longer available for Fox clients—instead Digest (SCRAM-SHA256) or Basic are the two choices. During a multi-station system upgrade, it may be necessary to (temporarily) set a Supervisor station’s FoxService property “Legacy Authentication” from the default “Strict” to one of the “Relaxed” settings. For related details, see “Update considerations” in the NiagaraAX 2013 Security Updates document.

It is also possible for a Workbench user to open one station from another station without being re-prompted for credentials. Such stations must be configured to use the same “realm,” and the user must also exist (with the identical credentials) in each station. See “Workbench single sign-on (SSO) realm” on page 6-17 for further details.

- **HTTP browser-to-station (WebService)**

Again, the user is prompted for user name and password. The authentication mechanism used is specified under the station’s WebService, as either Cookie Digest (default starting in AX-3.7), Basic, or as Cookie. Cookie Digest is the most secure, and replaces the previous default of Cookie.

Note: A station upgraded to AX-3.7 from an earlier release will have its WebService’s “Authentication Scheme” property automatically changed to **Cookie Digest**. In cases mentioned below, you may need to change this to one of the other options, in order to maintain the required behavior.

- If the station uses the **LdapUserService** or **ActiveDirectoryUserService** (replacing the **UserService**). In this case, set the WebService’s Authentication Scheme to **Cookie**.
- If the **Domain-wide cookies authentication** feature is being used. In this case, you will likely need to set the WebService’s Authentication Scheme to **Cookie**.

- **Station-to-station (FoxService)**

A preconfigured user name and password is used (stored under NiagaraStation.clientConnection), which must correspond to an existing user—typically, with admin write permissions. Again, Fox authentication is used, the same as with a Workbench to station connection.

Note: Although the password must match that for the user in the target (server) station, it is stored in the client station using a different encryption method—and one changed starting in AX-3.7u1. Thus, station-to-station communications can be affected during an upgrade. For related details, see “Update considerations” in the NiagaraAX 2013 Security Updates document.

In general, unless directed otherwise or noted above, it is recommended you leave authentication mechanisms (FoxService and WebService) at default settings. Default settings are also typically recommended unless the station is configured for LDAP (LdapUserService).

Network users

The **UserService** (baja module) and **NiagaraNetwork** (niagaraDriver module) permit “centralized management” of users in a multi-station system. This section provides an overview, summarizing the [Challenge](#), [User Service changes](#), and [NiagaraNetwork changes](#).

Note: The “network user” feature is available between stations that all use the standard **UserService**. This feature is not supported between any stations that use one of the LDAP user services instead (from **ldap** module). In those scenarios, centralized user management depends on the LDAP or Active Directory server. For related details, refer to the NiagaraAX LDAP / Active Directory Configuration Guide.

Challenge

In any NiagaraAX station database, station users are represented as individual **User** components, located under the station’s user service. Typically, you use the **User Manager** view of this service to add, modify, and delete users. Until recent station security changes, you were able to manually copy user components from one station to other stations. However, this method *no longer works*, due to more secure password storage. It also never provided change coordination. If an edit is needed for such a user, the same change had to be made to that user in each (separate) station. This was an inefficient process.

Solution: Stations are configurable to allow users to be added, modified, or deleted in one station, and then have those changes automatically replicated (or “synchronized”) in other stations. The term “network user” applies to these users. Related are configurable user “prototypes.” When adding users, prototypes can be used in network user “strategies” between station. Note these station user changes are standard, but optional—accomplished with additional slots on existing components.

To provide this functionality, there are areas of configuration changes in the station:

- [User Service changes](#)
- [NiagaraNetwork changes](#)

User Service changes

Every User component (user) in the station has 2 related configuration properties: Network User (boolean) and Prototype Name (string). Currently, prototype name matters only if the user is network user, as this can be used in a “sync strategy” for distributing changes to network users.

Related to this, the UserService has a frozen child container slot **User Prototypes**, with a frozen child “Default Prototype” user component. If establishing network users, you can duplicate and edit additional user prototypes. User prototypes currently have the same properties as users, and are seen in the Nav tree and in the property sheet of the UserService—but are *not* listed in the User Manager view. Instead, when you add a User, the new property “Prototype Name” provides a selection list of available prototypes.

Note: *In the User Manager view of the user service in any station, whenever you manually add a new user, property values in the **Default Prototype** are always used as defaults (regardless of whatever Prototype Name you may select in the Add dialog). In this way, the Default Prototype serves as a “template” to populate a new user’s properties (all except password).*

This can simplify user management even in a “non network user” scenario, to specify typical user property settings in the UserService’s Default Prototype. For more details see “[Default Prototype](#)” on page 6-37.

For more details on UserService items related to network users, see:

- “[Network user related properties](#)” on page 6-34
- “[About User prototypes](#)” on page 6-36

NiagaraNetwork changes

Each NiagaraStation (device component) under the station’s NiagaraNetwork has a Users device extension, in addition to other standard device extensions like Points, Histories, Schedules, and Alarms. The Users extension contains properties that enable/configure network user synchronization “in” and “out” of this station, in relation to the station with its NiagaraNetwork. There is no special view (apart from property sheet) on this Users device extension, nor is it a container for other components.

Associated with this device extension is a view on the parent NiagaraNetwork: the *User Sync Manager*. This tabular view provides an aggregate look at all Users device extensions (one for each NiagaraStation). Each row represents a station, and columns lets you see every station’s Users properties for sync configuration, sync status, sync “strategy,” and so on. You can select any or all rows for an edit dialog to make configuration changes, or issue manual sync commands.

Note: *User synchronization requires stations (Supervisor, JACEs) to be using compatible password storage mechanisms. So, for two stations to successfully synchronize user accounts, both stations must be running either “pre-update” builds, or 2013 update release or later builds (e.g. AX-3.8, AX-3.7u1, AX-3.6u4). Otherwise, a user sync from an AX-3.8 or AX-3.7u1 Supervisor to a station (JACE) running an earlier (pre-update) release will fail. The corresponding NiagaraStation’s Users device extension will also be in fault.*

For more details on the “Niagara Network side” of network users, see these sections in the *Drivers Guide*:

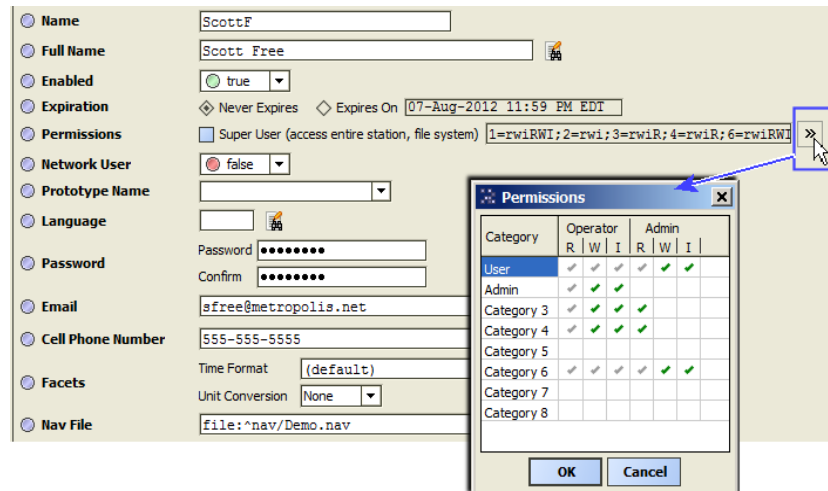
- “About the Users extension”
- “About the User Sync Manager”

Permissions and security

Permissions define what rights a user has within each of the [categories](#) in the station. There are two Niagara permission levels: operator and admin. Within both levels, separate options exist for read access, write access, and invoke (action) access. See “[About permission levels](#)” on page 6-10.

When you add a user, a Permissions field is available for you to set that user’s permissions, using a Permissions popup dialog (permissions map), as shown in [Figure 6-6](#).

Figure 6-6 Permissions in user's properties



Typically, you can use the Permissions dialog to assign specific rights to select categories for that user, such as shown in [Figure 6-6](#).

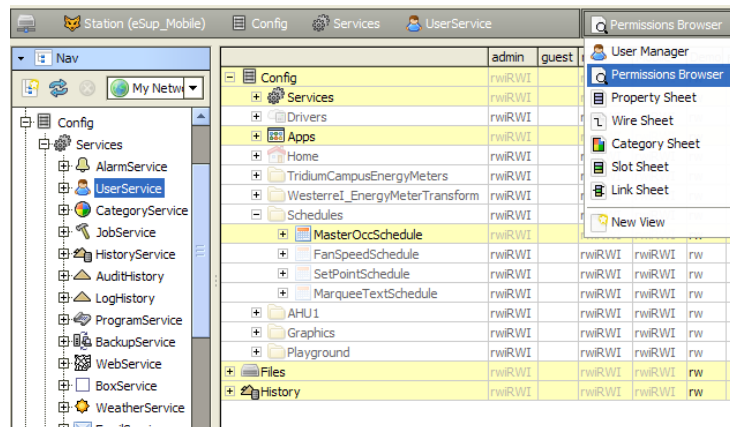
The built-in admin account has all possible rights for every category (super user). Only when logged in as the admin user or another “super user”, can you assign super user rights, via the Super User checkbox.

Note: Unlike in older NiagaraAX releases, a user cannot assign permissions to a user account that exceed the permissions assigned to their own user account.

In general, assigning super user rights should be limited on a strict “special needs” basis. A possible example is in a Supervisor station, for the user referenced in client connections from other stations (vs. login by a person), in scenarios where Program objects are exported from JACE stations via ExportTags. Or, for “person users” who need to add/edit Program or Robot components.

To review and compare permissions among all users, the UserService provides a PermissionsBrowser view, as shown in [Figure 6-7](#). Typically, you use this view to adjust permissions for any user.

Figure 6-7 PermissionsBrowser (UserService view)



In the PermissionsBrowser, *columns* represent individual [users](#) defined in the station. An expandable tree lets you navigate to objects of interest to review current permissions, shown in table cells using *abbreviations* (for read, write, and invoke). Operator level is small case; *admin* level is UPPER case.

Row coloring in the PermissionsBrowser provides the same data as in the CategoryBrowser, where:

- Yellow rows are objects explicitly assigned into one or more categories.
- Dimmed rows represent objects *inheriting* their parent's category or categories.

You can double-click in any column to access that user's permissions map. See “Permissions Browser” on page 6-34 for more details.

About permission levels

Once a user has been authenticated (see “[About authentication](#)” on page 6-6), that user is granted or denied permissions for each protected object in the system using the user’s “permissions map” (a category-based matrix, for an example see [Figure 6-38](#) on page 6-35).

Depending on object type, permission meanings vary as follows:

- [Component permission levels](#)
- [File permissions](#)
- [History permissions](#)

Component permission levels

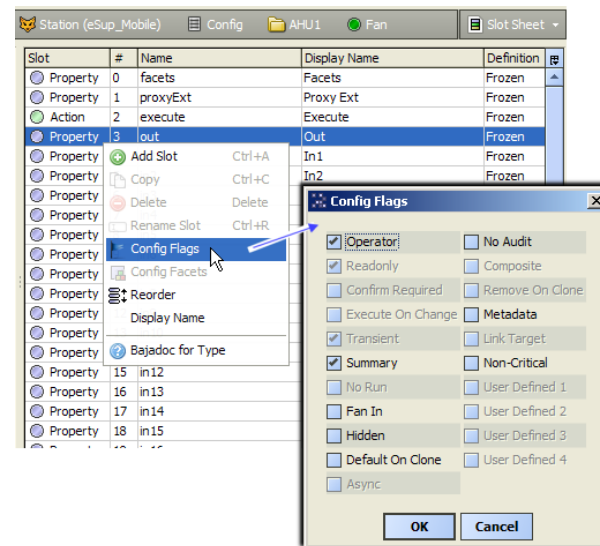
Note: By design, the *UserService* component enjoys a special permission scheme—and one that varies from the scheme described in this section. See “[UserService security notes](#)” on page 6-11 for details.

Each slot in any component is assigned to one of two permission *levels*, called *operator* and *admin*. This level is determined by whether the “Operator” config *flag* is set for that slot.

- If set (checked), the slot can be accessed by a user with a minimum of operator-read [permissions](#).
- If cleared (unchecked), a user requires at least admin-Read permissions to access the slot.

With admin-Write permissions, you can see and change slot flags from the slot sheet of any component, as shown in [Figure 6-8](#). By *default*, most slots are admin level (“out” is typically operator level).

Figure 6-8 Slot is admin level unless Operator flag is set

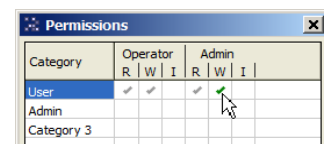


Permissions To control user access from a permission level, *six permissions* are derived, as follows:

- Operator-read — Allows the user to view operator-level information.
- Operator-write — Allows the user to modify operator-level information (if not read-only).
- Operator-invoke — Allows the user to view and invoke operator-level operations (actions).
- Admin-Read — Allows the user to view admin-level information.
- Admin-Write — Allows the user to modify admin-level information (if not read-only).
- Admin-Invoke — Allows the user to view and invoke admin-level operations (actions).

When you assign a user’s permissions, higher-level permissions automatically “include” lower-level ones.

Figure 6-9 Lower-level permissions include automatically



For example, as shown in [Figure 6-9](#) if you enable admin-level write (w), admin-level read (R) is automatically enabled, as well as operator-level read and write (r and w).

Ancestor permissions Often, you may wish to grant a user permissions to components using categories not included in their parent components, such that permissions to a component's "ancestor tree" are not explicitly granted. In this case, the system automatically grants "Operator-read" access to all "ancestor" components in the component tree. Otherwise, a user would be unable to navigate to target components in the Nav tree.

This "automatic ancestor permission" assignment is done by the station periodically, but you can force it at any time with a right-click "Update" action on the [CategoryService](#).

File permissions

Note: *Starting in AX-3.7 and in security patches for other release levels, basic changes occurred with file access to increase security. A station's `config.bog` file (and `config.bog.backup` files) are no longer accessible (even by super users) in the station's File space. Note other station files and folders may be "blacklisted" from remote station access, if needed.*

Additionally, any new station created using the New Station Wizard, by default, has the entire station's File space assigned to category index 2 (named "Admin"), whereas prior to AX-3.7 all objects were assigned to category index 1. Users typically require operator-read permissions on station file folders like `^nav`, `^px`, `^images`, `^html`, and so on. However, permissions higher than "operator-read" on the Admin category should only be assigned to selected users on a "as needed basis".

Largely, [permissions](#) given to [categories](#) used by files and folders are "operator-keyed," as follows:

- **Files**
Files require operator-read access to view, and operator-write permissions for editing (if applicable). For instance, a user with operator-write permissions to an .html file can modify it using the **Text File Editor** in Workbench.
- **Folders**
Folders (directories) require operator-read access to list and to copy child files, and operator-write access to create new (or delete existing) child files.

A few views for files require *admin-write* permissions to access, such as the **Nav File Editor** for a "nav file." There are also other [special case file permissions](#).

Special case file permissions These additional rules apply to file system access from station access:

- Files in any NiagaraAX module are automatically granted operator-read.
- If a user is not a "super user," all access outside of the station's home directory is denied.
- Admin-read rights are needed to see a Supervisor station's `^provisioningNiagara` folder (written to by the Supervisor's "Niagara Provisioning" mechanism).

History permissions

Currently, histories require only operator-read permission to access all available views (e.g. History Chart, Collection Table, History Table). In addition, the ability to rename a history is included.

UserService security notes

In order to facilitate user management, a special [permissions](#) scheme applies to the UserService (only), where permissions are *inherited* by child users (unless a user is assigned to a different category) as follows:

- A user with only operator-read permissions (r) has *read-only* access to operator-level properties of their *own* user account (*all other users are hidden*).
- A user with operator-write permissions (rw) has *read/write* access to operator-level properties of their *own* user account (*all other users are hidden*).
- A user with admin-Read (rR) permissions has read-only access to all properties of *all available users*.
- A user with admin-Write permissions (rwRW) has read/write access to all properties of *all available non-super users*. Moreover, they have access to the [User Manager](#), so can *add* new users and also *delete* selected users. In addition, the [Permissions Browser](#) view of the UserService is available.

Note: *Starting in AX-3.7 and in security patches for prior releases, changes were made such that a user cannot assign permissions to other users that they do not have themselves. For example, a "non-super user" cannot assign other users permissions on categories that they lack.*

By default, operator properties of [User](#) are Email, Password, Cell Phone Number, and Facets (time format and unit conversion). If needed, from the slot sheet of [User\(s\)](#) you can edit config flags to change which slots are operator versus admin. For example, you might change the `fullName` slot to operator.

This simplified scheme is useful when you want to let each user reassign their own password, but not have access to other users. In this case, give all “non-super” users operator-write (rw) permissions *only* on the UserService. By default starting in AX-3.7, the New Station Wizard assigns the UserService to the “Admin” named category (category 2), along with the CategoryService and UserService. Remember, any user granted *super user* permissions has *all access to all objects*, and moreover can add more super users.

About user audits

An important aspect of station security is an audit trail to record user activity. Any NiagaraAX station can be configured for user audits through use of the **AuditHistoryService**. While enabled, this service creates a history of user-initiated changes to components in the station.

Note: *If this service is not already in your station's Services container, open the **history** palette and drag or copy the **AuditHistoryService** into your station's Services folder.*

The following sections provide more details:

- [AuditHistoryService configuration](#)
- [AuditHistory content](#)
- [AuditHistory station login activities](#)

AuditHistoryService configuration

AuditHistoryService configuration is simplified by having only the following writable properties:

- **Enabled**
Either true (default) or false (disabled). Typically, you always leave this enabled.
- **History Config**
Contains some read-only status properties and a few writable properties, including the following:
 - **Capacity**
Either: Record Count *nnn* records (default is 500) or Unlimited. Typically, a very large number is acceptable for Supervisor stations, for example 250000 (in general the “Unlimited” choice is not the wisest even for a Supervisor), while in JACE stations history capacities are more appropriately set closer to the default 500 records or less. If a JACE station, the resulting AuditHistory is typically exported to a Supervisor station for archives.
 - **Full Policy**
Either Roll (default) or Stop. In almost all cases, you leave full policy at roll to ensure latest user actions are recorded. Note the full policy has no effect if the capacity is “unlimited”.
 - **System Tags**
(Use optional). Allows using the System Tag feature for importing or exporting Niagara histories. For more details, see “[About System Tags](#)” on page 4-29.

AuditHistory content

The AuditHistory records user-initiated changes to property values, invoked actions, as well as any slots that are added, deleted, reordered, and renamed. Each record appears as a row that includes the timestamp of the change and the operation, plus other details including the responsible station user.

Using standard history views, you can review and resort data columns in the AuditHistory as needed to analyze user activity. [Figure 6-10](#) shows an AuditHistory displayed in the HistoryTable view. In this view (as shown), you can double-click any row for a popup dialog displaying record details.

Figure 6-10 AuditHistory in station Histories

Timestamp	Operation	Target	Slot Name	Old Value	Value	User Name
19-Jun-12 9:38:51 AM EDT	Logout	/Drivers/NiagaraNetwork	192.168.1.37			admin
19-Jun-12 9:39:04 AM EDT	Login	/Drivers/NiagaraNetwork	192.168.1.37			FacMan
19-Jun-12 9:39:32 AM EDT	Changed	/Schedules/MasterOccSchedule/schedule/week/friday/day/time	finish	7:00 PM	6:00 PM	FacMan
19-Jun-12 9:41:12 AM EDT	Invoked	/AHU1/Fan	auto			FacMan
19-Jun-12 9:42:03 AM EDT	Invoked	/AHU1/Fan	auto			FacMan
19-Jun-12 9:42:35 AM EDT						
19-Jun-12 9:43:21 AM EDT						
19-Jun-12 9:43:21 AM EDT						
19-Jun-12 11:05:56 AM EDT						
19-Jun-12 11:06:07 AM EDT						
19-Jun-12 11:06:50 AM EDT						
19-Jun-12 11:07:15 AM EDT						
19-Jun-12 11:08:09 AM EDT						
19-Jun-12 11:09:14 AM EDT						
19-Jun-12 11:09:22 AM EDT						
19-Jun-12 11:09:35 AM EDT						
19-Jun-12 11:13:05 AM EDT	Login	/Drivers/NiagaraNetwork	192.168.1.109			admin
19-Jun-12 11:14:17 AM EDT	Logout	/Drivers/NiagaraNetwork	192.168.1.37			Operator

Standard fields in an AuditHistory record include the following:

- **Timestamp**
Timestamp of when the operation on the slot became effective.
- **Operation**
Type of user operation, which may include one of the following:
 - Added — Added a slot (typically a point or extension)
 - Changed — Changed a property (slot) value
 - Invoked — Issued a command to an action slot
 - Removed — Removed a slot (typically a point or extension)
 - Reordered — Reordered one or more slots
- **Target**
Path to the container of affected slot, relative to the station root.
- **Slot Name**
Name of the affected slot.
- **Old Value**
Value of slot before a change.
- **Value**
Value of slot after the change.
- **User Name**
User name that initiated the operation.

AuditHistory station login activities

Station login activities are included in a station's AuditHistory. The following login-related events by either a Workbench (fox) user or a browser user appear in the station's AuditHistory as records, as one of the following operations:

- Login — valid user name and password supplied
 - Logout — close connection if Workbench, or explicit “/logout” submission if browser
- Note:** If a browser user simply closes the browser window, a logout audit record is not collected. Thus, a corresponding logout may not appear for every browser login, unless an explicit “/logout” is always forced.

- Login Failure — valid user name, but invalid password supplied

In any case, audit records are *not* produced for any disabled users (including users that are “locked out”) and for any unknown users.

Figure 6-11 shows various login-related records in an AuditHistory displayed in the HistoryTable view. As shown, you can double-click any row for a popup dialog displaying record details.

Figure 6-11 Login related records in station's AuditHistory

The screenshot shows the 'eSup_Mobile/AuditHistory' window with a table of 447 records. A 'History Record' dialog box is open, showing details for a record with the following fields:

Field	Value
Timestamp	19-Jun-12 9:37 AM EDT
Operation	Login Failure
Target	/Drivers/NiagaraNetwork/foxService/serverConnections/Session3363
Slot Name	192.168.1.37
Old Value	
Value	1
User Name	FacMan

Specific to a security (station login-related) event, fields in a record are used as follows:

- **Timestamp**
Timestamp of when the login/logout action occurred.
- **Operation**
The security action that occurred, as either “Login”, “Logout”, or “Login Failure”.
- **Target**
Depending on if a Workbench (fox) or browser (WebService) user connection, the path as either:
 - Workbench - Path to fox session, usually as /Drivers/NiagaraNetwork/FoxService/etc.
 - Browser - Path to the web service were validated, typically as /Services/WebService
- **Slot Name**
The remote host that attempted login/logout, typically by IP address.
- **Old Value**
Not used.
- **Value**
Not used unless the Operation is “Login Failure”, in which case the value indicates the number of consecutive failed login attempts (1—5). After 5 consecutive failed login attempts, if lockout is enabled, the user is locked out for some period of time. Subsequent login attempts during that period are not audited. Also see “[Lockout notes](#)” on page 6-29.
- **User Name**
User name supplied during the attempted login/logout.

Multi-station security notes

Many installations have multiple stations. Typically, this means a Supervisor station plus one or more JACE stations. In this scenario, often the same user(s) need to access different stations. In addition, you typically create a special “station-to-station user” in each station for use in the NiagaraNetwork. Please note the following:

- By convention, a station-to-station user can be named something memorable (perhaps unique to your company or even a job site), *and* you should *carefully guard its password*—as you typically assign it many “admin write” permissions. When adding this [User](#), properties that apply to browser access are inconsequential, for example, Facets, Nav File, and Web Profile.
Do not use this account for station login! Instead, you reference this user in another station, when adding a NiagaraStation device under a NiagaraNetwork. Refer to the “Station Add notes” section in the *Drivers Guide* for related details.
- You can use the “network user” feature to automatically replicate users among stations, and keep them centrally managed. Configuration involves two different areas of stations: UserService and NiagaraNetwork. For an overview, see “[Network users](#)” on page 6-7.
- Due to security changes, you can no longer (alternatively) copy users from one station to another with any success. While this was permitted in earlier releases of NiagaraAX, it was cumbersome and inefficient when a change to such a user was needed. You can copy *categories* from station to station; however, note that the index property in each copied category is reset to 0, and the category has a disabled status (until you reassign the index).
- Often, a user that has already logged into a station is re-challenged for login ([authentication](#)) again. This can happen when navigating from one station to another station. Depending on whether a browser user or a (full) Workbench user, it may be possible to configure stations such that a user needs to be authenticated only *once* (upon initial station connection).
 - If a browser user (Web Workbench or Hx access), and Niagara hosts are installed on a *DNS domain*, you can configure stations such that browser users are authenticated only *once* during the same browsing session. See [Domain-wide cookies authentication](#).
 - If a Workbench user, you can configure stations such that the user is authenticated only *once*. See [Workbench single sign-on \(SSO\) realm](#).

Domain-wide cookies authentication

This feature is available for multi-station jobs where all Niagara hosts are installed on the same DNS domain. It allows for stations that have the same user accounts (matching user credentials) to be accessed by any such user using a “domain-specified” browser connection and a *single*, initial, login.

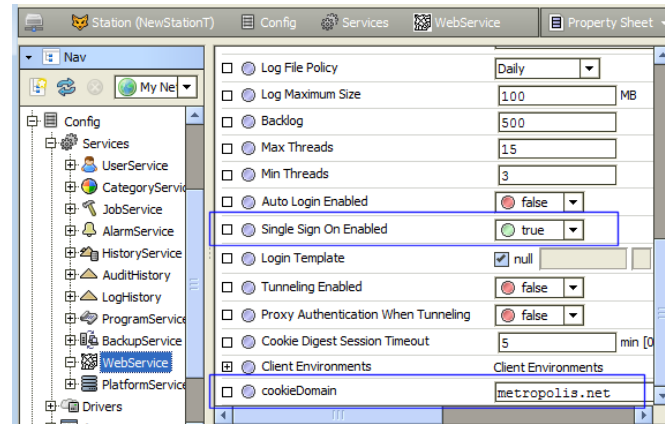
To configure domain-wide cookies authentication

Perform the following in any station that may provide the initial browser connection.

To configure domain-wide cookies authentication, do the following:

- Step 1 Open the slot sheet of the **WebService** component (in the station’s Services container).
- Step 2 Add a new slot, naming it “`cookieDomain`”, leaving other fields at defaults (type `baja:String`).
See “[Using the slot sheet](#)” on page 9-19 for details on adding a slot.
- Step 3 Go to the property sheet of the WebService.
In the new `cookieDomain` property, enter your DNS domain name (see [Figure 6-12](#) for example).
- Step 4 Set the Single Sign On Enabled property to **true**, changing it from the default false.
Note: *Property Single Sign On Enabled is new starting in AX-3.7u1; it is also available in AX-3.6u4/AX-3.5u4.*
- Step 5 Verify that the WebService’s property Authentication Scheme is set to `Cookie`.
Note: *In AX-3.7 and later the Authentication Scheme property defaults to `Cookie Digest`, for best security. However, in some cases it may be necessary to set it differently, as in this case to `Cookie`.*
- Step 6 **Save** this configuration of the WebService.

Figure 6-12 Example configured WebService in AX-3.7u1



Now, if a user points his or her browser to this Niagara host using the full DNS name (for example: supervisor.metropolis.net), after station login, they can navigate using links to other stations without need for login. The credentials used in the initial login are used in other station connections.

Note: To support the domain-wide cookies authentication feature in browsers, hyperlinks to views in other stations must reference “hostname.domain” in either an “ip ord” format or “full url” format. Using a numerical IP address instead of “hostname.domain” in hyperlinks results in a login rechallenge.

Examples using “hostname.domain” (jace1.metropolis.net):

- ip ord
ip:jace1.metropolis.net|fox:|station:|slot:/PxHome
- full url
http://jace1.metropolis.net/ord?station:|slot:/PxHome

Providing that the WebService in the target station is using the standard HTTP port (port 80), using the “ip ord” format in hyperlinks may be best, as it works in both Workbench and web browsers. Whereas, the “full url” format works in web browsers, but produces a “Cannot display page” exception in Workbench. However, the “full url” format is required if the target station is running a non-standard HTTP port. For example, if the target station is using an httpPort of 8080, an example hyperlink would look like:

- full url (specifying port)
http://jace1.metropolis.net:8080/ord?station:|slot:/PxHome

Notes on using Hosts files for domain-wide cookies authentication A private network of JACEs and/or Supervisor may be installed on a LAN without a domain controller or local DNS server. In this case, when configuring the WebService in stations, you can specify a “virtual” (fictional) domain name. Then, in order to provide domain-wide cookies authentication, you must manually edit the “hosts” file on each LAN-connected PC at the job that needs browser access to the Niagara system. In each PC’s hosts file, you associate the same virtual domain name with the actual IP address for each Niagara host.

Note: On most Windows PCs, the hosts file (no file extension) resides under the subdirectory “%SYSTEMROOT%\SYSTEM32\DRIVERS\ETC”, e.g. “C:\WINDOWS\system32\drivers\etc”. The hosts file is a simple text file you can edit and save with Notepad or similar editor.

For example, in the WebService of your Supervisor station you added the cookieDomain slot, and entered a virtual domain of golly.net. You also did the same thing in the station running on each of three JACEs. The static IP addresses for these four Niagara hosts are:

- Supervisor 192.168.1.99
- JACE A 192.168.1.85, JACE B 192.168.1.86, JACE C 192.168.1.87

So the contents of the hosts file on each PC (that needs browser access to the system) should include entries like:

```
127.0.0.1 localhost
192.168.1.99 supervisor.golly.net
192.168.1.85 jace1.golly.net
192.168.1.86 jace2.golly.net
192.168.1.87 jace3.golly.net
```

Once the hosts file is edited on a PC, you should be able to start a browser session with the URL pointed to a Niagara host by its domain name (for example: supervisor.golly.net). After initial user login, “hostname.domain” links to views in other stations should occur without being rechallenged for login again, providing of course that the same user and password exists in the target station.

Note: *It is not necessary to edit the `hosts` file on any Niagara platform, unless it is a PC that is also used for browser access of stations.*

Workbench single sign-on (SSO) realm

A multi-station “realms” feature is also available. This allows a Workbench user to open one station (fox), then have Workbench access to other stations, without re-prompting for credentials. For example, a Px view in a station may contain widgets with hyperlinks to views in other stations.

This feature requires that the user has an account (with same credentials) on each station in the realm. Otherwise, the user will still be prompted to login. Therefore, this feature works well with the “network user” feature (see “[Network users](#)” on page 6-7 for details).

Note: *The “SSO realm” feature works only for a “full Workbench” (Fox) connection to a station. It does not apply to “Web Workbench” connections, that is browser access to a station via the `WebService` and `WbApplet`.*

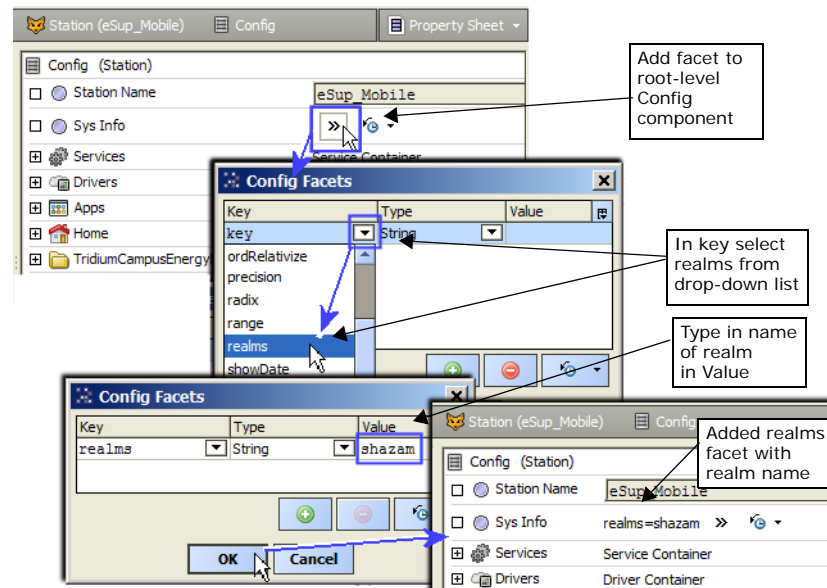
To configure station’s realms for Workbench single sign-on

In a multi-station job, perform the following in *each* station, where each station must have the same “realm” name.

To configure a station’s realm, do the following:

- Step 1 Open the property sheet of the root-level **Config** component in the station.
- Step 2 In the “Sys Info” property, click the Facets control (>>) for the Config Facets editor.
The Config Facets editor appears.
- Step 3 Add a key named “realms” of type String, with the string value (whatever desired) for the realm name.
To do this in the Config Facets dialog:
 1. Click the Add (+) control.
 2. In the Config Facets dialog, click the drop-down control for `key` and select `realms`.
 3. In that same facet row, ensure that “Type” is `String`, and click in the “Value” field and type the name of the realm. Note this realm value has no literal meaning, but must be the *same* for all stations.
 4. Click **OK** to add the facet.
- Step 4 See [Figure 6-13](#) for an example of the realm “shazam” being added.
Repeat this procedure for every station in the job.

Figure 6-13 Add key named “realms” in Facets for “Sys Info” property of the root-level Config component



Now, if a user opens a station in Workbench (enters their credentials), they can navigate using links to other stations in this realm without need for login. The same credentials used in the initial login are used in other Workbench station connections.

Alarm ack permission notes

When assigning permissions to station users, keep in mind that all alarm ack (acknowledgement) rights in NiagaraAX are determined only by a user's permissions on *AlarmClass* components (and not on the source points with alarm extensions, which “generate” alarms). Alarm classes are under a station's Alarm-Service. As needed, you link alarm classes to alarm recipients, to define alarm routing in the station.

Specifically, a user requires *write* permissions on any alarm class in order to acknowledge alarms or events routed by it. Minimum permissions vary depending if AX-3.8 or AX-3.7, as follows:

- In AX-3.8, users require *operator write* (rw) permissions on the alarm class, in order to either acknowledge the alarm, or add “notes” to it. Note *this is a change starting in AX-3.8*—see below.
- In AX-3.7 and earlier releases, users require *admin write* (RW) permissions on the alarm class, in order to acknowledge the alarm, or add “notes” to it.

This topic is also mentioned because it is a different alarm ack scheme than was used in the older Niagara (R2) systems. Note that in a scenario where many “user-divided” ack rights are needed, you may need to create additional alarm class objects, assigning them to different security categories.

For more details about alarming in general, and also AlarmClass components in particular, see [“About Alarms”](#) on page 5-1, and [“About alarm class”](#) on page 5-32.

About password expiration and reset

Starting in AX-3.7, station user security was enhanced by a password expiration mechanism, to require users to periodically *change* their login password. Usage is optional, but *recommended*. Independently, you can also selectively configure any user to reset (change) their password upon next login.

- Note in AX-3.8, a user with a password about to expire (or a password set to reset) must enter a new password that meets the UserService's minimum “Password Strength” criteria. The login dialog for the user clearly specifies this criteria. For related details, see [“Password Strength \(AX-3.8\)”](#) on page 6-25 and [“Password entry dialog improvements”](#) on page 6-26.
- In AX-3.7, if the UserService's “Require Strong Passwords” property is true (the default), the user must enter a strong password. For related details, see [“Require Strong Passwords \(AX-3.7\)”](#) on page 6-27.
- Finally, in some cases a password change upon login attempt may not succeed. This can happen to any “network user” that is *not* logging into the “source station”—typically the Supervisor. Starting in AX-3.7u1, such an unsuccessful password reset results in a message that tells the user that they must log into the station that *does* centrally manage their user account, and reset their password there. For related details, see [“Network users”](#) on page 6-7.

For more details, see the following sections which apply to AX-3.7 and later stations:

- [Password expiration](#)
- [Password reset](#)
- [Password history \(unique passwords\)](#) — applies to both password expiration and password reset

Password expiration

At some specified interval, a user in a AX-3.7 or later station can be periodically prompted to change their password at login, to avoid expiration of their user account. This applies whether the user is opening a station connection from Workbench or accessing the station from a browser.

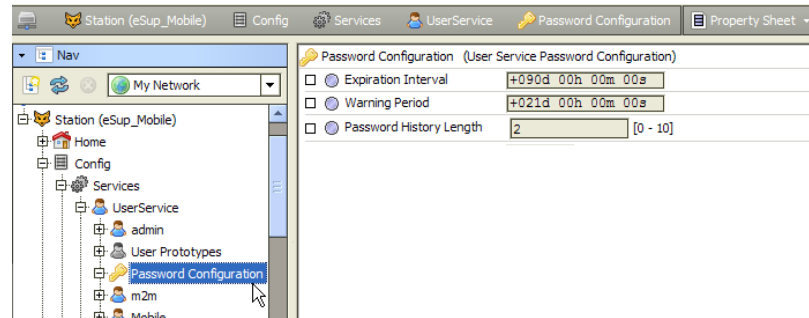
Details are in the following sections:

- [“Configuring expiring passwords”](#) on page 6-18
- [“Expiring password operation”](#) on page 6-20

Configuring expiring passwords Configuring password expiration involves two separate areas:

- “Global” properties of a “Password Configuration” slot directly under the station's UserService.

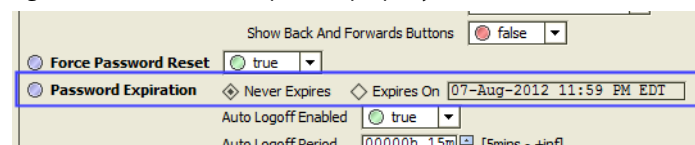
Figure 6-14 Global password configuration properties under UserService's PasswordConfiguration slot



These *global* password configuration properties are as follows:

- Expiration Interval**
 Specifies the repeating interval for password expiration. At the time of this document update, the default is 1 year (365 days). Often, this is configured for a *shorter interval*, for example 90 days. This interval applies to any users configured for periodic password expiration.
- Warning Period**
 Specifies the time before a user's password expires that is a "warning period", with a default of 30 days. During this period, any station login by the user produces a popup warning about the upcoming expiration, and offers the user a choice to reset (change) their password.
Note: Any user that allows their password to expire will be unable to login to the station! Users need to be cautioned about this, as this differs from other some other systems. See the related [Caution](#) on page 6-20.
- Password History Length**
 At the time of this document update, the default is 0 (no effective password history); the maximum value is 10. Any positive non-zero value (1, 2, 3, etc.) means a user with an expiring password cannot simply re-enter the identical last (1) password, or one of the last (2) passwords, and so on. Instead, the user is prompted to enter a password *unique* from any of these passwords. This setting also applies to a "password reset" given to any user—e.g. a user with Password Configuration of "Never Expires".
Note: For improved security, change this from the default (0) to at least 1 or 2.
 See "Password history (unique passwords)" on page 6-23 for more details on this feature.
- Each User has a **Password Expiration** property, editable in the **User Manager** (Figure 6-16).
Note: Each User has another "Expiration" property, which has a different application completely. Always leave it at "Never Expires", except in the case of a temporary user account.

Figure 6-15 Password Expiration property for each user, from New/Edit user dialog in User Manager

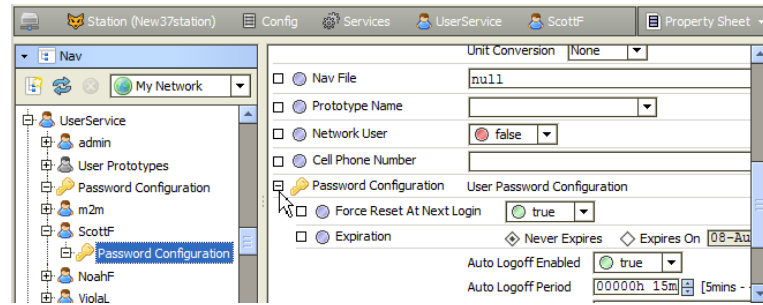


Default is "Never Expires" (no periodic password expiration for this user). Change to "Expires On" and enter a date in the *future* to configure a user for automatic password expiration.

If creating a user with an expiring password, typically you set the "Force Password Reset" property, also shown in Figure 6-16 above, to true. Then, when that user first logs in, they are prompted to change their password, where after successful entry their password expiration date is reset to the full (global) expiration interval.

Note: These two properties are actually in a separate "Password Configuration" container under each user, visible in a User's property sheet (Figure 6-16).

Figure 6-16 Password configuration properties unique to each station user



As shown above, these properties appear on a User's property sheet listed as "Force Reset At Next Login" and "Expiration". They are the same properties seen from the User Manager, as shown in [Figure 6-15](#)



Caution

The user account used for station-to-station NiagaraNetwork connections (service user) should have a password configuration of "never expires", and never forced to password reset (although you should not use this account for user login anyway). However, it is strongly encouraged to have a local policy to periodically change the password for these (service) user accounts.

Any (person) user that allows their password to expire will be unable to login to the station! A system administrator must change that user's "Password Configuration, Expiration" date to allow them to regain access. Users need to be cautioned about this, as this differs from account expiration in some other systems.

Expiring password operation A user with an expiring password (in the "warning period") sees a related message when opening a station from a browser ([Figure 6-17](#)) or from Workbench ([Figure 6-18](#)).

Figure 6-17 Browser access login example when password is expiring (warning period)

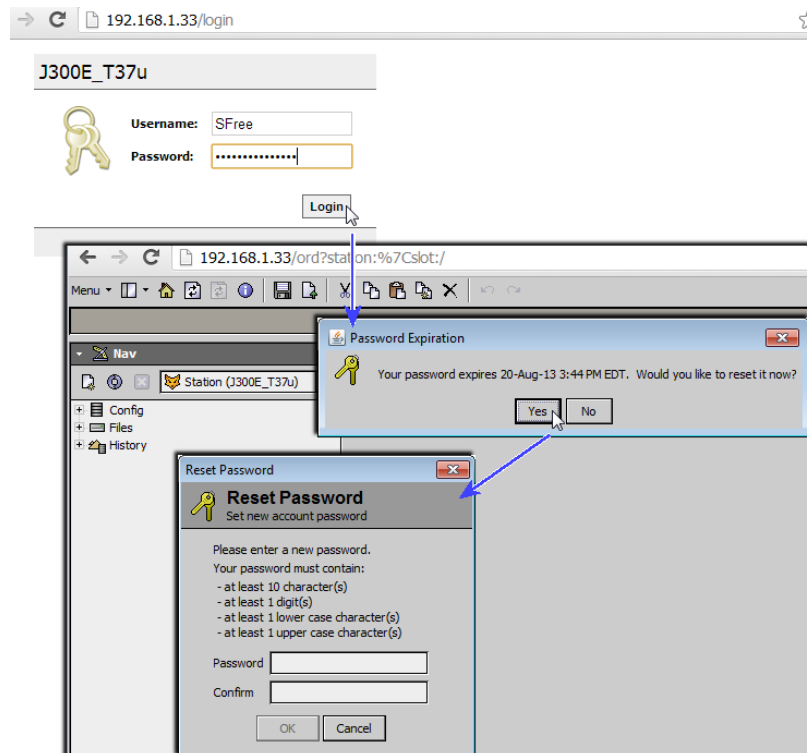
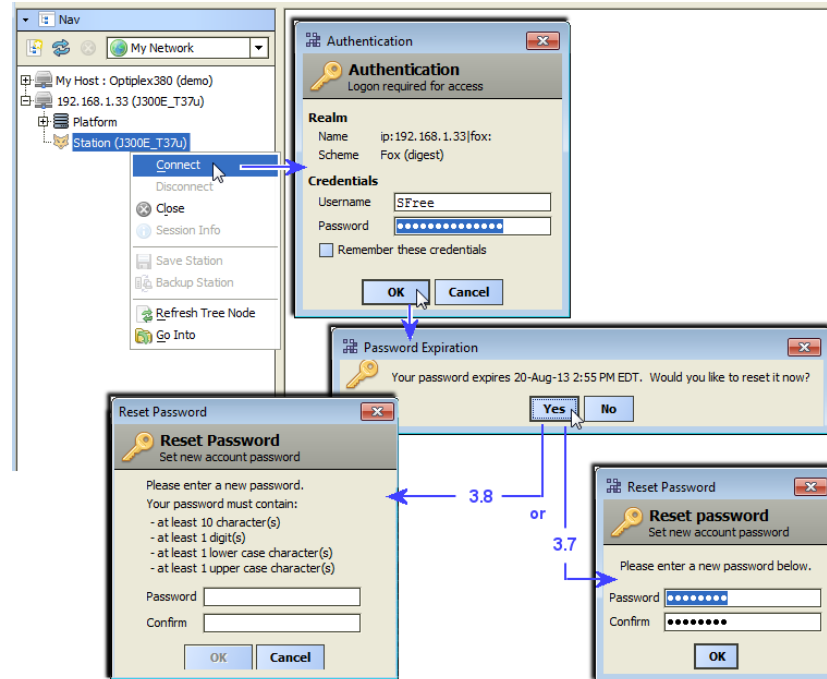


Figure 6-18 Workbench login example when password is expiring (warning period)



Caution Again, any user that allows their password to expire will be unable to login to the station! Users need to be cautioned about this, as this differs from other some other systems. A system administrator must change a user's "Password Configuration, Expiration" date to re-allow access—see [Figure 6-16](#) on page 6-20.

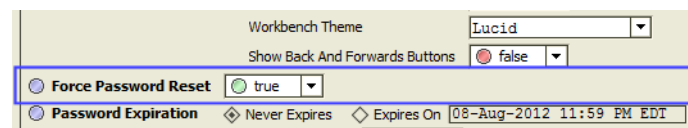
Upon a user's password change, the globally-defined expiration interval (say 90 days) for that user is reset, as well as the globally-defined warning period (say 30 days before expiration), and this cycle repeats.

Coupled with expiring passwords, you also typically configure to prevent reuse of the previous password(s). See "[Password history \(unique passwords\)](#)" on page 6-23.

Password reset

Independent from the automatic "password expiration" mechanism, you can force a "password reset" on any user, including one configured to "never expire". At the user's next login to the station, they are prompted to *change* their password.

Figure 6-19 Force Password Reset for a User as seen in Edit dialog from User Manager



As shown in [Figure 6-19](#) above, this appears in the New/Edit dialog for each user in the User Manager. Note this is the same property "Force Reset At Next Login" as seen in the Password Configuration container under each User component (see [Figure 6-16](#) on page 6-20).

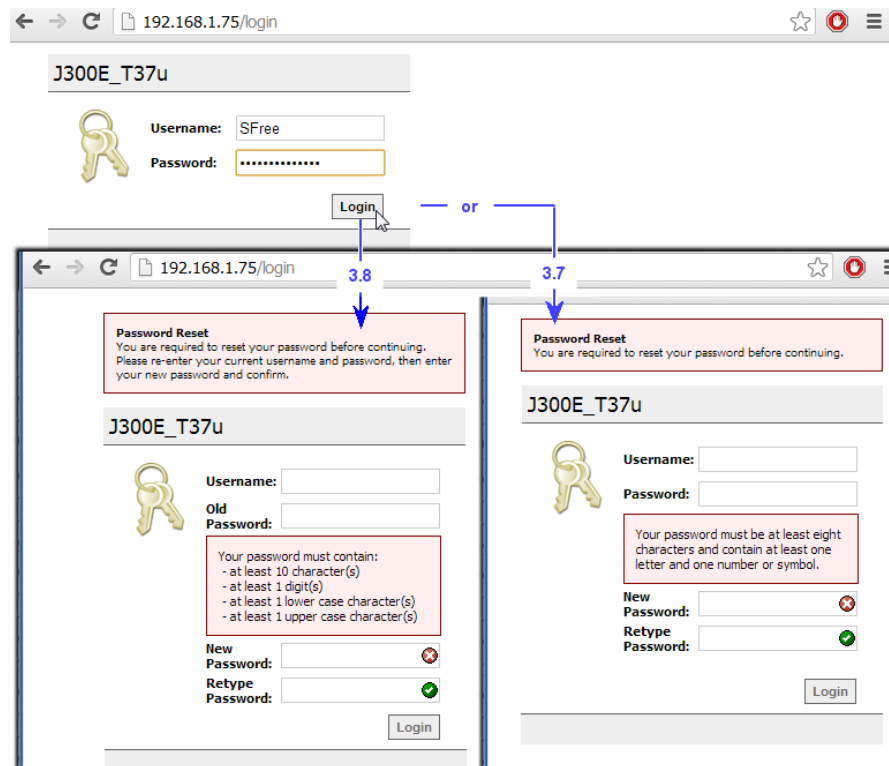
In the original AX-3.7 release, by default when using the **User Manager** to create a *new* user, "Force Password Reset" was set to *true*. (This default changed starting in AX-3.7u1 and AX-3.8 to false.) In either case, if you wish to change this behavior, change this property in the "Default Prototype" under the UserService. For more details, see "[Default Prototype](#)" on page 6-37.

You can initiate this reset for any user(s) from the **User Manager**, by selecting the user(s) and clicking the **Edit** button to access the **Edit** dialog. This dialog includes the "Force Password Reset" entry, which by default is *true*. Or, do this from the property sheet of a User by expanding their "Password Configuration" container to access this same property (see [Figure 6-16](#) on page 6-20).

Again, coupled with password reset you also typically configure to prevent reuse of the previous password(s). See "[Password history \(unique passwords\)](#)".

Password reset operation Figure 6-20 shows an example of how a password reset appears when accessing the station from a browser.

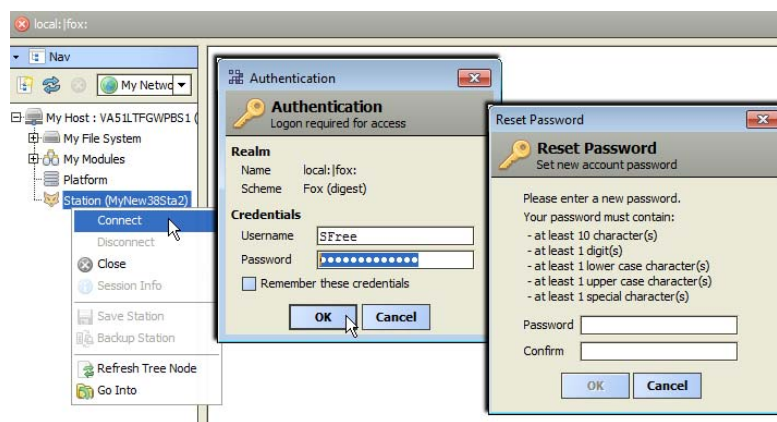
Figure 6-20 Browser access login example when password has been marked for reset



As shown above (left side), if an AX-3.8 station, the effective “password strength” rules are given. If an AX-3.7 station enabled for strong passwords, the standard “fixed” strong password rules are shown.

Figure 6-21 shows how a password reset appears when opening the station in Workbench.

Figure 6-21 Workbench login example when password has been marked for reset



Note: For related details, see “Strong password notes” on page 6-25.

After the user’s subsequent password change, their “Force Reset At Next Login” property (Force Password Reset) returns to `false`. If the user is also configured for periodic password expiration, their expiration deadline is reset to the full period.

Note: In certain scenarios where a leaked password is suspected, or the system security has been compromised, you may wish to reset the passwords of multiple users. You can do this from the **User Manager** view, using a “gang edit” of multiple selected users, changing the “Force Reset Password” entry to `true`.

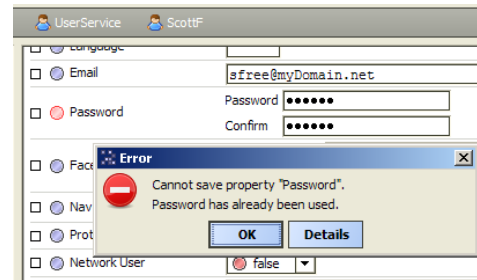
Password history (unique passwords)

When users' station passwords are changed in an AX-3.7 or later station, you can require the new password to be *unique* from the previous one, two, or three (and so on) entered passwords for each account. This is specified in a globally-defined "Password History Length" property, under the UserService's "Password Configuration" container slot (see [Figure 6-14](#) on page 6-19).

The default value of this property is "0", which permits the reuse of the current password. However, it is recommended you set this to at least 1 or 2, especially if users are configured for periodic password expiration, or if you use the password reset feature on any user. The maximum value is 10.

[Figure 6-22](#) shows an example popup **Error** dialog from an attempt to save a password that was changed to a previously used value, as edited on a User property sheet.

Figure 6-22 Popup error dialog as global "Password History Length" is enforced on a password change



After clicking OK to close the popup, the password remains at the same value as before the edit. The password change can be retried with another value, which must be unique from the previous one (at a minimum). Note that the station tracks n number of password values for each user, where n equals the "Password History Length" value. Any password change attempt is compared against tracked password value(s), with the save rejected if found to be a match.

UserService

Note: The UserService is included in the **Services** container of any new station created using the **New Station** tool in Workbench. Note that some installations may use LDAP or Active Directory integration, where another user service is required to replace the standard **UserService**. However, the basic operation of users remains the same, especially for “local” users created using the **Ldap User Manager**. For related details, refer to the NiagaraAX LDAP / Active Directory Configuration Guide.

The UserService is found under a station’s Services container. You use a station’s UserService to add, edit, and delete users. If you have already configured [categories](#) in the station, you can use it to assign users permissions as well.

Note: For an overview of station security and users, see [“Security model overview”](#) on page 6-2 and [“Users and security”](#) on page 6-4.

Before working with users, you should review the [UserService properties](#), and see [“Strong password notes”](#). After setting properties as needed in the service, the following UserService views are your primary access to users:

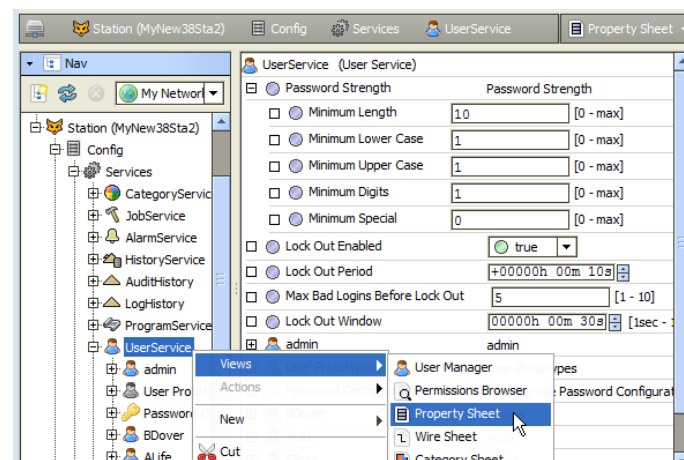
- [User Manager](#) to add Users
- [Permissions Browser](#)

Starting in AX-3.7, a station’s UserService also has a “Password Configuration” container slot. Properties are used in configuration of periodic password expirations, and enforcement of unique password changes. Each user component also has related “Password Configuration” properties, which appear in the New/Edit dialog for a user in the **User Manager**. For more details, see [“About password expiration and reset”](#) on page 6-18. Also see [“UserService security notes”](#) on page 6-11.

UserService properties

Right-click the UserService and choose **Views > Property Sheet**, as shown in [Figure 6-23](#).

Figure 6-23 UserManager property sheet access (AX-3.8 station shown)



UserService configuration properties are described as follows:

- **Password Strength**
(AX-3.8 only) A container with five properties that define the *minimum* characters/types required in station user passwords. Any future user additions or edits of any user’s password will require an entry that meets these minimum specifications, otherwise an “error” popup dialog results. For further details, see [“Password Strength \(AX-3.8\)”](#) on page 6-25.
- or
Require Strong Passwords
(pre-AX-3.8 only) Either true (default) or false—note the default in AX-3.7 has always been true. If enabled (true), which is strongly recommended, any future user additions or edits of any user’s password will require entry of a “strong password,” otherwise an “error” popup dialog results. If set to false, in AX-3.7u1 any password is valid. For further details, see [“Require Strong Passwords \(AX-3.7\)”](#) on page 6-27.
- **Lock Out Enabled**
Either false or true (default). If enabled (true), then a number of consecutive [authentication](#) failures will temporarily “lock out” login access to that user account, for the duration of the lock out period (next property). Using lock out makes it difficult to automate guessing of passwords.

Note: Each user also has a “Clear Lock Out” action. See “[Lockout notes](#)” on page 6-29.

- **Lock Out Period**

Default is 10 seconds, adjustable to any duration in hours, minutes, seconds. If lock out is enabled, then this is the period of time a user account is locked out before being reset. While locked out, any login attempt (even a *valid* one) will be unsuccessful.

Note: Default Lock Out values are intended to guard from an automated “brute force” password attack, where a computer application might issue hundreds of login attempts in a second. The 10 second latency is typically sufficient to thwart such an attack, as it must wait 10 seconds upon each unsuccessful 5 login attempts. If deemed necessary, you can adjust to guard against “human attack”.



- **Max Bad Logins Before Lock Out**

Default is 5, adjustable from 1 to 10. If lock out is enabled, in conjunction with the “Lock Out Window”, this specifies the number of consecutive failed login attempts that trigger a lock out for a user.

- **Lock Out Window**

If lock out is enabled, and the number of “Max Bad Logins Before Lock Out” occurs within this window of time, the user is locked out for the “Lock Out Period” duration. The lock out window default is 30 seconds, and is adjustable to any duration up to 1 day.

Note: Changes to lock out properties are enforced on the next login attempt for any user. For example, suppose “max bad logins” is set to 5, and user “ScottF” has failed to log in 4 times within the lock out window. Suppose an admin-level user now changes “max bad logins” to 3. This does not lock out user “ScottF”; user ScottF still has one more chance to log in before getting locked out. If that login attempt fails, ScottF will be locked out, since 5 failed attempts is greater than or equal to the max bad logins setting (3).

Note: Starting in AX-3.7, the UserService has other configuration properties under a  **Password Configuration** container, and each User also has a  **Password Configuration** container. Note that because they are “mix-in” properties, a new (offline) station will not have them until that station is started and saved. For related configuration and operation details, see “[About password expiration and reset](#)” on page 6-18.

Strong password notes

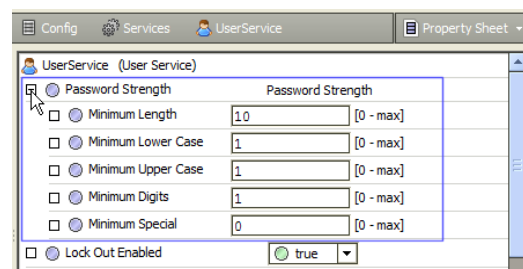
Strong passwords are always recommended for NiagaraAX station users, and in AX-3.8 improvements were made in this area. See the following for more details:

- [Password Strength \(AX-3.8\)](#)
 - [Password entry dialog improvements](#)
- “[Require Strong Passwords \(AX-3.7\)](#)” on page 6-27 (AX-3.7)
- “[Stronger passwords](#)” on page 6-28 (applies to all releases)

Password Strength (AX-3.8)

Starting in AX-3.8, the definition of strong passwords for station users is configurable in each station. A new “Password Strength” container slot in the **UserService** holds configuration properties you can adjust as needed. This slot effectively replaces the former “**Require Strong Passwords**” slot.

Figure 6-24 Password Strength properties in AX-3.8 UserService property sheet



Password strength values shown above reflect the “default” strong password rules, as enforced by the “**New Station**” wizard in Workbench (from menu bar, **Tools** > **New Station**). These *defaults* are a minimum 10 character password, using at least 1 upper case, 1 lower case, and 1 digit (numeral), but no special characters. This wizard does not complete without a valid password for the “admin” user.

Once the wizard completes, you can adjust the station’s password strength properties as needed. If changed in a station, any future password change for any station user (including the “admin” user) requires the minimum values specified in those properties.

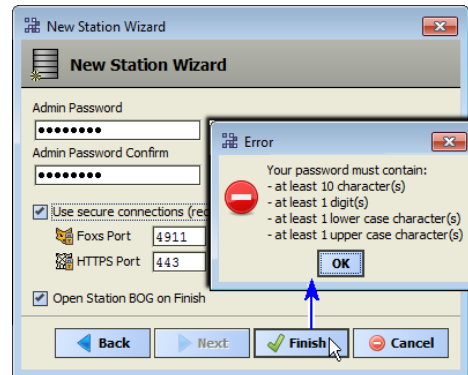
Password Strength properties are as follows:

- **Minimum Length**
Minimum total required characters for a station password, where 10 is the default.
- **Minimum Lower Case**
Minimum alphabetic lower case characters (a-z) in a station password, where 1 is the default.
- **Minimum Upper Case**
Minimum alphabetic upper case characters (A-Z) in a station password, where 1 is the default.
- **Minimum Digits**
Minimum numerals (0-9) in a station password, where 1 is the default.
- **Minimum Special**
Minimum “non-alphanumeric” characters in a station password, where 0 (none) is the default. This includes punctuation and symbols (e.g. among others “!”, “@”, “#”, “\$”, “%”, “&”, space).

Note: Although “Password Strength” properties allow reducing password strength (e.g. entering 0s in values), it is strongly recommended to retain a level of password strength similar to the “default” level, if not greater. For example, you may wish to require at least one (1) “special” and at least two (2) upper case characters. For related details, see “Stronger passwords” on page 6-28.

Password entry dialog improvements Starting in AX-3.8, improvements in station password entry dialogs were made that correspond to a station’s configured “Password Strength” criteria. Examples include error popups that can appear when using the “New Station” wizard in Workbench (when specifying the password for the built-in “admin” user), as shown in [Figure 6-25](#).

Figure 6-25 Example error dialog in AX-3.8 Workbench “New Station” wizard




Similar error dialogs can appear when changing a user’s password when online with the station, whether using Workbench or browser access. If you specify a station’s Password Strength from defaults, note that the station’s current password strength criteria appears in these type of error popups.

The needed password criteria also applies to any user with a subsequent *password reset* or expired password.

For example, say you *modify* “Password Strength” properties from defaults to require a minimum of 12 characters, 2 upper case characters, and 1 special character. You then set a user’s password to reset at next login (Password Configuration, Force Reset At Next Login = true).

When that user accesses the system, after entering their current credentials they see the new password requirements, as shown in [Figure 6-26](#).

Figure 6-26 Example password strength criteria seen when user must reset their password (AX-3.8)

Browser access example	Workbench access example
<p>Password Reset You are required to reset your password before continuing. Please re-enter your current username and password, then enter your new password and confirm.</p> <p>MyNew38Sta2</p> <p> Username: <input type="text"/></p> <p>Old Password: <input type="password"/></p> <p>Your password must contain:</p> <ul style="list-style-type: none"> - at least 12 character(s) - at least 1 digit(s) - at least 1 lower case character(s) - at least 2 upper case character(s) - at least 1 special character(s) <p>New Password: <input type="password"/></p> <p>Retype Password: <input type="password"/></p> <p>Login</p>	<p>Reset Password Set new account password</p> <p>Please enter a new password. Your password must contain:</p> <ul style="list-style-type: none"> - at least 12 character(s) - at least 1 digit(s) - at least 1 lower case character(s) - at least 2 upper case character(s) - at least 1 special character(s) <p>Password <input type="password"/></p> <p>Confirm <input type="password"/></p> <p>OK Cancel</p>

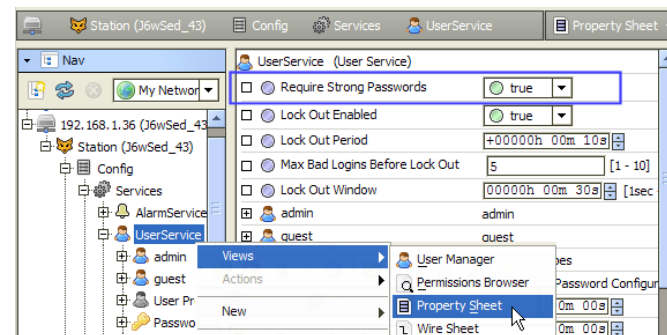
For related details, see the following sections:

- “About password expiration and reset” on page 6-18

Require Strong Passwords (AX-3.7)

In AX-3.7 (and earlier NiagaraAX releases), the UserService has a single Boolean property to either enable or disable requiring strong passwords for station users, as shown in Figure 6-27.

Figure 6-27 Require Strong Passwords property in AX-3.7 station's UserService



In AX-3.7, the default value for this property has always been true (and is *strongly recommended*). Note the NiagaraAX definition of “strong passwords” is fixed in AX-3.7 (and earlier releases), where a station user’s strong password must meet these barest minimum requirements:

- Minimum of 8 characters
 - Characters can include letters (a—z, A—Z), digits (numerals 0—9), and symbols (e.g. among others “!”, “@”, “#”, “\$”, “%”, “&”, space), but *cannot be either*:
 - all letters (e.g. abcdefgh or BadIdeas *do not* qualify), or
 - all digits (e.g. 12345678 *does not* qualify)
- For example, the following passwords meet *minimum* requirements:
- abcd1234 (contains letters and digits)
 - abcdefg\$ (contains letters and symbol)
 - !2345678 (contains symbol and digits)
- *Cannot be identical to station name!*

See the “Stronger passwords” section for guidelines on using strong passwords.

If among **UserService** properties, the “Required Strong Passwords” property is true, any subsequently entered station password that does meet the requirements above is not accepted—instead an error dialog appears, explaining a password violation.

Note: Changing this to true does not immediately force users with weak passwords to change to strong passwords. Only when such a user changes their password will a strong password be necessary.

Stronger passwords

Note even “stronger” password guidelines are encouraged (especially in an AX-3.7 station enabled for strong passwords), where the following general password concepts should be followed:

- Use a mix of UPPER and lower case (cAsE SensItiVe) letters.
- Don’t use any part of the user account *name* in a password. For example, if the user account name is ScottF, then ScottF! or ScottF123 are not good ideas (even though the last is considered “strong” in AX-3.7).
- Don’t use birthday year in a password, for example James1971.
- Don’t use password in a password (password1 as one example of a *very* unsafe “strong” password in AX-3.7). Or, Password1234 as an example of an unsafe “default” strong password in AX-3.8.
- Avoid use of dictionary words, as they are commonly used by “brute force” hacking applications.
- Use characters that require typing with both hands, which helps protect against somebody watching you type your password on a keyboard.
- Consider a *string* of words or *nonsensical phrase* that you can easily remember, yet would be difficult to guess. For example: Correct Horse Battery Staple #11

Remember, a good password is easy for a user to remember, yet difficult for an attacker to guess.

User Manager

Double-click the [UserService](#) for the User Manager (default view when you have admin-Write access), as shown in [Figure 6-28](#).

Figure 6-28 UserManager is table based view

Name	Full Name	Enabled	Expiration	Network Use	Protot	Lang	Email	Default V
admin	Administrator	true	Never	false				Default V
guest		false	Never	false				Default V
m2m	Machine To Machine	true	Never	false				Default V
Mobile	Mobile Demo	true	Never	false				Default M
Demo	Demo Standard	true	Never	false				Basic Wb
TempUser	Temporary User	true	Expired	false				Simple Ac
ScottF	Scott Free	true	Never	false			sfree@mx	Default V
FacMan	Facility Manager	true	Never	false			facman@	Default V
Operator	Operator User	true	Never	false			operator@	Default V

Rows in the User Manager table represent existing users, where you can edit by either double-clicking a single user, or clicking to select one or more users and then clicking the **Edit** button.

Note: By default, a row for a disabled user is colored grey; for an “expired” user is colored orange; and for a “locked out” user is colored red.

As in other table-based manager views, you can select *multiple* rows (users) for edit. This can be useful for a mass change to a property like Facets or Web Profile (or most recently, “Force Reset Password”). However, be careful when doing multiple user edits of things like Permissions and Passwords.

See subsections “[Lockout notes](#)” on page 6-29 and “[Network user notes](#)” on page 6-29 for additional information about working in a station’s User Manager.

To add a new user, click the **New** button. This produces the New User dialog, as shown in [Figure 6-29](#).

Figure 6-29 New (user) dialog

You can create multiple users by typing in a number in “Number to Add.” When you click **OK**, the Add dialog includes that number of user rows in the top table ([Figure 6-30](#)).

Figure 6-30 Adding multiple users

Name	Full Name	Enabled	Expiration	Permissions	Network User	Prototype Name	Language	Password
ChrisP	Chris Peterson	true	Never		false			--password--
User1		true	Never		false			--password--
User2		true	Never		false			--password--

☐ Name: ChrisP
☐ Full Name: Chris Peterson
☐ Enabled: true
☐ Expiration: Never Expires
☐ Permissions: Super User (access entire station, file system)

As needed, click to highlight a single user row before entering unique user properties such as User Name, Full Name, Permissions, and so on. For properties you wish to enter identically for all users (for example, Facets), hold down Shift and click to select all user rows, then enter property value.

Note: For a listing of all the User properties, see “User” on page 6-30. Note two new User properties (in AX-3.7) relate to password handling. See “About password expiration and reset” on page 6-18 for related details. When you click **OK**, the user(s) are added with the property values you entered. Users appear as rows in the User Manager, and as child components under the UserService container.

Lockout notes

If you have lockout enabled in the UserService properties, a user will be locked out after several unsuccessful attempts to login (login using their user name, but with incorrect password). A locked-out user is indicated with a red row in the User Manager, as shown in Figure 6-31.

Figure 6-31 Locked out user is indicated by red

Name	Full Name	Enabled	Expiration	Network User	Protot
admin	Administrator	true	Never	false	
guest		false	Never	false	
m2m	Machine To Machine	true	Never	false	
Mobile	Mobile Demo	true	Never	false	
Demo	Demo Standard	true	Never	false	
ScottF	Scott Free	true	Never	false	
FacMan	Facility Manager	true	Never	false	
Operator	Operator User	true	Never	false	

Note: The User component that represents each user has a boolean “Lock Out” slot that provides read-only status of whether that user account is currently locked out (true) or not (false). If needed, you can link it to other logic in the station.

If locked out, a user will be unable to login into the station until the lockout period expires, as specified in the UserService. Also during a lockout, any attempted login activity for that user will not appear in the station’s AuditHistory (see “AuditHistory station login activities” on page 6-13). After the lockout period expires, the user displays in the User Manager as normal.

As administrator, be aware that each user has an action “Clear Lock Out,” that you can invoke to clear any lockout in progress. Access this command with a right-click on a user, either selected in the Nav tree or in the User Manager, as shown in Figure 6-32.

Figure 6-32 Clear lockout action for any user

Name	Full Name	Enabled	Expiration	Network User	Protot
Mobile	Mobile Demo	true	Never	false	
Demo	Demo Standard	true	Never	false	
ScottF	Scott Free	true	Never	false	
FacMan	Facility Manager	true	Never	false	
Operator	Operator User	true	Never	false	
ChrisP					

Views
 Actions
 New
 Clear Lock Out
 Set Modified

Network user notes

When a network user is replicated (sync’ed) to a “user receiving” station, it is accessible in that station’s User Manager just like any local “non network” user. In earlier NiagaraAX releases, this could be confusing because configuration properties appeared to be writable in user edits from User Manager.

However, any changes made to such a proxied network user (received from another station) were *overwritten* upon the next user synchronization from the source station. Only by looking at the property sheet of an (incoming) network user was it obvious that properties were read-only. This applies to both the source-user derived properties (most properties) as well as the “local override” properties from the (local) [User Prototypes](#).

Starting in AX-3.7, note that proxied network users now appear read-only when accessed from the User Manager (as shown in [Figure 6-33](#)) as well as from their property sheet.

Figure 6-33 Edit dialog from User Manager for a proxied network user (properties read-only)

In addition, note that the *property sheet* for a proxied user includes additional properties (near the bottom) useful in determining the original source of the user. See [Figure 6-34](#).

Figure 6-34 Special status properties in property sheet for proxied network user

- *syncExt* (containing Source Station) -- Shows the ord to the NiagaraStation that proxies the station with the source User.

Note: In previous NiagaraAX builds, an additional “Version” property appeared, now (by default) a hidden slot. It holds the last sync timestamp in milliseconds—which is data used in the synchronization process.

User

All available user properties, at least those seen in the New or Edit user dialog ([Figure 6-5](#) on page 6-5), are described as follows:

- **Name**
Unique name of the user, as known to the station. Used in login credentials (along with Password), also appears in audits of changes made by the user (see [“About user audits”](#) on page 6-12).
- **Full Name**
Full formal name of the user, or any descriptive name needed.
- **Enabled**
Either true (default) or false. Determines if the user account is operational. If disabled (false), login attempts are blocked. By default, user [guest](#) is disabled (and for the best security, should remain so).
- **Expiration**
Either “Never Expires” (default) or “Expires on <date>”. After the expiration date, login attempts are blocked.

- **Permissions**
Either “Super User” (all permissions) or a specific permissions map (click “>>” control). Permissions are category-based. For an overview, see [“Permissions and security”](#) on page 6-8 and also [“About permission levels”](#) on page 6-10.
Note: In general, it is strongly recommended to not assign super user rights. Instead, assign only those permissions to categories needed by a user. Note you can configure a “minimum set” of permissions in the station’s Default Prototype. These will serve as “starting point” set of permissions for any new user you add using the User Manager. For related details, see [“Default Prototype”](#) on page 6-37.
- **Network User**
See the next section [“Network user related properties”](#).
- **Prototype Name**
See the next section [“Network user related properties”](#).
- **Language**
To specify a lexicon (typically two character language code, such as “fr” or “de”), as needed for language support or other customizing.
Note: If customizing lexicons in an English language station, i.e. any U.S. application for example, enter the language code “en” to ensure the WbApplet (for a browser connected user) uses lexicons. Note you may wish to specify this in station’s **Default Prototype**, such that this is the default.
- **Password**
Password entered by user in login credentials (along with user Name). Two entry fields are provided, both entries must match. Can be any combination of alphanumeric characters.
Note: If the UserService is configured for strong passwords (property Require Strong Passwords), the password must be a minimum of 8 characters, using at least two of the three types of characters: either alpha (a–z, A–Z), numeric (0–9) or any other special character, such as “!”, “@”, “#”, “\$”, or “%”. See [“Strong password notes”](#) on page 6-25.
- **Email**
User’s email address. Informational, and also used in EmailService (and/or OnCallService) handling of incoming alarm acknowledgments via the “EmailAlarmAcknowledger” component, to verify the user. For related details, see [“About the Email Alarm Acknowledger”](#) on page 5-15.
- **Cell Phone Number**
User’s cell phone number. Informational, and also used in SMS (Short Message Service) handling of incoming alarm acknowledgments via the “SmsAlarmAcknowledger” component, to verify the user. For related details, see [“About the Sms Alarm Acknowledger”](#) on page 5-22.
- **Facets**
Includes two separate settings, as follows:
 - **Time Format**
How timestamps are formatted in property sheets, and so on. The default time format is sourced from Baja lexicon.
 - **Unit Conversion**
Selectable as either None (default), Metric, or English. Primarily affects value display of “out” slots on control points.
- **Nav File**
To specify a Nav file that provides custom navigation for the user, defining *locator bar* content and home page. Click the folder control for a File Chooser dialog to navigate to the location of the station’s .nav files (by convention, a “Nav” folder under the station directory). For more details, see the *NiagaraAX Graphics Guide* section “About the Nav file”.
- **Default Web Profile**
Includes two settings for station “auto logoff,” as follows:
 - **Auto Logoff Enabled**
Either true (default) or false. If true, a browser-connected user will be automatically disconnected from the station after the “auto logoff period” expires, providing that no user activity was detected during that time (changing views, expanding/contracting containers, and so on). If disabled (false), this browser-connected user is never automatically disconnected.
 - **Auto Logoff Period**
Any number of hours, minutes, seconds needed (default is 15 minutes). Specifies the time period used by the auto logoff routine (“inactive user time”), if auto logoff is enabled for the user.In addition to the two properties above, other Default Web Profile properties specify the user’s expected method and level of HTTP browser access, by “Type”.
 - **Type**
Specifies the user’s expected type and level of HTTP browser access. For more details on profile types, see the *NiagaraAX Graphics Guide* section “About Web Profiles”.

Note: In AX-3.8, any station user connecting via a browser with an assigned Wb Web profile (Web Workbench) requires that browser client PC's Java installation to have Java configured with "Unlimited Strength Policy Files". For details refer to "Additional AX-3.8 client-side Java installation" in the NiagaraAX 2013 Security Updates document.

A few of the available Web Profile types include the following:

- **Basic Hx Profile**
Lowest level and simplest browser requirements (browser with Java plug-in not required). Provides locator bar, ability to access properties and graphics (Px views).
- **Default Hx Profile**
Provides more navigation controls with simple browser requirements (browser with Java plug-in not required). Provides locator bar, access to property sheets and graphics (Px views), plus a view selector. Access to slot sheets is included, as well as PDF creation from property sheets or graphics.
- **Basic Wb Web Profile**
Requires the browser client to have the Java plug-in. Provides Workbench station access within a browser (Fox connection), but without top menu bar or ability for side bars. View access is limited to property sheets and Px views.
Providing that Type is any of the Web Workbench (Wb Web) selections, three additional properties are available. These may be useful in the handling of the Web Workbench applet (WbApplet) in a browser's Java plugin. These properties are listed below.
- **Default Wb Web Profile**
Requires the browser client to have the Java plug-in. Allows nearly full Workbench station access within a browser (Fox connection), including top menu bar and side bars (palettes and Nav tree). View access includes property sheets, wire sheets, category sheets, slot sheets, link sheets and Px views. Additional related properties are listed below.

Note: Prior to AX-3.7, if the same user needed to access the station differently, that is from a simple browser as well as from Workbench or a browser with Java plug-in support, it was recommended to use separate user accounts for that person, each with a different Web Profile. However, starting in AX-3.7 with mobile device support, this may be unnecessary. For related details see the "Mobile Web Profile" property of a User, in the list below.

- Providing that **Type** is any of the Web Workbench (Wb Web) selections, three additional properties are available. These may be useful in the handling of the Web Workbench applet (WbApplet) in a browser's Java plugin. These properties include:
 - **Applet Reload On Hyperlink**
Available only if Type is one of the Workbench ("Wb Web") selections, where the default value is `true`. The WbApplet reloads on a hyperlink—meaning when you change the current Workbench view. If set to `false`, this results in less memory usage by a browser, because the WbApplet is *not* reloaded upon a hyperlink to a new Workbench view. Note the following additional hyperlink behaviors if this property is set to `false`:
 - Just the current Workbench view changes.
 - The browser URL *remains the same*, meaning it does not change.
 - Refresh of the browser takes the user back to their original home page.
 - The **Back** and **Forward** buttons in the browser *no longer function* (as the URL remains the same when hyperlinking to a new Workbench view).
 - Changing Workbench views is much *quicker*!
 - Tunneling is the exception where the WbApplet still reloads.
 - To summarize, there are both positives and negatives regarding this option. To help with one of the less intuitive behaviors, another property ("Show Back and Forward Buttons") can be set from defaults.
 - **Workbench Theme**
(New starting in AX-3.7) Available only if Type is one of the Workbench ("Wb Web") selections, where the default value is `Lucid`. Specifies the Workbench "theme" to use in WbApplet access in the browser. For related details, see ["About Workbench themes"](#) on page 2-32.
 - **Show Back And Forward Buttons**
Available only if Type is one of the Workbench ("Wb Web") selections, where the default value is `false`. If set to `true`, and the user Type is either "Basic Wb Web Profile" or "Default Wb Web Profile", this shows some extra "browser like" Back and Forwards buttons.
 - If "Applet Reload On Hyperlink" is true, the extra back and forward buttons call the browser's back and forward commands. Note that these buttons are never disabled, as the browser's history is unknown to the WbApplet.

- If “Applet Reload On Hyperlink” is false, the extra back and forward buttons call the Workbench history back and forward commands.
As a consequence, this also fixes the kitPx backwards and forwards buttons so each works in a browser (following the above behavior).
Note that for handheld Workbench profiles, the station designer needs to supply their own form of navigation, which could include the backwards and forwards buttons.
- **Mobile Web Profile**
(AX-3.7 and later, and available only if the station’s host is licensed for the mobile feature):
***Note:** Starting in AX-3.7, the station’s WebService has a “ClientEnvironments” container slot with a “Mobile Client Environment” child container with a few properties. This 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 Java-enabled device like a PC, or else “Mobile Web Profile” if a mobile device like a cell phone or tablet).*
- **Type**
Specifies the type and level of HTTP browser access to use for a detected *mobile* client. For more details on profile *types*, see the *NiagaraAX Graphics Guide* section “About Web Profiles”. Current choices include:
 - **Default Hx Profile**
Default mobile profile based on Hx, which does not require the host to have the `mobile` module installed. Other Hx profiles are also available, and may be appropriate in certain cases for tablets or other touch-based applications.
 - **Default Mobile Web Profile**
Mobile profile based on Bajascript, optimized for devices that do not support a Java plugin, but do support HTML5, CSS3, and JavaScript. Often, devices have small displays. When selecting this, the station should have Niagara Mobile Apps in its App container.
***Note:** For detailed information about properties below, refer to the NiagaraAX Mobile Guide section “Common Mobile interface controls and indicators”.*
 - **Theme**
Specifies the “theme” to use when the user accesses the station using a mobile device (e.g. cell phone). Theme is always “mobile” with the following subtypes available:
 - **DefaultjQueryMobileTheme**
 - **DefaultBlueMobileTheme**
 - **LucidMobileTheme**
 - **Show Header**
Default value is `Show`. If set to `Hide` the header area does not appear.
 - **Show Header Back Button**
Default value is `Show`. If set to `Hide` the Back button in the header does not appear.
 - **Show Select Views**
Default value is `Show`. If set to `Hide` selected views do not appear.
 - **Show Home**
Default value is `Show`. If set to `Hide` the Home button does not appear.

Network user related properties

Among [User](#) component properties are two that apply to the “network users” function.

Figure 6-35 User properties related to network user

Name	Full Name	Enabled	Expiration	Permissions	Network User	Prototype Name	Language
NoahF	Noah Fence	true	Never	1=rwRWI;2=rw	true	HvacMgr	

Name: NoahF
Full Name: Noah Fence
Enabled: ☒ true
Expiration: ☒ Never Expires ☐ Expires On 20-Jun-2012 11:59 PM EDT
Permissions: ☐ Super User (access entire station, file system) 1=rwRWI;2=rw;3=rw
Network User: ☒ true
Prototype Name: HvacMgr
Language:
Password: Password: Confirm:
Email: nfence@newmetropolis.net
Cell Phone Number: 8005551234
Facets: Time Format: (default) Unit Conversion: None

Note: These properties are unused for any users in a station with an LDAP user service (e.g. *LdapV3UserService*). These properties are described as follows:

- **Network User**
A boolean that specifies whether this user can be made available in other stations. When using the [User Manager](#) to add new users, this typically defaults to “false” (unless the “User Prototypes > Default Prototype” component has been edited from defaults, where it has been set to “true”. Leave or set to false whenever you wish this user to be local to this particular station only. For an overview of this feature, see “[Network users](#)” on page 6-7.
- **Prototype Name**
Pick from a selection list showing available local User Prototypes. Blank or no selection is effectively the same as the frozen Default Prototype. Currently, this property setting matters only if the Network User property is “true”. See “[About User prototypes](#)” on page 6-36 for related information.

Permissions Browser

From the [User Manager](#), use the view selector to select the Permissions Browser, as shown in [Figure 6-7](#) on page 6-9. By default, the Permissions Browser shows the three object types *collapsed* into just three rows in the tree: station (components), file, and history.

As needed, expand and contract the object tree to find objects that have been explicitly assigned into one or more categories, and scroll across to see each user’s permissions for those objects.

Again, displayed row color provides the same data as in the [Category Browser](#), where:

- Yellow rows are objects explicitly assigned into one or more categories.
- Dimmed rows represent objects *inheriting* their parent’s category or categories.

You can also use [Show Configured](#) to automatically fully expand the tree.

Show Configured

In the Permissions Browser, select **Category Browser > Show Configured** from the menu, or click the icon, as shown as shown in [Figure 6-36](#). The PermissionsBrowser tree expands to show all explicitly assigned objects.

Figure 6-36 Show Configured in PermissionsBrowser

	admin	guest	m2m	Mobile	Demo	ScottF	FacMan	Anital	JoeK	Noah
Config	rwRWI	rwRWI	rwRWI	rw	rwRWI	rwRWI	rwRWI	rwRWI	rwRWI	rwRWI
Files	rwRWI	rwRWI	rwRWI	rw	rwRWI	rwRWI	rwRWI	rwR	rwR	rwR
History	rwRWI	rwRWI	rwRWI	rw	rwRWI	rwRWI	rwRWI	rwR	rwR	rwR

	admin	guest	m2m	Mobile	Demo	ScottF	FacMan	Anital	JoeK	Noah
Config	rwRWI	rwRWI	rwRWI	rw	rwRWI	rwRWI	rwRWI	rwRWI	rwRWI	rwRWI
Services	rwRWI	rwRWI	rwRWI	rw	rwR	rwRWI	rw	rw	rw	rw
AlarmService	rwRWI	rwRWI	rwRWI	rw	rwR	rwRWI	rw	rw	rw	rw
UserService	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
admin	rwRWI	rwRWI	rwRWI	rw	rwR	rwRWI	rw	rw	rw	rw
guest	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
User Prototypes	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
Password Configuration	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
m2m	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
Mobile	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
Demo	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
ScottF	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
FacMan	rwRWI	rwRWI	rwRWI	rw	rwR	rwRWI	rw	rw	rw	rw
Anital	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
JoeK	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
NoahF	rwRWI	rwRWI	rwRWI	rw	rwR	rw	rw	rw	rw	rw
CategoryService	rwRWI	rwRWI	rwRWI	rw	rwR	rwRWI	rw	rw	rw	rw

In each cell intersecting a user (column) and object (row), that user's permissions are shown with abbreviations—see [Permissions abbreviations](#). For any user, you can [access permissions](#).

Permissions abbreviations

Permissions for each object are shown in each of the user columns using abbreviations ([Figure 6-36](#)).

Figure 6-37 Permission abbreviations

	admin	quest	ScottF	FacMan	TempUser
HousingUnit	rwRWI	r	rwR	rwRWI	rwRWI
AirHandler	rwRWI	r	rwR	rwRWI	rwRWI
Lighting	rwRWI	r	rwR	rwRWI	rwRWI
Floor1	rwRWI	r	rwR	rwRWI	rwRWI
Floor2	rwRWI	r	rwR	rwRWI	rwRWI
Floor3	rwRWI	r	rwR	rwRWI	rwRWI
Floor4	rwRWI	r	rwR	rwRWI	rwRWI

	admin	quest	ScottF	FacMan	TempUser
HousingUnit	rwRWI	r	rwR	rwRWI	rwRWI
AirHandler	rwRWI	r	rwR	rwRWI	rwRWI
Lighting	rwRWI	r	rwR	rwRWI	rwRWI
Floor1	rwRWI	r	rwR	rwRWI	rwRWI
Floor2	rwRWI	r	rwR	rwRWI	rwRWI
Floor3	rwRWI	r	rwR	rwRWI	rwRWI
Floor4	rwRWI	r	rwR	rwRWI	rwRWI

Full permissions are “rwRWI”, where the capital RWI is for admin-level Read, write, Invoke. It is possible for a user to have no permissions for some objects (blank cell), if those objects are assigned to a category the user does not even have operator read “r” permissions for. See [“About permission levels”](#) on page 6-10 for more details.

Access permissions

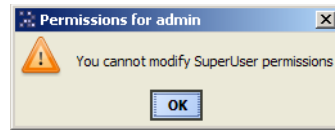
Double-click anywhere in a column to access a user's permissions. This produces the same Permissions dialog as from the user's properties, only user name shows in the window title bar ([Figure 6-38](#)).

Figure 6-38 Permissions map from PermissionsManager

Category	Operator			Admin		
	R	W	I	R	W	I
User	✓	✓	✓	✓	✓	✓
Admin	✓	✓	✓	✓	✓	✓
History	✓	✓	✓	✓	✓	✓
Lighting	✓	✓	✓	✓	✓	✓
AlarmConsole	✓	✓	✓	✓	✓	✓
Playground	✓	✓	✓	✓	✓	✓
Category 7	✓	✓	✓	✓	✓	✓
Category 8	✓	✓	✓	✓	✓	✓

Note: If you double-click on any user with “super user” rights, a popup warning reminds you that you cannot modify permissions, as shown in [Figure 6-39](#).

Figure 6-39 Super user permissions warning



If you *could* see the permission map for any super user, it would simply be all possible permissions checked for all possible [categories](#) (rows).


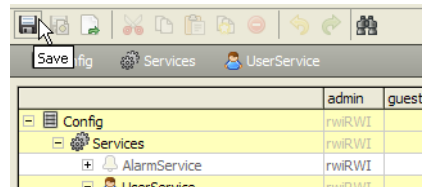
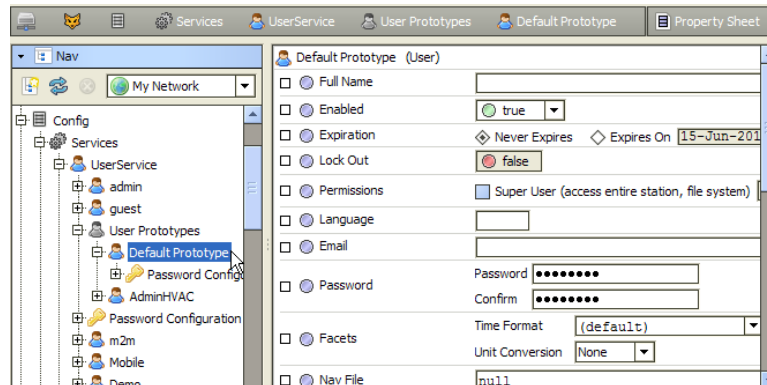
As shown in [Figure 6-40](#), remember to  **Save** after making any user permissions changes.

Figure 6-40 Save user permissions changes



About User prototypes

Figure 6-41 User Prototypes under UserService



See the following about User Prototypes in a station's UserService.

- [Properties of User Prototypes](#)
- [Default Prototype](#) (important in *any* station)
- [Additional \(non-default\) User Prototypes](#)
- [Specifying additional “local override” properties](#)
- [Naming User Prototypes](#)

Properties of User Prototypes

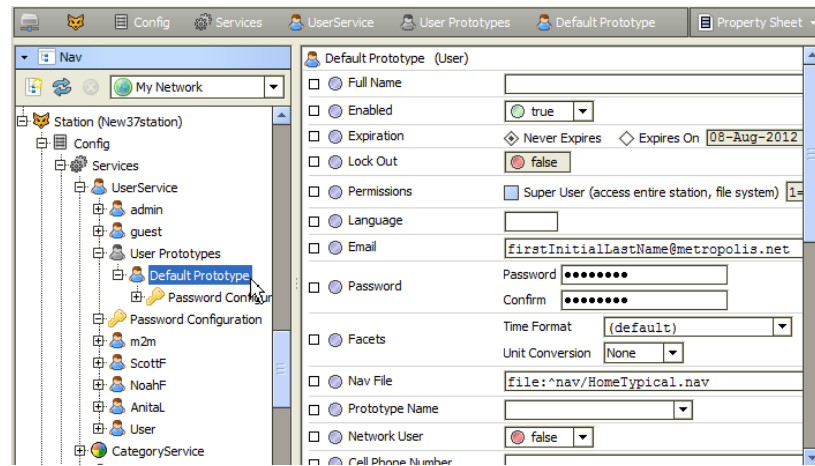
User prototypes under a station's [UserService](#) have the same properties as User components (see the section “User” on page 6-30 for User property descriptions). The importance of user prototypes can be divided between the frozen “[Default Prototype](#)”, and any [Additional \(non-default\) User Prototypes](#) you add, for example by duplicating and renaming. Currently, non-default user prototypes are only used when synchronizing network users between different stations. In this case, (identically-named) prototypes in both the source station and receiving station can be used in a “sync strategy” of “Prototype Required.”

Note the alternative LDAP user services, for example **LdapV3ADUserService** or **ActiveDirectoryUserService**, also leverage user prototypes. When using one of these other user services (in place of the standard **UserService**), you also create user prototypes. However, operation differs from the “network user” usage under the UserService. For details refer to “Mapping of permissions and other preferences to LDAP users” in the *NiagaraAX LDAP / Active Directory Configuration Guide*.

Default Prototype

Among [User Prototypes](#), the “Default Prototype” is *important in any station* with any user service, as it is always the default source of User property values whenever you use the service’s **User Manager** view to add a *New user* to the station.

Figure 6-42 Default Prototype of the UserService’s Default Prototypes is source of default user properties



In this regard, all of the **Default Prototype** properties can be considered important—with one exception: password—which is *not* copied up to a new user created in the **User Manager**.

However, all other property values (except password) in the **Default Prototype** are used as “defaults” when you create any new user in the **User Manager**. This can be useful, for example, if you have some (minimum) collection of permissions for all users, or a typical Nav file, and so on.

Note: Starting in AX-3.7, User Prototypes have a child “Password Configuration” container, just like User components. Inside are two properties as follows:

- **Force Reset At Next Login** - The default is “false” in AX-3.7u1 or later, for any new station (however, in the initial AX-3.7 release the default was “true”). If you find yourself typically changing this each time you create a new User, change it to desired value in the **Default Prototype**.
- **Expiration** - Default value is “Never expires”. Typically you leave this at default. However, it is possible you might change this to some “far future” date in the **Default Prototype**.

For more details on the operation and configuration of these two properties, see [“About password expiration and reset”](#) on page 6-18.

Default Prototype importance in Network users scenario

Additionally, in a multi-station job where you are using “Network users”, the user “sync strategy” of “Use Default Prototype,” the default prototype (in each station receiving network users) specifies the “local properties” that override the received properties of any network user. Note that by default only 2 properties are “local override” types: Permissions, and Nav File.

However, by going to the slot sheet of the Default Prototype (in each station *receiving* network users), and setting the “user defined 1” config flag, you can specify *additional* properties as “local override” types. You can also use this same technique with other (non-default) User Prototypes in stations that receive network users. For details, see [“Specifying additional “local override” properties”](#) on page 6-38.

Additional (non-default) User Prototypes

The importance of properties in an additional (non-default) [User Prototypes](#) vary among stations that are either sending or receiving network users:

- In a user-sending station (e.g. Supervisor) non-default user prototypes are important only in “name”, where you can simply duplicate the “Default Prototype” and rename each duplicate uniquely. Property values in these replicated prototypes are *not used in any users*—whether a user is local only to the Supervisor, or specified as a network user with this “Prototype Name.”
- In a user-receiving station (e.g. JACE) non-default user prototypes are important both in “name”, where matching prototype names in “user sending” stations provide sync strategy options, and also (by default) in two “local override” properties, described below.
 - **Permissions** - (either a permissions matrix of local categories and rights, or a “Super User”)
 - **Nav File** - (referencing a specific nav file under the local station’s file structure)

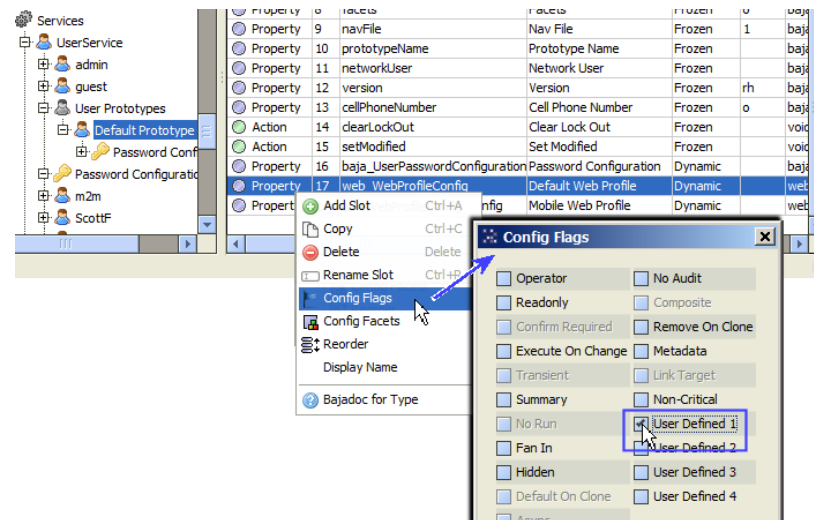
When a network user is added or modified in a “user sending” station, the two properties above are used in the “user receiving” station, instead of those same properties in the source network user. Note prototypes are configurable for other local overrides—see “[Specifying additional “local override” properties](#)”.

Specifying additional “local override” properties

In a “network user receiving” station (e.g. JACE) you can specify other properties of User Prototypes to act as “local overrides” for network users created in its station. This applies both to the single **Default Prototype**, as well as any additional (non-default) User Prototypes.

Do this from the slot sheet of the User Prototype: right-click the property, and select **Config Flags**, as shown being done for the web_WebProfileConfig slot (Default Web Profile) in [Figure 6-43](#).

Figure 6-43 Example config flag being set in the Default Prototype of a “user receiving station”



In the Config Flags dialog, click the “User Defined 1” flag and click **OK**. Notice that in the slot sheet view, the “Flags” column now includes a “1” for that property, similar to the permissions and navFile slots. When a network user sync occurs for a user referencing this prototype, all properties with this flag use the local values as overrides.

Naming User Prototypes

If creating additional [User Prototypes](#) (apart from the “Default Prototype”), it is recommended that you name them using descriptive text that can be logically associated with groups of station users, such as AdminHvac, GenOperations, LtgAndAlarms, and so on. You pick from these names when adding a new user and selecting a “Prototype Name.”

Keep in mind that in a multi-station job, if you choose a network “sync strategy” based upon the “Prototype Required” scheme, network users in the “user sending” station are replicated/sync'ed in the “user receiving” stations *only* if the UserService in each remote station has an *identically named* user prototype. Note there is also an alternative “Use Default Prototype” sync strategy you can use instead.

For related details, refer to the “About Users sync strategy” section in the *Drivers Guide*.

CategoryService

The CategoryService is found under a station’s Services container. You use a station’s CategoryService to add [categories](#) and assign objects (components, files, and histories) to those categories.

Note: For an overview of station security including categories, see “[Security model overview](#)” on page 6-2, and “[Categories and security](#)” on page 6-3.

Apart from being the container for child categories, the CategoryService has only two slots:

- **Update Time**
A property that sets the interval at which [ancestor permissions](#) are automatically assigned. Default value is one (1) minute. If assigned a zero value, this feature is disabled.
- **Update**
An action to manually force ancestor permissions to be assigned to objects in the station.

Note: Typically, you leave the service’s update time at default, and use the update action only if needed while testing (while actively assigning categories and/or changing permissions).

The two unique views of the CategoryService are:

- [Category Manager](#)
- [Category Browser](#)

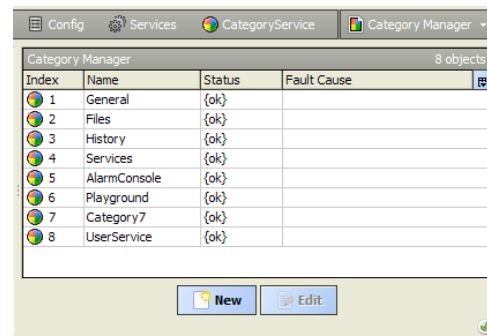
The CategoryBrowser is the default view, and typically where you spend more time after initially creating categories.

Note: In the case of components, you can also make category assignments directly from any component's CategorySheet view. See [“Category Sheet”](#) on page 6-41.

Category Manager

You use the Category Manager to add, edit, or delete categories. As shown in [Figure 6-2](#) on page 6-3, you can access the Category Manager using the view selector from any [CategoryService](#) view.

Figure 6-44 Category Manager is table based view



Index	Name	Status	Fault Cause
1	General	{ok}	
2	Files	{ok}	
3	History	{ok}	
4	Services	{ok}	
5	AlarmConsole	{ok}	
6	Playground	{ok}	
7	Category7	{ok}	
8	UserService	{ok}	

Buttons: New, Edit

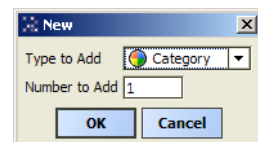
Rows in the CategoryManager table ([Figure 6-44](#)) represent existing categories, where you can edit by double-clicking one. Typically, you edit only the category *name*, to clarify some logical grouping. All categories must have both a unique name and index.

Note: Starting in AX-3.7, the Workbench tool **New Station Wizard** creates a station with two default category components, for category indexes 1 (named “User”) and 2 (named “Admin”). Formerly, no category components were created, and all objects defaulted to category index 1.

In addition, while most components and all history objects still default to index 1 (“User”), selected components (UserService, CategoryService, and ProgramService) and the entire station File space now default to category index 2 (“Admin”). This “two category model” can be expanded by adding more categories.

To add a new category, click the **New** button. This produces the New dialog, as shown in [Figure 6-45](#).

Figure 6-45 New (category) dialog



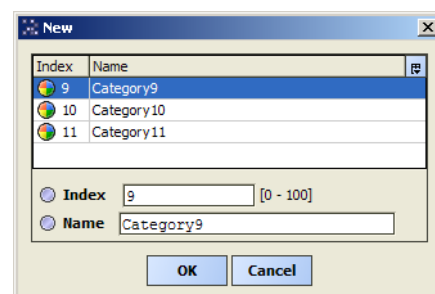
Type to Add:

Number to Add:

Buttons: OK, Cancel

You can create multiple categories by typing in a number in “Number to Add.” When you click **OK**, the Add dialog includes that number of category rows in the top table ([Figure 6-46](#)).

Figure 6-46 Adding multiple categories



Index	Name
9	Category9
10	Category10
11	Category11

Buttons: OK, Cancel

Index: [0 - 100]

Name:

As needed, click to highlight a single category row before entering a unique Name. For a listing of all the category properties, see [“Category properties”](#) on page 6-40.

When you click **OK**, the categories are added with the names you entered, and appear as rows in the CategoryManager and as child components under the CategoryService container.

Note: *While a maximum of 100 categories could be created, in general it is recommended to limit the number of categories to as few logical divisions as needed. Included are performance reasons, as well as other considerations. See “Category caveats” on page 6-40.*

Category properties

As shown in the New or Edit dialog (Figure 6-46), there are 2 main properties for a category, as follows:

- **Index**
Unique index number for the category, as it is known to the station.
- **Name**
Any descriptive name needed, typically to reflect some logical grouping.

An additional property is available on each category's property sheet:

- **Fault Cause**
Read-only text descriptor to describes why a category is in fault, if applicable (typically blank).

Category caveats

Please note the following about using categories:

- Performance-wise, too many categories can consume excess station memory, as each object must maintain a bitmap for category membership. With the default 8 categories, this is only 1 byte, but increments another byte for each additional 8 categories added (9—16, then 17—24, and so on). Therefore, you should limit categories to the fewest needed, and keep them indexed contiguously.
- If you delete a category (using the [Category Manager](#)), you may notice it still appears listed in users' permission maps, with default “Category *n*” name instead of its former user-assigned name. This underscores that categories are simply indices to the station. Each category Name is simply “metadata” (data about data).
Moreover, when you assign an component to one or more categories (e.g. using the [Category Browser](#)), that component's category bitmap is updated—as part of its configuration. If you copy that component to another station or save it in a bog for reuse, please note that its category bitmap (to category *indices*) is *included*.

Category Browser

The Category Browser is the default view of the [CategoryService](#). By default, this view shows the three object types *collapsed* into just three rows in the tree: Config (components), File, and History. In this view, the *columns* represent [categories](#) in the station.

Again, displayed row color provides the same data as in the [Permissions Browser](#), where:

- Yellow rows are objects explicitly assigned into one or more categories.
- Dimmed rows represent objects *inheriting* their parent's category or categories.

To automatically fully expand the tree, use [Show Configured](#).

Show Configured


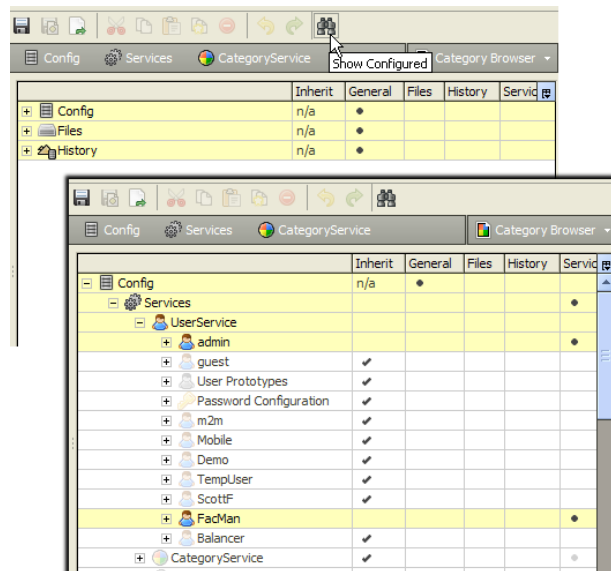
Select **Category Browser > Show Configured** from the menu, or click the  icon, as shown in [Figure 6-47](#). The tree expands to show all objects explicitly assigned into one or more categories.

Figure 6-47 Show Configured in CategoryBrowser



Objects in dimmed rows have a check mark in the first column (Inherit), meaning that they *inherit* the category or categories of their parent container.

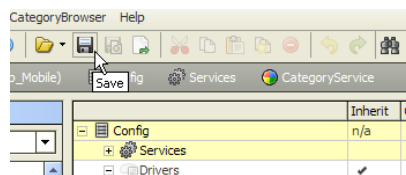
As needed, in any object row, click either:

- In any category column to explicitly assign an object to a category, or again (toggle) to remove the object from that category.
- In the Inherit column to return category assignments to match the object's parent.

Note: With the exception of the three “root” objects (Config, Files, History), each object must either belong to at least one category or inherit its parent's category assignments. The three root objects cannot inherit—they must belong to one or more categories.

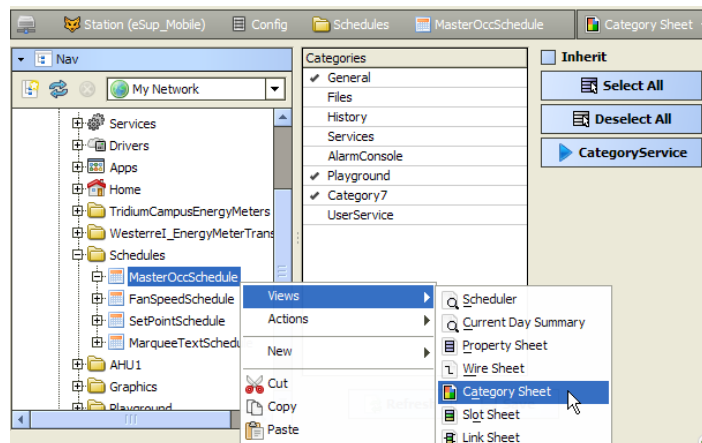
As shown in [Figure 6-48](#), remember to **Save** after making any needed category assignment changes.

Figure 6-48 Save category assignment changes



Category Sheet

Included for every component in the station database is a Category Sheet view, which you can access using the right-click **View** menu in the Nav tree ([Figure 6-49](#)), or by using the view selector.

Figure 6-49 Accessing CategorySheet view

This view lists all categories in the station, and shows a check mark beside any that are assigned to this component.

In addition, the view's right side offers various controls, as follows:

- **Inherit (checkbox)**
Click to toggle. While checked, this component inherits category assignments from its parent component. While cleared, you can make explicit category assignments for this component.
- **Select All**
Effective if **Inherit** is cleared. Click to assign component to *all categories* in the station.
- **Deselect All**
Effective if **Inherit** is cleared. Click to remove all category assignments for this component.
Note: Either click again to assign the component to at least one category, or else click **Inherit**. Otherwise, you receive an error when you try to Save (must be assigned to one category).
- **Category Service**
Click to go to the CategoryBrowser view. See “[Category Browser](#)” on page 6-40.

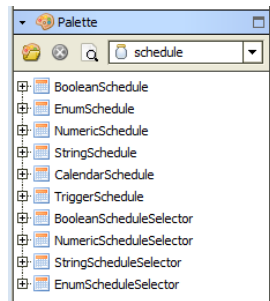
Note: Remember to click **Save** after making any needed category assignment changes.

CHAPTER 7

About Scheduling

Schedules in NiagaraAX are done using schedule *components*, as found in the `schedule` palette (Figure 7-1). You place these components in a station, configure, and link as needed to provide scheduling control of other components.

Figure 7-1 Schedule palette



Each schedule component has a “scheduler” view, which you use to define events. The following main sections provide more details:

- [“About the scheduling model”](#) on page 7-1
- [“About weekly schedules”](#) on page 7-5
- [“About calendar schedules \(holidays\)”](#) on page 7-18
- [“About trigger schedules”](#) on page 7-23
- [“About ScheduleSelector components”](#) on page 7-27
- [“Using schedules”](#) on page 7-29 (schedule-related tasks)

About the scheduling model

NiagaraAX scheduling is summarized in the following sections:

- [“Schedule component categories”](#) on page 7-1
- [“Schedule component views”](#) on page 7-2
- [“Schedule component links”](#) on page 7-2
- [“Schedule special events”](#) on page 7-3
- [“Schedule exports and imports \(master/slave\)”](#) on page 7-4

Schedule component categories

Schedule components may be categorized as follows:

- **Weekly schedules**
These schedules define regular, repeating, events by “time-of-day” and “day-of-week.” Also, any number of “special events” are configurable. Typically, these are the *most used* schedule components. Four different *types* vary by data category (Boolean, Numeric, Enum, and String). Each is identical except for input/output. For more details, see [“About weekly schedules”](#) on page 7-5.
- **Calendar schedule**
A CalendarSchedule component is available to define specific *days*. Typically, you use them to define days with scheduling exceptions (e.g. *holidays*), and *reference* them in the “special events” setup of weekly schedules. For details, see [“About calendar schedules \(holidays\)”](#) on page 7-18.

- **Trigger schedule**
ScheduleSelector components provide an easy way for users to select a schedule to use for controlling a particular component. As needed, you typically link its output to an *action* of a control point or extension. For more details, see “About trigger schedules” on page 7-23.
- **Schedule selectors**
A TriggerSchedule component is used to schedule the firing of topics. With a set of pre-configured schedules and the ScheduleSelector component, you can simply choose from a list of valid schedules to setup or change the schedule of a device. For more details, see “About trigger schedules” on page 7-23.

Schedule component views

The default view for any schedule component is its “scheduler” view, where you define related days and events.

- The [Weekly Scheduler view](#) for any *weekly* schedule component provides four *tabs*, as follows:
 - Weekly Schedule — Specifies Sunday-through-Saturday (weekly) event times/values.
 - Special Events — All exceptions to the defined weekly schedule, as special events.
 - Properties — Important properties such as default output, schedule effective times, special event cleanup operation, and schedule facets.
 - Summary — For any selected day, provides summary of all schedule events, with source.
 In addition, any weekly schedule component provides a “Current Day Summary” view, providing a simple *linear* 24-hour graph of schedule event times and values for the current day.
- The [Calendar Scheduler view](#) for a CalendarSchedule provides a “calendar” where you add, edit, or delete calendar days and establish relative priorities.
- The [Trigger Scheduler view](#) for a TriggerSchedule provides a two-part view combining a “day picker” and an event “time picker” for specifying when topics are fired on those days.

At the bottom of *any scheduler view* are buttons [Refresh](#) and [Save](#), described as follows:

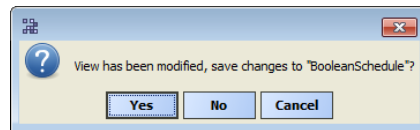
Refresh

The **Refresh** button is always available.

When you click Refresh, one of two things happens:

- If the Save button is not available (no unsaved changes), clicking it re-synchronizes the view with the component’s current configuration.
- If the Save button is available (unsaved changes), clicking it produces a dialog ([Figure 7-2](#)).

Figure 7-2 Refresh confirmation if unsaved changes



Your refresh confirmation dialog choices are as follows:

- **Yes**
Save all changes made in the view since last save. Equivalent to clicking [Save](#).
- **No**
Clear all changes made in the view *since last save* (effective *reset*).
- **Cancel**
Cancels refresh, all unsaved changes remain as unsaved.

Save

The **Save** button is available only if you have made unsaved changes in the component’s scheduler. Clicking it downloads your changes to the schedule component’s configuration. Immediately following, the Save button is unavailable again.

Note: For weekly schedules (tabbed [Weekly Scheduler view](#)), you typically save while working in each tab, even though any save applies to changes made on all tabs.

Schedule component links

To use weekly schedules and the trigger schedule, you link the component’s “Out” slot (as source) to a slot on another component. The same schedule can be linked this way to *many* target components.

- [Weekly schedule links](#)
- [Trigger schedule links](#)

Typically, you *do not link* a CalendarSchedule. Instead, you *reference* one or more CalendarSchedules from a weekly schedule, in its “special events” setup. This allows global editing of day definitions.

Weekly schedule links

Typically, you link a weekly schedule to one or more “writable” control points. For some examples, you could link the Out slot of weekly schedule types as follows:

- BooleanSchedule to a BooleanWritable that is a proxy point for a Binary Output object in a BACnet device.
 - NumericSchedule to a NumericWritable that is a proxy point for setpoint NVI in a LON device.
- By *convention*, when linking to a target writable point (with 16-level priority array), you select “In16” among its different priority array inputs. However, you are free to select any available input level.

In a few cases, you may wish to “chain” weekly schedules from “Out” slot to “In” slot. This technique is typically useful only if *one* of the chained schedules is “effective” during any period in time. In this case, the [Default Output](#) value of all schedule components (except last in chain) must be *null*.

Trigger schedule links

Typically, you link a trigger schedule to an *action* of a control point or perhaps even more often, an action of a point *extension*. For example, you could link a TriggerSchedule to the “ResetElapsedActiveTime” action of a DiscreteTotalizerExt, a point extension for a BooleanPoint used to accumulate runtime. If the trigger schedule was configured to fire *only* on the first day of every month (at 12:00am) that extension could be used to hold the current month’s runtime.

Schedule special events

Special events apply to *weekly schedules* only, and are considered any exception to the (normal) weekly schedule. Special events can be “one-time” only event changes *or* recurring event changes, such as *holidays*. Configuration includes both day(s) of occurrence and related time-of-day events.

In the time-of-day event definitions of special events, you can have them “intermingle” with regular weekly events, or completely override the weekly schedule. In addition, you visually prioritize special events, via list order. This allows any overlapping special events to occur in an ordered fashion.

Each weekly schedule component has its *own* special events, configured on a [Special Events](#) tab in its scheduler view. Event times (and values) entered for any special event apply to that schedule only.

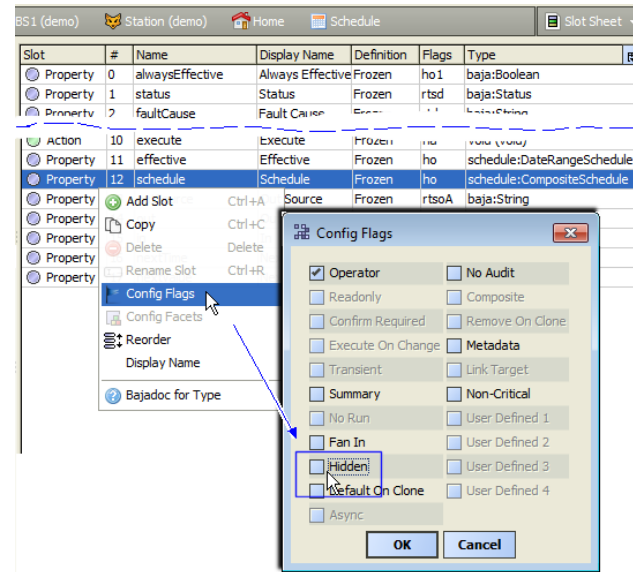
Note: *If the special event is a “reference” type, days of its occurrence are specified in the CalendarSchedule component that is referenced. This allows you to globally change the days that special events occur in weekly schedules, by editing one or more referenced CalendarSchedules. For more details, see “Calendar-Schedule Usage” on page 7-18.*

Special events permission change

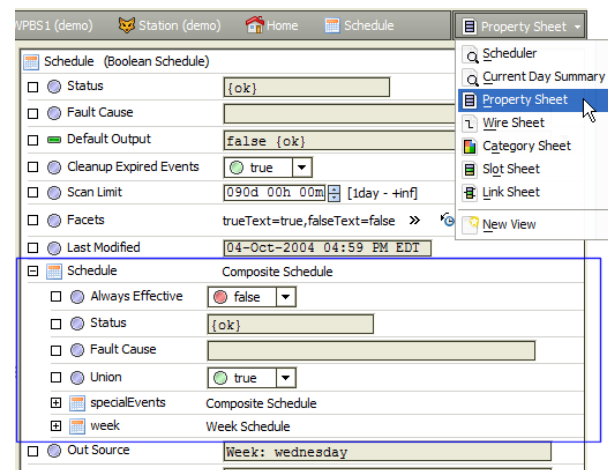
If needed, you can provide different permissions for the *special events* of a weekly schedule (BooleanSchedule, NumericSchedule, etc.) than for the rest of the schedule’s configuration, using advanced techniques. In early NiagaraAX releases, *operator Write* access for the schedule was required to have write ability on *any* of the four tabs in its [Weekly Scheduler view](#) (including Special Events).

Note: *There is also now automatic “read-only” access to all tabs of the Weekly Scheduler view for a user with only “operator Read” permissions on a schedule. Previously, such a user could only see the [Summary](#) tab.*

To configure a schedule to allow special events access separately, you must “unhide” its “Composite-Schedule” slot (named Schedule) working from the schedule component’s *slot sheet* ([Figure 7-3](#)).

Figure 7-3 Slot sheet of schedule component

Once the child `CompositeSchedule` is unhidden, you can expand it in the schedule's property sheet (Figure 7-4).

Figure 7-4 Expand unhidden `CompositeSchedule` on schedule's property sheet

As shown, this “Schedule” slot contains *child slots*, including a “specialEvents” schedule and a “week” schedule. Now, you can assign the child “specialEvents” `CompositeSchedule` to a *different category* that the top-level schedule (right-click the “specialEvents” slot, then choose **Views > Category Sheet**).

In this case, you would deselect the “Inherit” option and clear that category, assigning a different category—one that a user could have “operator Write” permissions on, versus only “operator Read” as on the parent schedule component. This would allow such a user to have read access to *all* the tabs on this schedule's [Weekly Scheduler view](#), but write access *only* for managing special events—and *not* the regular weekly schedule (or other properties).

For related details on security, including permissions and categories, see [“About Security”](#) on page 6-1.

Schedule exports and imports (master/slave)

Using the NiagaraAX driver architecture, you can create “master/slave” schedules in order to *share* schedule configuration *between devices*. This allows you to globally update the configuration of any slave schedule by simply making changes to its master schedule.

The typical application of this is in a *multi-station* Niagara network, where you:

- *Import* a schedule component *from* another station. Typically, you do this in a JACE station, importing a schedule component that resides in a Supervisor station. This creates a local copy that you can use and link into control logic, but cannot otherwise configure (change events, and so on).

When you import a NiagaraAX schedule, a “schedule export descriptor” is automatically created on the sending (master) station, under the NiagaraStation component that represents the receiving side. This allows for “sending-side” management of configuration synchronization. For more details, refer to the section “Station Schedules import/export notes” in the *Drivers Guide*.

If using the Bacnet driver, the same basic architecture is available. You can import BACnet Schedule and Calendar objects from a BACnet device, and model them as NiagaraAX schedule components.

For more details, see the following topics in the *Drivers Guide*:

- “About the Schedules extension”
- “About Schedules extension views”

Note: The Bacnet driver lets you export NiagaraAX schedules from the station to existing BACnet Schedule and Calendar objects in a BACnet device, acting as the “master” source. Also, you can expose Niagara schedule components as BACnet Schedule or Calendar objects for access by any networked BACnet device. You do this through configuration of the “Export Table” under the BacnetNetwork’s “Local Device.” For more details, see the Bacnet Users Guide.

About weekly schedules

The following sections explain weekly schedule concepts:

- “Types of weekly schedule components” on page 7-5
- “Weekly schedule output processing” on page 7-6
- “Weekly Scheduler view” on page 7-7, which includes four tabs described as follows:
 - “Weekly Schedule” on page 7-7
 - “Special Events” on page 7-8
 - “Properties” on page 7-11
 - “Summary” on page 7-13
- “Event colors in weekly schedules (AX-3.8)” on page 7-14

Types of weekly schedule components

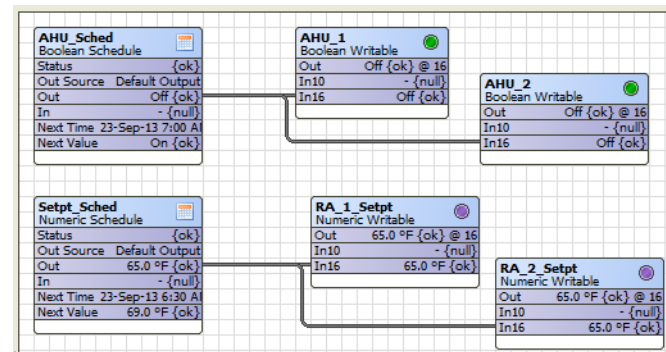
Each component type varies only by data *value category*, meaning its **Out slot** (generally called “output”) and its **In slot** (generally called “input”). For data category concepts, see “About data value categories” on page 3-2. The schedule’s output and input are “Status<Type>” according to schedule component type, for example, StatusBoolean if a BooleanSchedule, StatusEnum if an EnumSchedule.

The four weekly schedule component types are:

- BooleanSchedule
- EnumSchedule
- NumericSchedule
- StringSchedule

In typical use, only *outputs* of schedules are linked to other components for scheduling control, such as “writable” points of the same type. [Figure 7-5](#) shows a BooleanSchedule and NumericSchedule, each linked to two writable points. See “Schedule component links” on page 7-2 for more details.

Figure 7-5 Schedules linked to writable points



About the BooleanSchedule

The BooleanSchedule is typically the *most used* weekly schedule. Use it directly for schedule control of BooleanWritable points (typically proxy points). If needed, it also has application for linkage to slots in *extensions*. For example, using an intermediate kitControl object (say, a “Not” logic-type object) use a BooleanSchedule for linkage to the Enabled slot of an alarm extension.

About the EnumSchedule

The EnumSchedule allows schedule control of EnumWritable points (typically proxy points). For example, link it to an EnumWritable that proxies a BACnet Multistate Output object, or to an EnumWritable that proxies a LON NVI (using an enumerated SNVT).

To properly use an EnumSchedule, first define its [Facets](#) to match those in linked EnumWritable(s).

About the NumericSchedule

The NumericSchedule allows schedule control of NumericWritable points (typically proxy points), which may represent setpoints, limits, or any number of other variables.

About the StringSchedule

The StringSchedule allows schedule control of StringWritable points (possibly proxy points).

Weekly schedule output processing

For any weekly schedule ([BooleanSchedule](#), [EnumSchedule](#), [NumericSchedule](#), [StringSchedule](#)), output recalculation occurs upon any of the following:

- Any saved change to its configuration
- Any change at its input
- Station startup
- Any change to the system clock

Each component has an [Out slot](#) and an [Out Source slot](#). In addition, [Next Time](#) and [Next Value slots](#) are available. By default, all of these slots are “pinned” on the component’s shape on the wire sheet, as shown in [Figure 7-5](#). Upon any output change, all of these slots are updated.

Out slot

A schedule’s *output value* is determined by the following, in highest-to-lowest priority:

1. To any “non-null” value at its In slot (if linked)
This value is immediately passed to its output. Otherwise (if null), processing continues.
2. If the schedule is *not effective*, the output goes to the [default output](#) value.
If the schedule is *effective*, the output goes to the (highest priority) active *special event* (if any).
3. To the active weekly schedule event (if any).
4. To the [default output](#) value.

For more details on effective schedules, see [“Effective Period”](#) on page 7-12.

Out Source slot

Out Source provides a string “source description” of the current output, as one of the following:

- Input
- Special Event: <SpecialEventName>
- Week: <day_of_week>
- Default Output

Examples might appear as “Week: monday” or “Special Event: Christmas Break”

Next Time and Next Value slots

These slots provide “look-ahead” data: the next known schedule output value, and its time of change.

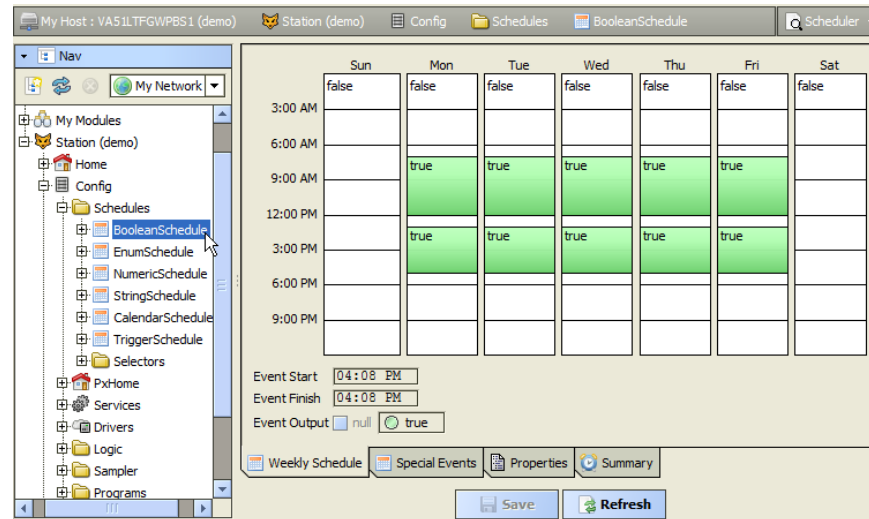
- Next Time — Displays in Baja AbsTime format, for example: 03-Feb-05 5:00 PM
- Next Value — The next scheduled output value.

Typical application is for informational display. If needed, slots can be linked into control logic. For example, TimeDifference and CurrentTime objects (kitControl, Timer) provide AbsTime slots too.

Weekly Scheduler view

Double-clicking any weekly schedule in Workbench produces its Scheduler view (Figure 7-6).

Figure 7-6 Scheduler view for weekly schedule



There are four tabs: **Weekly Schedule**, **Special Events**, **Properties**, and **Summary**, as described ahead.

Note: Buttons **Refresh** and **Save** apply to all tabs in the Scheduler view (not just the one displayed).

Weekly Schedule

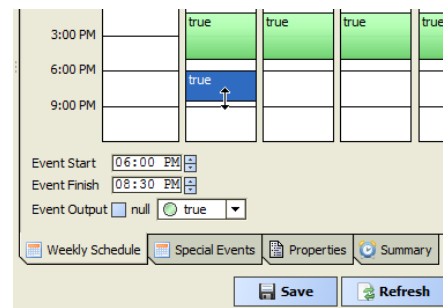
Use this tab in the **Weekly Scheduler view** to enter regular schedule events, that is “normal schedule events” that repeat from week to week, based on the day of the week and the time of day. By default, any existing events appear as colored blocks, while unscheduled (default output) time appears in white.

Note: In AX-3.8, events in *BooleanSchedules* and *EnumSchedules* have specified (and different) colors for each different state value, versus the default “greenish” color seen for all schedule events in AX-3.7 and prior releases. For details see, “*Event colors in weekly schedules (AX-3.8)*” on page 7-14.

Use is mostly straightforward, to add a new event simply click in a day at the approximate event start time, and drag down to define the start and finish time (Figure 7-7). The event remains *selected* (by default, dark blue) when you release the mouse button.

Note: If an *EnumSchedule*, you should define its range facet from the Scheduler’s **Properties** tab before adding events. See “*Facets*” on page 7-12 for more details.

Figure 7-7 Click and drag to enter weekly events



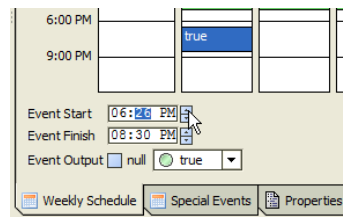
As needed, click again and drag on the event’s top or bottom to change its start or finish time (in broad increments).

Additional details about the Weekly Schedule tab are as follows:

- **Event time “tuning”**
- **Output value**
- **Right-click menus**

Event time “tuning” With any event selected, “fine tune” its start and finish time using the controls, selecting the hours portion or minutes portion (Figure 7-8). Or, click and type values in directly.

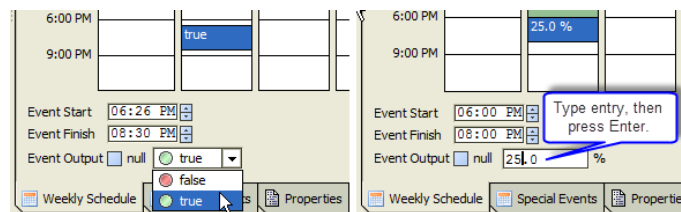
Figure 7-8 Define start and finish time



Note: For any event, start time is inclusive, and the event extends to (but is exclusive of) the end time. In other words, there is no output “blip” between adjacent events, even if across days. For example, if a Monday event ends at midnight, then a Tuesday event starts at midnight, the schedule output is continuous (providing both events have the same [Output value](#)).

Output value For any event, you can select the “null” checkbox (the schedule’s calculated value is null for that event). However, you typically select or type a value instead, as follows:

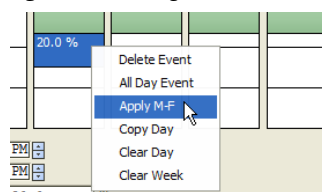
Figure 7-9 Select (Boolean, Enum) or type (Numeric, String) output value



- If a Boolean or EnumSchedule select the event value in the output field, see Figure 7-9, left.
Note: If an EnumSchedule, first specify its facets (on [Properties](#) tab) before entering values. This allows selection of possible values.
- If a NumericSchedule or StringSchedule, you *type* the value in the output field, then press Enter to register it in the event block, as shown in Figure 7-9, right.

Right-click menus Right-click in the weekly schedule area for an event menu. If you have any event *selected*, this menu provides the most commands, as shown in Figure 7-10.

Figure 7-10 Right-click menu with event selected



Event menu options are straightforward, and may include the following:

- Delete Event — Deletes the selected event.
- Paste Day — Appears only if copy day option was used first. Copies all events into selected day.
- All Day Event — Makes currently selected (or last entered) event extend to entire day.
- Apply M-F — Copies all events in the selected day to Mon, Tue, Wed, Thu, and Fri (and *overwrites* any existing events on those days).
- Copy Day — Copies all events in the selected day, to use with paste day option.
- Clear Day — Clears all events in the selected day.
- Clear Week — Clears all events in the entire weekly schedule.

Special Events

Use this [Weekly Scheduler view](#) tab to enter all *exceptions* to the schedule’s weekly schedule, broadly called “special events.” For general information, see “[Schedule special events](#)” on page 7-3.

As shown in Figure 7-11, existing special events (if any) are listed in the table by name and summary. When you select a special event, its day(s) of occurrence are highlighted in the monthly calendars at the top of the view, and its associated event actions are displayed in the right-side column.

Figure 7-11 Special Events tab in weekly Scheduler

The screenshot shows the 'Special Events' tab in the weekly Scheduler. At the top, there are navigation buttons: 'Prev Page', 'Prev Month', 'Today', 'Next Month', and 'Next Page'. Below these are three monthly calendars for September, October, and November 2013. The September calendar has the 18th highlighted in blue and the 30th in green. The October calendar has the 31st highlighted in green. The November calendar has the 30th highlighted in green. Below the calendars is a table with columns for Name, Summary, Cycle, and MonthlyEnd. The 'Name' column contains 'Summary'. The 'Cycle' column contains 'Date Range: 29 Jan - 2 Feb'. The 'MonthlyEnd' column contains 'Custom: Next Event: 30-Sep-13...'. To the right of the table are fields for Event Start, Event Finish, and Event Output. The Event Start field is set to '12:00 PM'. The Event Finish field is set to '05:30 PM'. The Event Output field is set to 'true'. At the bottom are buttons for Add, Edit, Priority, Rename, and Delete.

Additional details about the Special Events tab are as follows:

- [Adding special events](#)
- [Event times and output values](#)
- [Special event priorities](#)
- [Right-click menus and other controls](#)

Adding special events Click the **Add** button to add a new special event. An Add dialog appears, as shown in [Figure 7-12](#).

Figure 7-12 Special Events Add dialog

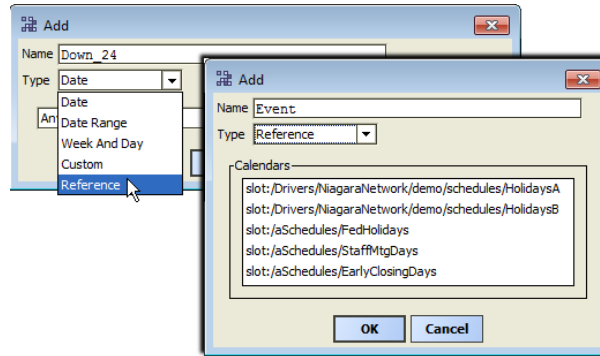
The screenshot shows the 'Add' dialog for special events. The dialog has a title bar with a close button. Inside, there is a 'Name' field with the value 'Event'. Below it is a 'Type' dropdown menu set to 'Date'. Underneath the dropdown are four input fields: 'Any Weekday' (set to 'Any Weekday'), '20' (set to '20'), 'Sep' (set to 'Sep'), and '2013' (set to '2013'). At the bottom are 'OK' and 'Cancel' buttons.

Dialog options are described as follows:

- **Name** — Your descriptive name for special event, perhaps “Christmas_Break” or “Half_Day.” The default value is simply “Event.” You can change this later, if needed.
- **Type** — Determines selection criteria for day or days, with the following choices:
 - **Date** — (default) By various combinations of weekday, numerical date, month or month combinations, and year.
 - **Date Range** — By start and end range, using for each a combination of day, month, year.
 - **Week and Day** — By combination of day of week, week in month, month.
 - **Custom** — By various combinations of day, month, weekdays, and year.
 - **Reference** — By reference to a specific CalendarSchedule component in the station.

For details on most Type selections, see [“Calendar day selections”](#) on page 7-21. If you select type *Reference*, a second Add dialog appears, as shown in [Figure 7-13](#). It lists all CalendarSchedules (Calendars) available in the station, by slot path. Select any one for the day(s) portion of this special event.

Figure 7-13 Reference special event Add dialog

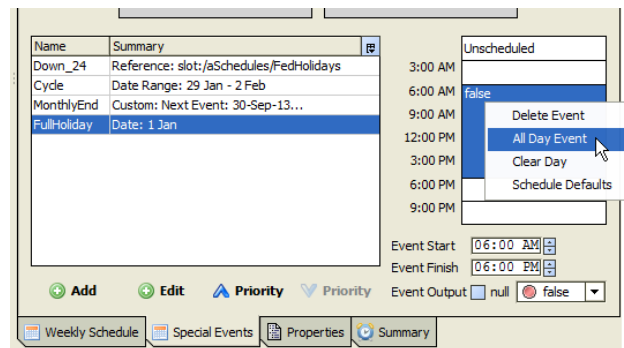


After you have a name and type selected (and defined as needed), click **OK** to add it to this schedule's special events. It remains selected for further editing, *except* for type.

Event times and output values A newly-created special event has no events defined. With the special event selected, click in the right-side events column and enter events as necessary. Start, finish, and output controls work the same as in the [Weekly Schedule](#) tab. See “[Event time “tuning”](#)” on page 7-8 and “[Output value](#)” on page 7-8 for details.

You can also right-click in the column for an event menu, as shown in [Figure 7-14](#). This is useful to add an all-day event or set the entire day to the schedule's default value.

Figure 7-14 Special event schedule actions



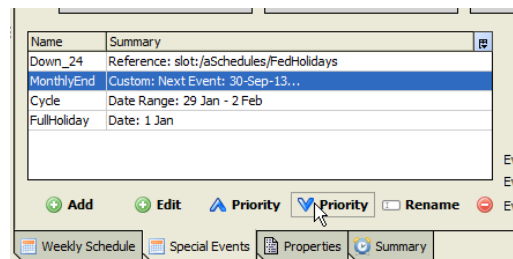
Note: You must specify events for any special event to occur. Where nothing is scheduled, the special event relinquishes control back to any lower-priority schedule events, and finally “intermingles” with the weekly schedule. To completely override the weekly schedule, configure a special event for the entire day.

Special event priorities All special events take priority over regular weekly events. Among special events, you define *relative priorities* by the order of listing in the Special Events table, as follows:

- *Highest* priority is at *top* of list. Events in this special event, when active, always occur.
- *Lowest* priority is at *bottom* of list. Events occur only if not overlapped by other special events active during the same period.

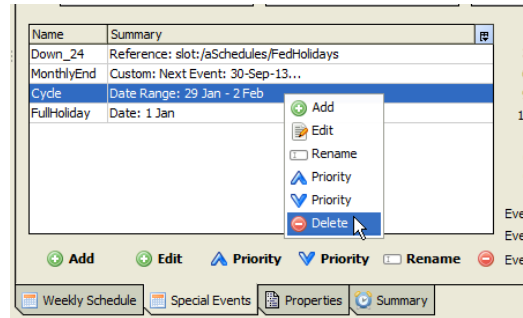
Change a special event's priority by selecting it and using the priority arrow buttons ([Figure 7-15](#)).

Figure 7-15 Change priority by listing order



Right-click menus and other controls Right-click in the special events table for a menu. If you have any special event *selected*, this menu provides the most commands, as shown in [Figure 7-16](#).

Figure 7-16 Right-click menu with event selected

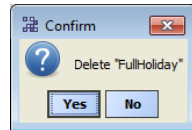


Special event menu options are straightforward, and may include the following:

- **Add** — Add a new special event (same as using **Add** button).
- **Edit** — Edit day(s) selection criteria (but *not changing* special event *type*). Same as **Edit** button.
- **Rename** — Rename selected special event (same as using **Rename** button).
- **Priority (up)** — Move special event up in priority list (same as using **Priority** button).
- **Priority (down)** — Move special event down in priority list (same as using **Priority** button).
- **Delete** — Removes selected special event from the schedule component.

Note: When you delete a special event, a confirmation dialog appears as shown in [Figure 7-17](#).

Figure 7-17 Delete special event confirmation

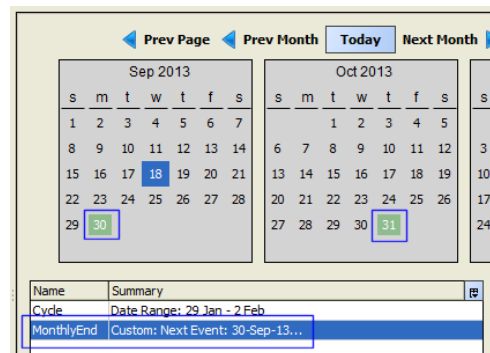


Click **Yes** to delete the special event, or **No** to keep it.

When you first access the Special Events tab, the current day is highlighted in the left-most calendar month at the top of the view. As needed, click on **Next Month** and **Prev Month**, or **Next Page** and **Prev Page** to traverse the calendar ahead or back in time.

When you select a special event in the table, if it occurs in any currently displayed month, its associated day or days are highlighted as shown in [Figure 7-18](#).

Figure 7-18 Special event highlights in calendar block



Note: A special event must have at least one defined event action to be highlighted in a calendar. Return to the current calendar month and day by clicking the **Today** button.

Properties

As shown in [Figure 7-19](#), this tab in **Weekly Scheduler view** is where you can specify the schedule's:

- **Effective Period**
- **Default Output**
- **Facets**
- **Cleanup Special Events** action

Figure 7-19 Properties tab in weekly Scheduler

Note: Another configuration property is also available, but only on the property sheet of a weekly schedule. See [“Scan Limit”](#) on page 7-13.

Effective Period By default, a weekly schedule added from the schedule palette is always effective. Whenever a schedule component is *not effective*, its *output* (Out slot) goes to its **default output** value, regardless of its weekly schedule or any special events.

In most cases, you *leave* weekly schedules as *always effective*. However, if you have an application for a schedule effective only at certain times, use the “start” through “end” range fields to limit the effective period. When you **Save** the changes, only effective days in the calendar months are shown highlighted green.

Default Output Whenever a schedule event (special or weekly) is not defined, the schedule component’s output (“Out” slot) is this value. The white area in listed events indicates where the default value is used and displays the current default value, as shown in [Figure 7-20](#). The default output value is also used whenever the schedule is *not effective*.

Figure 7-20 Default output is white area in schedule events

Default Output Value

	Sun	Mon	Tue	Wed
3:00 AM	Off	Off	Off	Off
6:00 AM		On	On	On
9:00 AM		On	On	On
12:00 PM		On	On	On

Note that “null” is an available choice—depending on control logic, this may be a valid choice.

As copied from the schedule palette, the *default* “Default Output” varies by schedule type, as follows:

- BooleanSchedule — false
- EnumSchedule — null
- NumericSchedule — null
- StringSchedule — null

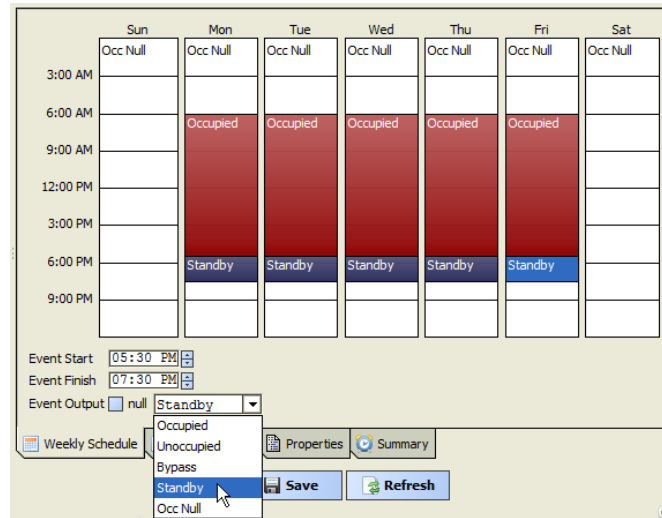
Facets The schedule component’s facets determine how its output value is formatted for display. For example, instead of “true” and “false” for a BooleanSchedule, you may need “On” and “Off” instead. Assigned facets appear in scheduler views when adding events, displaying summary data, and so on. For complete details, see [“About point facets”](#) on page 3-7.

Note: Facets are especially important for [EnumSchedules](#). You need to define “range” facets before you add weekly schedule events (in order to pick an event’s enumerated value). Range facets should match those used in any controlled (output-linked) EnumWritable. For related details, see [“Facets importance for Enum points”](#) on page 3-8.

In the case of StringSchedules (as for all string-type components) facets have no application.

Figure 7-21 shows output selections for an EnumSchedule with its range facet defined as “lonworks:LonOccupancyEnum,” one of the available frozen facets.

Figure 7-21 Facets determine event value selections for EnumSchedules



Note: Configured events shown in Figure 7-21 above reflect a few “default colors” available for EnumSchedules in AX-3.8. In AX-3.7 and earlier releases, configured schedule events all appear as “greenish” blocks. For related details, see “Event colors in weekly schedules (AX-3.8)” on page 7-14.

By default, facets for schedule components as copied from the schedule palette are as follows:

- BooleanSchedule — trueText: true, falseText: false
- EnumSchedule — range: <not defined>
- NumericSchedule — units: (null), precision: 1
- StringSchedule — (not applicable)

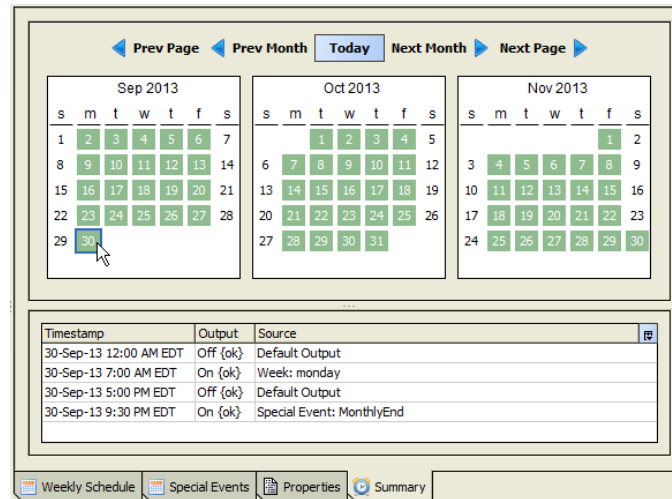
CleanUp Special Events This property is either true (default) or false.

- If true, “one-time” special events that have occurred (and will not be effective again) are automatically *deleted*. When a special event is deleted, a message is sent to the schedule log, and that special event no longer appears in the [Special Events](#) tab.
- If false, “one-time” special events are retained, even though they will not occur again.

Scan Limit (On a weekly schedule’s *property sheet*, but not on the Properties tab in Weekly Scheduler view.) Specifies a limit on how far ahead the schedule searches to find the next event output change, where the default is 90 days, and range is from 1 day (minimum) up. This can prevent excessive CPU usage. If changed, a value *less* than the default is typically recommended—for example, 14 days.

Summary

The summary tab in the [Weekly Scheduler](#) view shows a summary listing of *all* scheduled events for *any one selected day* in a weekly schedule (Figure 7-22). Events may be from the normal weekly schedule, special events, or a combination of both. Unlike with other tabbed views, this one is *read-only*.

Figure 7-22 Summary tab shows all events for any selected day

In the top calendar month area:

- Days with schedule events are shown *highlighted green*.
- Days without schedule events (only **default output**) are shown in *white*.

As needed, click on **Next Month** and **Prev Month**, or **Next Page** and **Prev Page** to traverse the calendar ahead or back in time.

- Click any day to see its events.
- Click **Today** (at top) to see the *current* day's events.

The table lists each event's *start* timestamp, the schedule's output value, and the event *source*. See "[Out Source slot](#)" on page 7-6 for how event source information appears.

Event colors in weekly schedules (AX-3.8)

In AX-3.8, configured events in weekly BooleanSchedules and EnumSchedules display with *different colors* according to event state (value). Also, as an alternative to the default colors used to display events in these schedule types, you can specify *custom colors* for the different event states. Different event colors can be particularly useful when using the scheduler view on mobile devices.

Note: Currently, schedule event colors do not apply to NumericSchedules or StringSchedules.

See the following for related details:

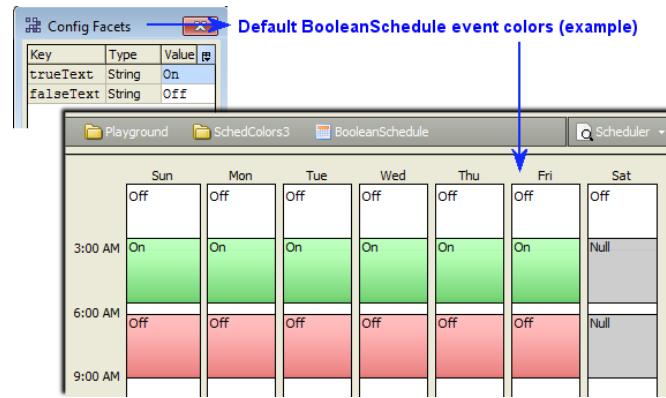
- "[Default event colors for EnumSchedules and BooleanSchedules](#)" on page 7-14.
- "[Customizing schedule event colors globally \(station-wide\)](#)" on page 7-15.
- "[Customizing schedule event colors at the component level](#)" on page 7-16.

Default event colors for EnumSchedules and BooleanSchedules

By default in AX-3.8 (and without any configuration), events configured in EnumSchedules and BooleanSchedules appear with in the Scheduler view with *different* colors for different event state values. This is a change from how configured events appear in AX-3.7 and prior EnumSchedules and BooleanSchedules, where all configured events appear with the *same* greenish color (varying only by state *text*).

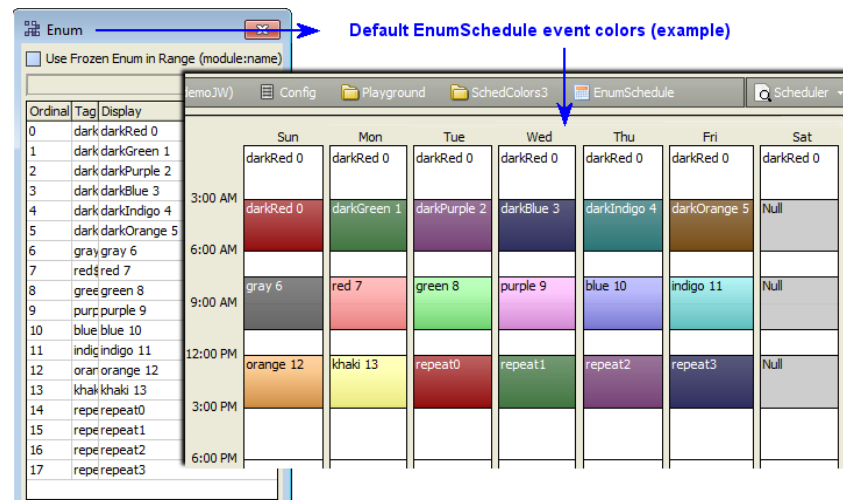
Default AX-3.8 schedule event colors are shown in an example BooleanSchedule ([Figure 7-23](#)) and EnumSchedule ([Figure 7-23](#)).

Figure 7-23 Default event colors in BooleanSchedule (AX-3.8)



As shown above, the default Boolean “true” state (On) is green, and the default Boolean “false” state (Off) is red. Any event configured for null is silver gray.

Figure 7-24 Default event colors in EnumSchedule (AX-3.8)



As shown above, the default colors for Enum states using ordinals 0 through 13 use different colors, starting to repeat again at ordinal 14. In addition, any event configured for null is silver gray.

Note: The “default value” areas of any weekly schedule (including BooleanSchedule and EnumSchedule), always display as white, regardless of the value of the schedule’s Default Output property. Only configured events display using colors. You can see this in examples above, where adjacent “Off” (Boolean) or “0” (Enum) schedule blocks are either colored (configured) or white (default).

Color names Note that acceptable color names derive from the SVG 1.0 specification. At the time of this document update, a web URL to a sortable table of acceptable color names, along with their associated RGB hex codes, can be found here: <http://www.uize.com/examples/sortable-color-table.html>

Customizing schedule event colors globally (station-wide)

For an AX-3.8 station, you can *globally* specify custom schedule event colors for BooleanSchedules and/or EnumSchedules to use, by entering corresponding *lexicon keys* for the `schedule` module.

Such lexicon keys use the format:

`BooleanSchedule.colors.n=colorNameOrCode` [where *n* is either 0 (false) or 1 (true)], or
`EnumSchedule.colors.n=colorNameOrCode` [where *n* is an ordinal integer value, e.g. 0, 1, 2...]

Where you can also specify a null color with `TypeSchedule.colors.null=colorNameOrCode`
See the “Color names” section.

EnumSchedule entries in schedule lexicon

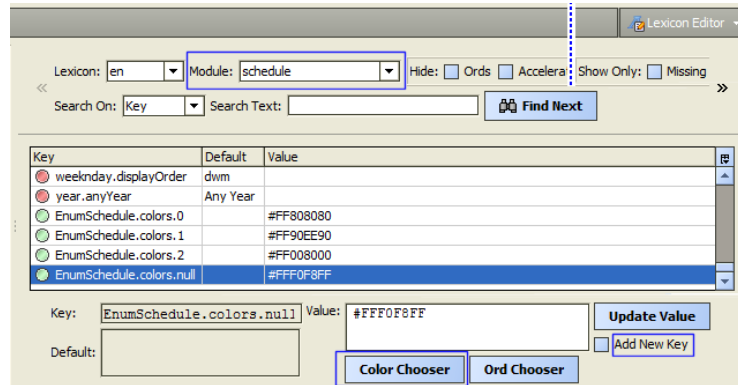
Some entries in the `schedule` lexicon for EnumSchedules, shown below (and in **Lexicon Editor**).

`EnumSchedule.colors.null=#FFF0F8FF (aliceBlue)`

`EnumSchedule.colors.0=#FF80800 (gray)`

EnumSchedule.colors.1=#FF90EE90 (lightGreen)

EnumSchedule.colors.2=#FF008000 (green)



Note: Click the **Color Chooser** button when adding or editing lexicon keys for schedule colors, which produces a popup **Color Chooser** dialog where you can easily choose colors (without knowing codes or names).

Lexicon event color notes Keep in mind the following when using the lexicon (global) method to customize event colors in BooleanSchedules and EnumSchedules on a AX-3.8 station:

- After editing and saving the `schedule` lexicon, if a local station (e.g. Supervisor), you will need to restart that station (and Workbench) before changes become effective.
- Assign station users the two-character lexicon code, for example, “en” if any English language/U.S. application. This provides support for browser-connected station users running “Web Workbench”, that is the WbApplet. For related details see the “Language” property in section “User” on page 6-30.
- Schedule event colors specified will override *default* event colors on *any* station that runs on the host with the lexicon changes. In other words, this is not a “station specific” change.
- If event color changes are to be used by a remote (JACE) host, you typically build a *lexicon module* with the `schedule` module changes and install it using the platform **Software Manager**.
- BooleanSchedules or EnumSchedules can also have custom event colors specified *directly at the component level*, using added facets. In this case, these custom event colors *override* any “global colors” from lexicon keys, in addition to schedule “default colors”.

See the following for related details:

- In the *NiagaraAX Lexicon Guide*, “Notes on English (en) lexicon usage”, as well as other sections including the “Lexicon Editor view” and “Lexicon Module Builder view”.
- “Customizing schedule event colors at the component level” on page 7-16.
- “Default event colors for EnumSchedules and BooleanSchedules” on page 7-14.

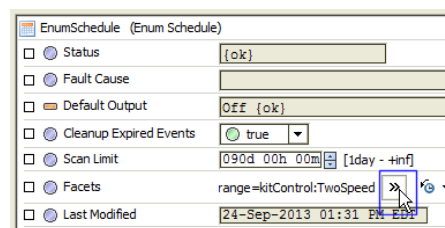
Customizing schedule event colors at the component level

In an AX-3.8 station, you can *individually* specify custom schedule event colors for BooleanSchedules and/or EnumSchedules to use, by adding additional *facets* on each schedule. These custom event colors *override* any colors “globally” sourced from schedule lexicon keys (if applicable), as well as “default colors”.


Customize schedule event colors using facets

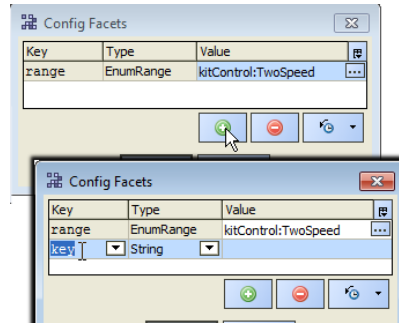
To do this, from the property sheet of the EnumSchedule or BooleanSchedule:

- Step 1 Click the **Facets** control (>>) for the popup **Config Facets** dialog.

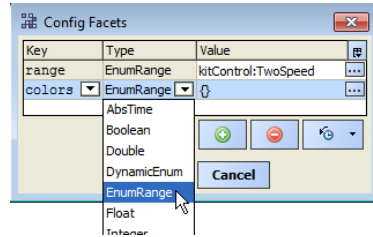


Note: Typically, you should already have “normal facets” defined, such as “range” facets if an EnumSchedule, or if a BooleanSchedule, “falseText” and “trueText” facets. If not, it is recommended you configure these first.

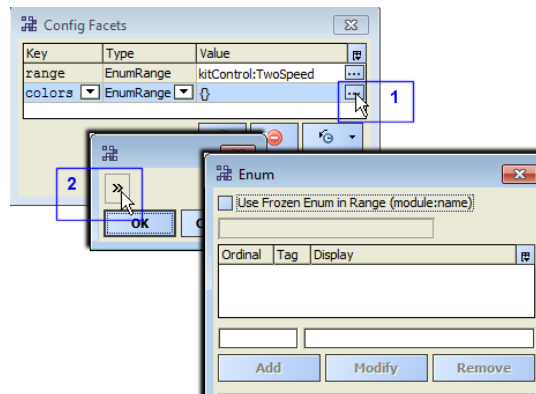
Step 2 In the **Config Facets** dialog, click  to *add* a new facets key, which by default is **Key** “key”.



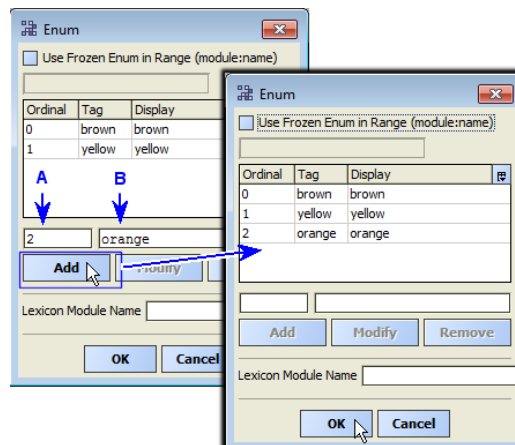
Step 3 Type over “key”, entering: **colors**, and in the **Type** drop-down select **EnumRange**.



Step 4 Click its **Value** expand (...) control, and next the (>>) control for the popup **Enum** dialog, which is initially empty.



Step 5 As needed, add facet entries for colors by clicking in the left-side (Ordinal) field and entering a numeral, then in the right-side field entering either a color name or color code (see “[Color names](#)” on page 7-15). Then click **Add**.

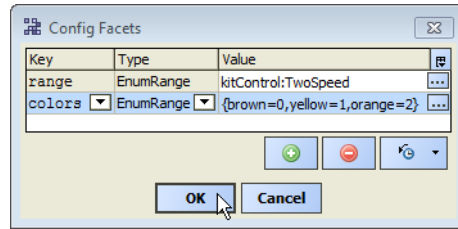


The example above shows colors assigned to three enum states corresponding to ordinals 1, 2, and 3.

- In the case of a **BooleanSchedule**, ordinal 0=**false** and 1=**true**.
- Currently, there is no colors key entry for a configured “null” value. However, you can set a “global”

null event color, for either/both BooleanSchedules and EnumSchedules using the *lexicon* method. When finished, click **OK**, then **>>**, to close the **Enum** dialogs.

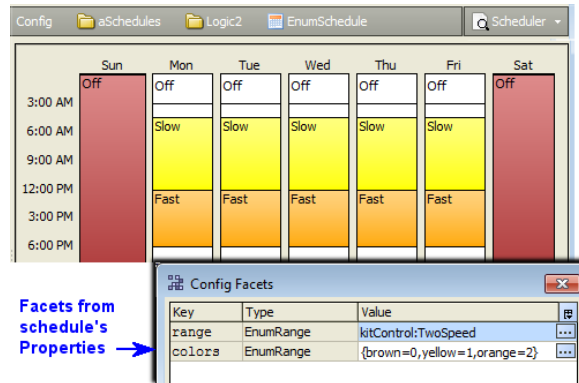
Step 6 The **Config Facets** dialog should reflect the added enum states in the `colors` key (example below).



Click **OK** to return to the schedule's property sheet, then click **Save**.

The new event colors are now immediately effective in this schedule. See [Figure 7-25](#) for results.

Figure 7-25 Example custom event color results using facets method in EnumSchedule



Note: Again, only “configured events” show the assigned event color. In the [Figure 7-25](#) example above, both Sunday and Saturday have the entire day configured as “Off”, while the remaining days of the week only have “Slow” or “Fast” configured events—the “Default Value” (in this case “Off”) always appears as white.

See the following for related details:

- “Default event colors for EnumSchedules and BooleanSchedules” on page 7-14.
- “Customizing schedule event colors globally (station-wide)” on page 7-15.
- “Color names” on page 7-15.

About calendar schedules (holidays)

Calendar schedules (CalendarSchedule component) specify *entire days*, using the following four types of day event selections:

- Date
- Date Range
- Week and Day
- Custom

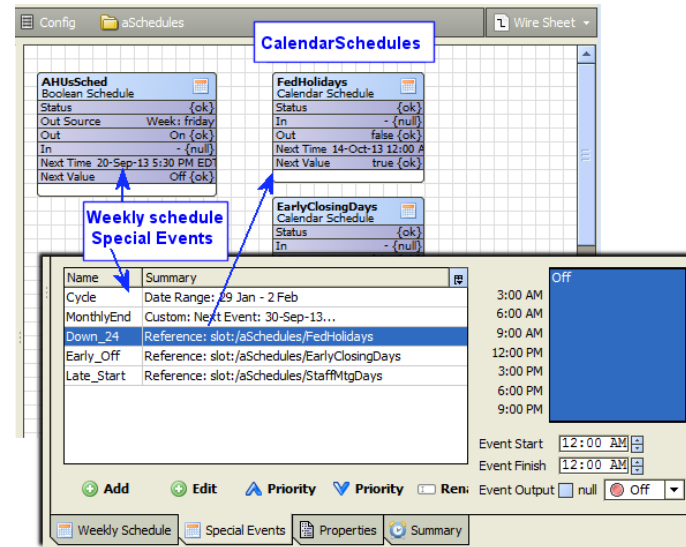
You can add as many day events as needed in the same CalendarSchedule. The following sections provide more details:

- “CalendarSchedule Usage” on page 7-18
- “Calendar Scheduler view” on page 7-19
- “Calendar day selections” on page 7-21
- “CalendarSchedule slots and other notes” on page 7-23

CalendarSchedule Usage

Instead of linking CalendarSchedules, you typically “reference” them from the “special events” configuration of one or more weekly schedules. Each referenced CalendarSchedule defines the “day portion” of a special event. Then, you configure time-of-day events in each special event, as needed.

Figure 7-26 Example referenced CalendarSchedules



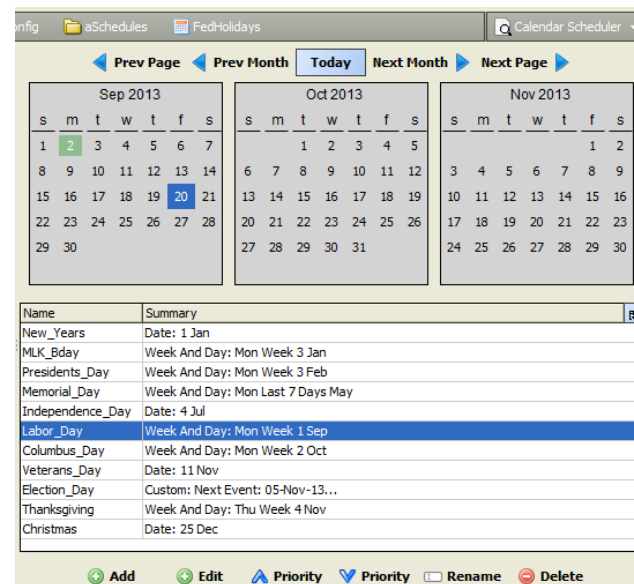
For example, [Figure 7-26](#) shows a [BooleanSchedule](#) and a portion of its special events tab, listing five special events. Three of these are “Reference” types, meaning their calendar day(s) are defined *remotely* in the configuration of the referenced CalendarSchedules. Although all components are shown here in the same container, quite often CalendarSchedules are located elsewhere in the station.

CalendarSchedule usage by “special event reference” allows *global changing* of day definitions, where *multiple* weekly schedules can reference one or more CalendarSchedules. Any edit of a CalendarSchedule affects all weekly schedules containing a special event that references it.

Calendar Scheduler view

Double-click a CalendarSchedule to see its default Calendar Scheduler view, as shown in [Figure 7-27](#).

Figure 7-27 Calendar Scheduler view



As shown in [Figure 7-27](#), existing calendar events (if any) are listed in the table by name and summary. When you select a calendar event, its day(s) of occurrence are highlighted in green in the monthly calendars at the top of the view.

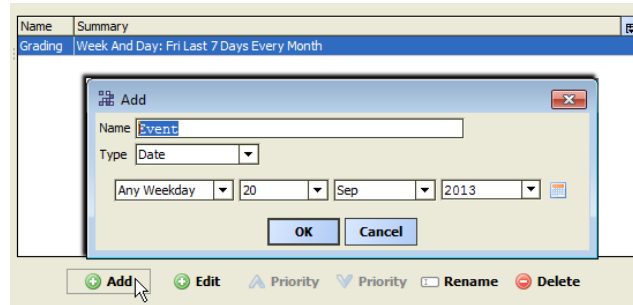
Additional Calendar Scheduler topics include:

- [Adding calendar events](#)
- [Right-click menus and other controls](#)

Adding calendar events

Click the **Add** button to add a new calendar event. An Add dialog appears, as shown in [Figure 7-28](#).

Figure 7-28 Special Events Add dialog



Dialog options are described as follows:

- Name — Your descriptive name for the calendar days, perhaps “Thanksgiving_Break” or “Cleaning_2.” The default value is simply “Event.” You can change this later, if needed.
- Type — Determines selection criteria for day or days, with the following choices:
- Date — (default) By various combinations of weekday, numerical date, month or month combinations, and year.
- Date Range — By start and end range, using for each a combination of day, month, year.
- Week and Day — By combination of day of week, week in month, month.
- Custom — By various combinations of day, month, weekdays, and year.

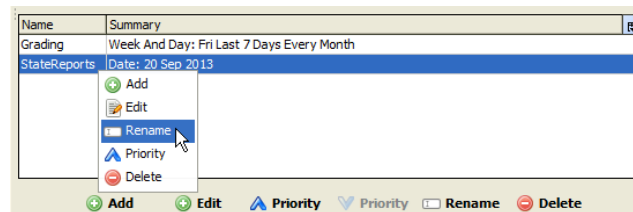
For details on Type selections, see “[Calendar day selections](#)” on page 7-21.

After you have a name and type selected (and defined as needed), click **OK** to add it to this calendar’s days. It remains selected for further editing, *except* for type.

Right-click menus and other controls

Right-click in the calendar events table for a menu. If you have any calendar event *selected*, this menu provides the most commands, as shown in [Figure 7-29](#).

Figure 7-29 Right-click menu with event selected



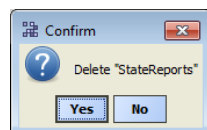
Note: Priority selections (right-click menu or in bottom buttons) only affect the list order for events in a *CalendarSchedule*—true priority applies only to special events (in weekly schedules).

Calendar event menu options are straightforward, and may include the following:

- Add — Add a new calendar event (same as using **Add** button).
- Edit — Edit day(s) selection criteria (but *not changing* calendar type). Same as **Edit** button.
- Rename — Rename selected calendar event (same as using **Rename** button).
- Priority (up) — Move calendar event up in display list (same as using **Priority** button).
- Priority (down) — Move calendar event down in display list (same as using **Priority** button).
- Delete — Removes selected calendar event from the schedule component.

Note: When you delete a calendar event, a confirmation dialog appears as shown in [Figure 7-30](#).

Figure 7-30 Delete calendar event confirmation



Click **Yes** to delete the calendar entry, or **No** to keep it.

When you first access the Calendar Scheduler, the current day is highlighted in the left-most calendar month at the top of the view. As needed, click on **Next Month** and **Prev Month**, or **Next Page** and **Prev Page** to traverse the calendar ahead or back in time.

Return to the current calendar month and day by clicking the **Today** button.

Calendar day selections

When adding calendar days in a CalendarSchedule, a special event in a weekly schedule, or a trigger event in a trigger schedule, the following “Type” selections are available:

- Date — see “[Date selection notes](#)” on page 7-21
- Date Range — see “[Date range selection notes](#)” on page 7-21
- Week and Day — see “[Week and day selection notes](#)” on page 7-21
- Custom — see “[Custom selection notes](#)” on page 7-22

Date selection notes

As shown in [Figure 7-31](#), *Date* calendar selection has 4 criteria: weekday, day-of-month, month, and year.

Figure 7-31 Calendar selection by date

The screenshot shows a dialog box titled 'Add'. It has a 'Name' field with 'NewYears' and a 'Type' dropdown set to 'Date'. Below these are four selection fields: 'Any Weekday', '21', 'Sep', and 'Any Year'. At the bottom, a legend identifies these as 'weekday', 'day-of-month', 'month', and 'year' respectively.

You can make only *one* selection in each criteria (weekday, day-of-month, month, year). Each criteria offers an “any” selection, in addition to a specific selection.

Result of selections is by “ANDing” all criteria. For example, if you select weekday of Tuesday, day of month as 5, and remaining criteria “any,” the event is specified only on Tuesday, the fifth of any month in any year. If a month does not have Tuesday the fifth, then there is no event that month.

Date range selection notes

As shown in [Figure 7-32](#), *Date Range* calendar selection has a start range and end range, each with 3 criteria: day-of-month, month, and year.

Figure 7-32 Calendar selection by date range

The screenshot shows a dialog box titled 'Add'. It has a 'Name' field with 'Winter Hours' and a 'Type' dropdown set to 'Date Range'. Below these are two range selection fields. The 'Start range' has '26', 'Dec', and 'Any Year'. The 'End range' has '1', 'Mar', and 'Any Year'. A 'Through' label is between the two ranges. At the bottom, a legend identifies the criteria as 'day-of-month', 'month', and 'year'.

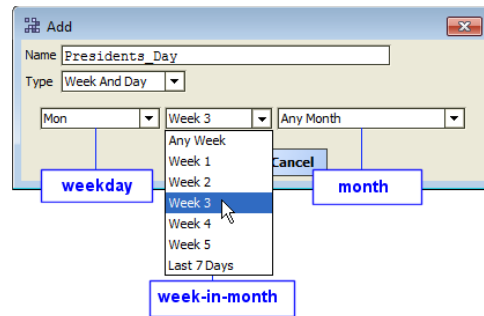
You can make only *one* selection in each criteria (day-of-month, month, year). Each criteria offers an “any” selection, in addition to a specific selection.

In each date range, result is from “ANDing” the criteria. In addition, the start day can be after the end date. For example, as shown in [Figure 7-32](#), the start day can be in December and the end date in March. This event occurs December, January and February.

Week and day selection notes

As shown in [Figure 7-33](#), *Week and Day* calendar selection has 3 criteria: weekday, week-in-month, and month.

Figure 7-33 Calendar selection by week and day



You can make only *one* selection in each criteria (weekday, week-in-month, month). Each criteria offers an “any” selection, in addition to a specific selection.

In addition, the following criteria offer additional selections, as follows:

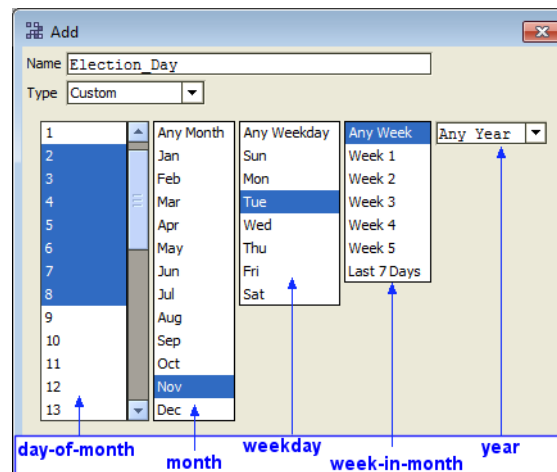
- weekday:
- week-in-month:
 - Last 7 Days
- month:
 - Jan-Mar-May-Jul-Sep-Nov
 - Feb-Apr-Jun-Aug-Oct-Dec

The selection result is from “ANDing” the criteria. For example, as shown in [Figure 7-33](#), if selections are for weekday as Monday, the month as February, and the week as 3, the event occurs only on the third Monday in February.

Custom selection notes

As shown in [Figure 7-34](#), *Custom* calendar selection has 5 criteria: day-of-month, month, weekday, week-in-month, and year.

Figure 7-34 Calendar selection by custom



Unlike with other calendar types, you can make *multiple* selections within each criteria (except if you select “any,” which allows only that selection). To select multiples, first select something *other* than “Any,” then hold down the Ctrl or Shift key while you select more values.

Each criteria offers an “any” selection, in addition to a specific selection. In addition, the following criteria offer additional selections, as follows:

- day-of-month:
 - Last Day
 - Last 7 Days
- week-in-month:
 - Last 7 Days

Within any criteria, selections are “OR’ed” The overall result is from “AND’ing” all criteria. For example, [Figure 7-34](#) shows a custom selection for U.S. General Election Day, which must be configured as the “first Tuesday after the first Monday in November.”

CalendarSchedule slots and other notes

The CalendarSchedule has a StatusBoolean input and output (slots “In” and “Out”). In most applications they are left unlinked.

- Output is *true* during any configured calendar day(s), otherwise it is false.
- Input overrides the calendar schedule calculation (if linked). Any non-null input (true or false) is immediately passed to the output.

By default, for any CalendarSchedule copied from the schedule palette, the following additional slots are pinned on the component’s glyph (shape on wire sheet):

- **Next Time**
Time of next scheduled output change for the CalendarSchedule. If more than a year away, this value is null.
- **Next Value**
The next scheduled output value, at “Next Time”. Value is meaningless if Next Time is null.

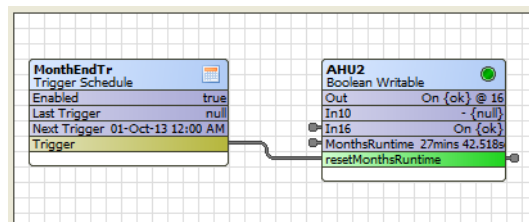
From the CalendarSchedule’s property sheet, you can access and change the following additional slots:

- **Cleanup Expired Events**
Either true (default) or false.
 - If true, calendar events of type “Date” and “Date Range” that have already occurred, *and cannot occur again* (as configured), *are automatically deleted* after they occur. This is recorded in the schedule log, and you no longer see them in the [Calendar Scheduler view](#).
 - If false, all calendar events are retained, *even if they cannot occur again* (as configured).
- **Scan Limit**
Specifies a limit on how far ahead the schedule searches to find the next event output change, where the default is 90 days, and range is from 1 day (minimum) up. This can prevent excessive CPU usage. If changed, a value *less* than the default is typically recommended—for example, 14 days.
- **Facets**
Facets applied to the input and output, which by default are simply “true” and “false.”

About trigger schedules

TriggerSchedules are special-purpose schedules, providing scheduling control for either linked *actions* or *topics* of other components. [Figure 7-35](#) shows a simple example of a TriggerSchedule linked to an action slot of a DiscreteTotalizerExt, which was “composited” in the parent BooleanWritable.

Figure 7-35 Example TriggerSchedule



This schedule is configured to simply fire *once* at midnight on the first day of every month. The trigger at the “ResetElapsedActiveTime” slot zeroes the runtime accumulated from the previous month.

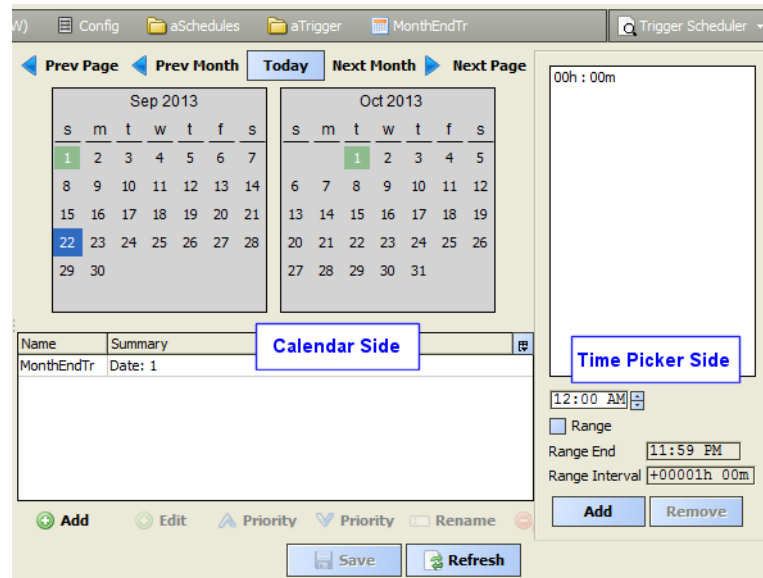
The following sections provide more TriggerSchedule details:

- [“Trigger Scheduler view”](#) on page 7-23
- [“TriggerSchedule slots and other notes”](#) on page 7-26

Trigger Scheduler view

Double-click a TriggerSchedule to see its default Trigger Scheduler view, as shown in [Figure 7-36](#).

Figure 7-36 Trigger Scheduler view



As shown in [Figure 7-36](#), the Trigger Scheduler has two sides:

- Calendar (left) side — where you *add* events. It operates like the [Calendar Scheduler view](#).
- Time picker (right) side — where you add *trigger times* for the schedule to fire its trigger output. Included is the ability to add repeating intervals.

Note: *Trigger times, as set in the time picker, apply to all calendar events (if more than one).*

Existing trigger events (if any) are listed in the table by name and summary. When you select a trigger event, its day(s) of occurrence are highlighted in green in the monthly calendars at the top of the view. Trigger times are listed in the time picker area.

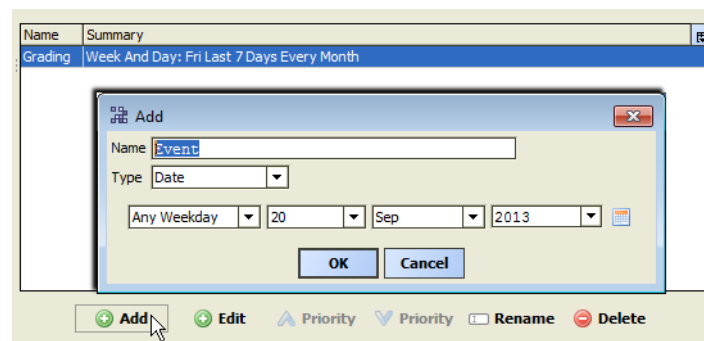
Additional Trigger Scheduler topics include:

- [Adding trigger events](#)
- [Adding trigger event times](#)

Adding trigger events

Click the **Add** button to add a new trigger event. An Add dialog appears, as shown in [Figure 7-37](#).

Figure 7-37 Trigger Event Add dialog



Dialog options are described as follows:

- Name — Your descriptive name for the trigger events, perhaps “FirstDOM” or “Each_WorkHr.” The default value is simply “Event.” You can change this later, if needed.
- Type — Determines selection criteria for day or days, with the following choices:
- Date — (default) By various combinations of weekday, numerical date, month or month combinations, and year.
- Date Range — By start and end range, using for each a combination of day, month, year.
- Week and Day — By combination of day of week, week in month, month.
- Custom — By various combinations of day, month, weekdays, and year.

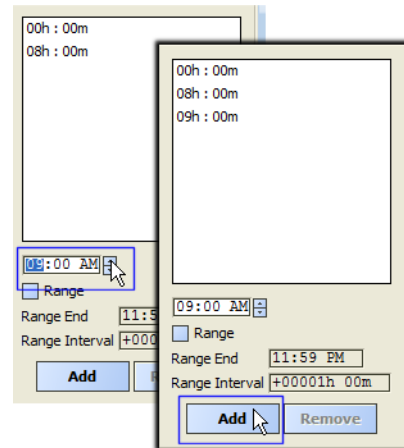
For details on Type selections, see “[Calendar day selections](#)” on page 7-21.

After you have a name and type selected (and defined as needed), click **OK** to add it to this calendar's days. It remains selected for further editing, *except* for type. Typically, you add one or more trigger event *times* on the time picker side. See “Adding trigger event times” on page 7-25.

Adding trigger event times

By default, a single “midnight” trigger time may already exist (you can delete it if desired). To add other trigger times, use the controls at the bottom of the time picker side, as shown in [Figure 7-38](#).

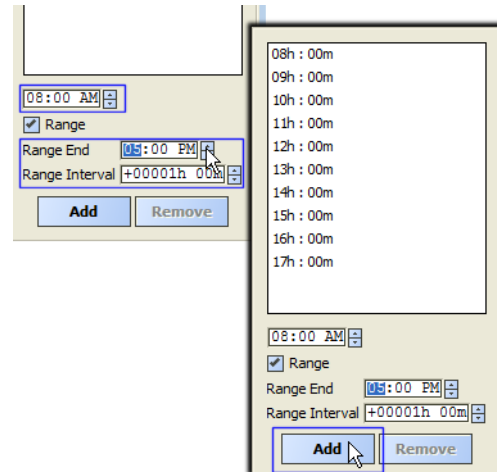
Figure 7-38 Trigger time picker controls



Set the desired time in the hour:minute editor, either by clicking up/down controls or typing in times directly. Click the **Add** button to add a trigger at that time, which adds it to the list. You can also enter multiple triggers simultaneously, using the [Range option](#).

Range option To add multiple triggers that occur at a repeating interval, select the Range checkbox. This enables the Range End and Range Interval fields for entering values, as shown in [Figure 7-39](#).

Figure 7-39 Range option in trigger time picker

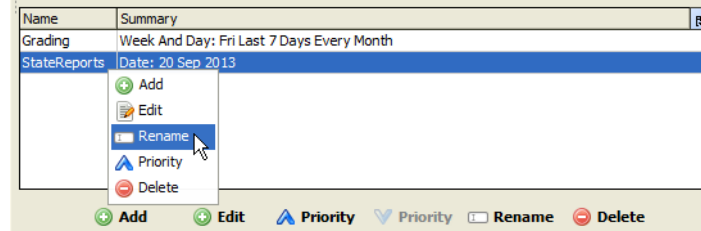


When entering a trigger range, note that the top (hour:minute) editor acts as the *first* (or Range Begin) trigger time. By default, the Range Interval is set to one hour (“+00001h 00m 00.000s”). You can set this to whatever interval is needed.

To delete a trigger time, click to select, then click the **Remove** button. To select multiple trigger times, hold down the Ctrl or Shift key while you select.

Right-click menus and other controls

The time picker (right side) has no right-click menu—simply use the bottom controls to configure trigger times. The calendar (left side) has an available right-click menu. If you have any calendar event *selected*, this menu provides the most commands, as shown in [Figure 7-40](#).

Figure 7-40 Right-click menu with event selected

Note: Priority selections (right-click menu or in bottom buttons) only affect the list order for events in a TriggerSchedule—true priority applies only to special events (in weekly schedules).

Event menu options are straightforward, and may include the following:

- **Add** — Add a new calendar event (same as using **Add** button).
- **Edit** — Edit day(s) selection criteria (but *not changing* calendar type). Same as **Edit** button.
- **Rename** — Rename selected calendar event (same as using **Rename** button).
- **Priority (up)** — Move calendar event up in display list (same as using **Priority** button).
- **Priority (down)** — Move calendar event down in display list (same as using **Priority** button).
- **Delete** — Removes selected calendar event from the schedule component.

Note: When you delete a calendar event, a confirmation dialog appears as shown in [Figure 7-41](#).

Figure 7-41 Delete calendar event confirmation

Click **Yes** to delete the calendar entry, or **No** to keep it.

When you first access the Trigger Scheduler, the current day is highlighted in the left-most calendar month at the top of the view. As needed, click on **Next Month** and **Prev Month**, or **Next Page** and **Prev Page** to traverse the calendar ahead or back in time.

Return to the current calendar month and day by clicking the **Today** button.

TriggerSchedule slots and other notes

By default, for any TriggerSchedule copied from the schedule palette, the main “Trigger” output slot is pinned on the component’s glyph (shape on wire sheet), as well as the following additional ones:

- **Enabled**
Either true (default) or false. While false, firing of trigger outputs is disabled. If linking, note that this slot uses simple Boolean, you may need a StatusBooleanToBoolean (kitControl) component.
- **Last Trigger**
Timestamp of the last firing of the trigger output.
- **Next Trigger**
Next scheduled trigger firing time.

From the TriggerSchedule’s property sheet, you can access the following additional slots:

- **Next Trigger Search Limit**
How far into the future to search for the next trigger. It is possible a poorly-configured schedule will never fire a trigger. This setting prevents an infinite search. Default is 90 days (2160 hours).
- **Last Modified**
(read only) Timestamp of last configuration change.
- **Status**
(read only) Status of the schedule component, typically “{ok}.”

Trigger Missed slot

In addition to the “main” Trigger output slot, the TriggerSchedule has “Trigger Missed” slot, also a topic type slot. If the station was not running when a scheduled trigger was to occur (appeared previously in “Next Trigger” property), upon station startup the Trigger Missed slot fires once.

Note: Trigger Missed always fires only once, no matter how many triggers may have been missed.

About ScheduleSelector components

ScheduleSelector components provide an easy way for users to select a schedule to use for controlling a particular component. For example, you can use a ScheduleSelector component to link a pre-configured schedule to a component that is controlling equipment, a door schedule, or setpoint temperature. With a set of pre-configured schedules and the ScheduleSelector component, users can simply choose from a list of valid schedules when they want to setup or change the schedule of a device. Users do not have to actually configure schedule properties.

Each ScheduleSelector component contains a property that allows you to target a single location or "container" that holds all the schedule components that you want to offer as options for the ScheduleSelector component. After you define the container location, the ScheduleSelector component populates the "Schedule" property option list with all valid schedule components. You can choose any of the options - or leave the property set to the default "null" value. When you select a schedule, the ScheduleSelector component creates a link between the selected schedule and the component that is linked to the output of the ScheduleSelector.

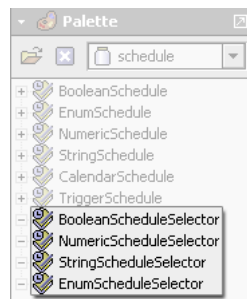
Types of ScheduleSelector components

There are four types of ScheduleSelector components available in the Schedule palette, representing each of the four data types:

- BooleanScheduleSelector
- NumericScheduleSelector
- StringScheduleSelector
- EnumScheduleSelector

ScheduleSelectors may only provide links between schedules and control components that are of the same data type. For example, a BooleanScheduleSelector component only links a BooleanSchedule component to a control component of the Boolean data type. An EnumScheduleSelector links between an EnumSchedule component and a control component of the Enum data type. The same relationship applies to Numeric and String schedule selectors.

Figure 7-42 Schedule Selector Components in the Schedule Palette



If you have schedules of different data types in the same container, the ScheduleSelector only displays the valid schedule components in the "Schedule" property options list.

Types of ScheduleSelector component properties

All ScheduleSelector components have the following properties:

- **Container**
This property provides a text field that uses an ORD to specify the location of the available schedules. Use the component chooser at the right side of the field to browse to and choose the desired schedule container.
- **Schedule**
This option list displays all available schedule components of the appropriate data type.
 - Selecting and saving a schedule using this property's option list automatically creates a link from the "out" slot of the selected schedule to the "in" slot of the ScheduleSelector component.
 - Selecting "null" for this property causes the "out" property to generate a null value, and automatically removes any link from a schedule component to the "in" slot of the ScheduleSelector component.
- **Facets**
This property does not require configuration. Facets are automatically inherited from whatever schedule is currently selected.

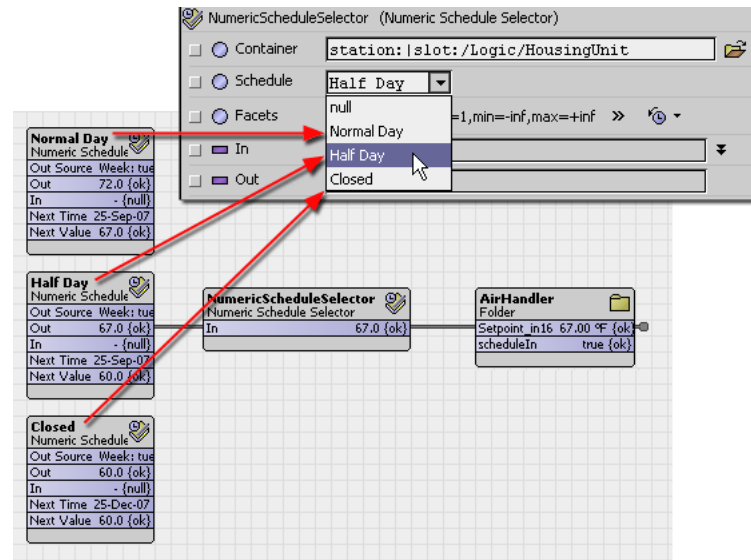
- **In**
Whenever the ScheduleSelector component has been used to select a schedule, the schedule's output supplies this value (via the automatic link) to this slot. The default setting of this slot is the null value, produced when no schedule is selected.
- **Out**
This is the output value (from the linked schedule) that is passed to whatever is controlled (linked to) this slot.

Example ScheduleSelector configurations

Example 1: Using a NumericScheduleSelector component

The following illustration shows an example of a NumericScheduleSelector:

Figure 7-43 Property Sheet and Wire Sheet Views of a NumericScheduleSelector Configurations



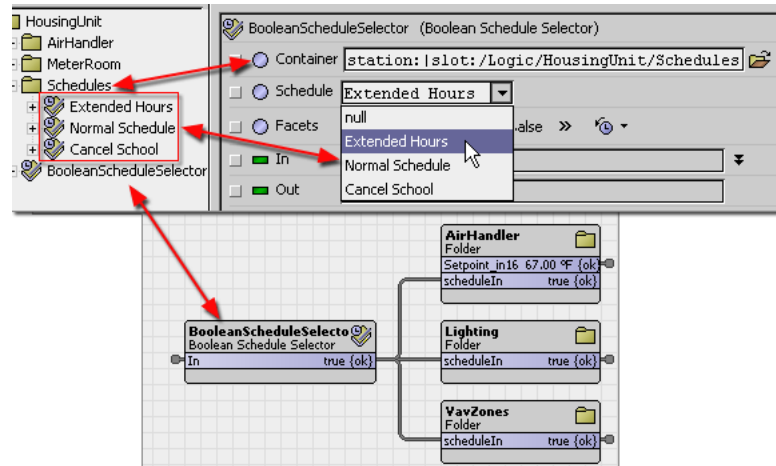
Notice the following points about this configuration:

- The container property is set to the "station:|slot:/Logic/HousingUnit" ORD. This is the container that holds the three numeric schedules.
- The Schedule property option list has three numeric schedules available for selection. Notice that the schedules are the same ones shown on the wire sheet view of the folder component labeled "HousingUnit".
- The link (visible in the Wire Sheet view) is connected from the numeric schedule Out to the schedule selector In.
- The linked value (67.0 in this example) is passed from the NumericScheduleSelector component Out to the AirHandler device Setpoint input.


Example 2: Using a BooleanScheduleSelector component

The following illustration shows an example of a BooleanScheduleSelector configuration.

Figure 7-44 Property Sheet and Wire Sheet Views of a BooleanScheduleConnector Configuration



Notice the following points about this configuration:

- The BooleanScheduleSelector Container property is linked to the "Schedules" container (see nav tree).
- Three BooleanSchedule components are in the container and are available as options in the Schedule property option list.
- On the wire sheet view, the BooleanScheduleSelector shows a "knob" link  from the selected schedule to the schedule selector In.
- The BooleanScheduleSelector Out is linked to three device inputs.

Using schedules

The following main sections provide common tasks for working with schedules in NiagaraAX:

- ["Adding a schedule or calendar"](#) on page 7-29
- ["Configuring schedules and calendars"](#) on page 7-30
- ["Linking schedules"](#) on page 7-32
- ["Importing schedules"](#) on page 7-33

Adding a schedule or calendar

In your station, you can copy a default schedule component directly from the schedule palette, or copy a pre-configured schedule component from a saved bog file or other station database.

Note: Once in your station and configured, you can also duplicate schedule components (see ["To duplicate a component \(property sheet, nav side bar, or wire sheet view\)"](#) on page 9-16).

To add a component from the schedule palette

You can add (copy) a new schedule component from the schedule palette, dragging onto a container in the Nav tree view or into the wire sheet view.

- Step 1 Open the **schedule** palette. See ["Using the palette side bar"](#) on page 9-10 for general details. When the palette is open, schedule components are listed, as shown in [Figure 7-1](#) on page 1.
- Step 2 Drag the component needed from the palette to either:
 - In the Nav tree, onto a folder in your station.
 - In the view pane, into the wire sheet of a folder in your station.
 A **Name** dialog appears.
- Step 3 In the **Name** dialog, type the desired name and click **OK**.
The schedule or calendar is now in your station, with default values.

To copy an existing, saved (configured) schedule component

You can add (copy) a new schedule component from one previously configured, dragging onto a container in the Nav tree view or into a folder's wire sheet view. To add a new schedule component as a copy of an existing saved component, do the following:

- Step 1 In the Nav tree, expand **My File System** and navigate to the saved bog file or station config.bog file. See ["Editing components"](#) on page 9-13 for more details.

- Step 2 When you locate the desired schedule component, right-click and select **Copy**.
- Step 3 Paste the copied component doing a right-click **Paste** from either:
- In the Nav tree, on a folder in your station
 - In the view pane, in the wire sheet of a folder in your station.
- A **Name** dialog appears.
- Step 4 In the **Name** dialog, type the desired name and click **OK**.
The schedule or calendar is now in your station, with pre-configured values.

Configuring schedules and calendars

Typically, you perform most configuration of any schedule component from its default “scheduler” view. For an overview, see “[Schedule component views](#)” on page 7-2.

- [Configuring weekly schedules](#)
- [Configuring calendar schedules](#)
- [Configuring trigger schedules](#)

Configuring weekly schedules

Weekly schedules ([BooleanSchedule](#), [EnumSchedule](#), [NumericSchedule](#), and [StringSchedule](#)) are among the most used schedule components.

To configure a weekly schedule

You use the [Weekly Scheduler view](#) to configure a weekly schedule. There are many ways you can approach configuration, the following procedure is only one way.

- Step 1 Double-click the weekly schedule component.
The Weekly Scheduler displays, as shown in [Figure 7-6](#) on page 7.
- Step 2 Configure important properties. See “[To configure a weekly schedule’s properties](#)” on page 7-30.
- Step 3 Configure the normal weekly schedule. See “[To configure the weekly \(normal\) schedule](#)” on page 7-30.
- Step 4 Configure special events, if any. See “[To add and configure special events](#)” on page 7-30.
- Step 5 Review all configuration. See “[To review a weekly schedule’s configuration](#)” on page 7-31.

To configure a weekly schedule’s properties

To configure a weekly schedule’s properties, do the following:

- Step 1 In the [Weekly Scheduler view](#), click the [Properties](#) tab, and configure the following:
- [Facets](#) — This is critical if an [EnumSchedule](#), and optional if a [BooleanSchedule](#) or [NumericSchedule](#). Note that facets do not apply if a [StringSchedule](#).
 - [Default Output](#) — This is the schedule’s output whenever an event (either weekly or special event) is not active. It is also used whenever the component is “not effective.”
- Typically, you leave other properties at default values.

- Step 2 Click [Save](#).

To configure the weekly (normal) schedule

To configure the normal weekly schedule, do the following:

- Step 1 In the [Weekly Scheduler view](#), click the [Weekly Schedule](#) tab.
- Step 2 In the events table, click and drag to add an event, and enter its output value.
For details, see “[Event time “tuning”](#)” on page 7-8 and “[Output value](#)” on page 7-8.
- Note:** *If a Boolean or EnumSchedule, select output value from the drop-down control.
If a Numeric or StringSchedule, type the output value and press Enter.*
- Step 3 Right-click the event and use whatever menus are needed.
For details, see “[Right-click menus](#)” on page 7-8.
- Step 4 Continue to add, delete, or adjust events as needed.
- Step 5 When the weekly schedule is what you want, click [Save](#).

To add and configure special events

Special events are exceptions to the normal weekly schedule, and typically include recurring holidays and “one-time” events. For more details, see “[Schedule special events](#)” on page 7-3.

- Step 1 In the [Weekly Scheduler view](#), click the [Special Events](#) tab.

- Step 2 Click the **Add** button for a dialog to add a special event.
In the **Add** dialog, specify:
- Name — Type a unique, identifiable name.
 - Type — Select the calendar type, and enter specific day criteria (according to type).
- For details, see [“Adding special events”](#) on page 7-9 and [“Calendar day selections”](#) on page 7-21.
- Step 3 After naming, selecting the event type, and entering the selection criteria, click **OK**.
The special event is listed in the special events table, with your assigned name.
- Step 4 Click the special event to select it.
- Step 5 In the right-side events column, you can either:
- Click and drag to enter events that override (yet “intermingle” with) normal weekly events.
 - Right-click and select an all-day event or default output value. This negates the normal weekly schedule.
- For more details see [“Event times and output values”](#) on page 7-10.
- Step 6 Continue to add, edit, rename, or delete special events as needed. See [“Right-click menus and other controls”](#) on page 7-10, and [“Special event priorities”](#) on page 7-10.
- Step 7 When special events are like you want, click **Save**.

To review a weekly schedule’s configuration

Use the *read-only* Summary tab to review a weekly schedule’s configuration.

- Step 1 In the [Weekly Scheduler view](#), click the [Summary](#) tab.
The calendar shows the current day with its schedule events, including output and output source.
- Step 2 Click any day on any calendar month to see its schedule events.
- Step 3 If adjustments are necessary, click the [Weekly Schedule](#), [Special Events](#), or [Properties](#) tab as needed, and make changes. Click **Save** when done.

Configuring calendar schedules

Calendar schedules let you globally specify holidays and other special days in the system. For details, see [“About calendar schedules \(holidays\)”](#) on page 7-18.

To configure a calendar schedule

You use the [Calendar Scheduler view](#) to configure a CalendarSchedule.

- Step 1 Double-click the CalendarSchedule component.
The Calendar Scheduler displays, as shown in [Figure 7-27](#) on page 19.
- Step 2 Click the **Add** button for a dialog to add a calendar event.
In the **Add** dialog, specify:
- Name — Type a unique, identifiable name.
 - Type — Select the calendar type, and enter specific day criteria (according to type).
- For details, see [“Calendar day selections”](#) on page 7-21.
- Step 3 After naming, selecting the event type, and entering the selection criteria, click **OK**.
The calendar event is listed in the events table, with your assigned name.
- Step 4 Continue to add, edit, rename, or delete calendar events as needed. See [“Right-click menus and other controls”](#) on page 7-20.
- Step 5 When calendar events are like you want, click **Save**.

Configuring trigger schedules

Trigger schedules provide scheduling control for *actions* of linked components or their child extensions. For details, see [“About trigger schedules”](#) on page 7-23.

To configure a trigger schedule

You use the [Trigger Scheduler view](#) to configure a TriggerSchedule.

- Step 1 Double-click the TriggerSchedule component.
The Trigger Scheduler displays, as shown in [Figure 7-36](#) on page 24.
This view has a calendar side (left) and time picker (right side).
- Step 2 Click the **Add** button for a dialog to add a trigger event.

In the **Add** dialog, specify:

- Name — Type a unique, identifiable name.
- Type — Select the calendar type, and enter specific day criteria (according to type).

For details, see [“Adding trigger events”](#) on page 7-24 and [“Calendar day selections”](#) on page 7-21.

Step 3 After naming, selecting the event type, and entering the selection criteria, click **OK**.

The event is listed in the (calendar-side) events table, with your assigned name.

Step 4 Click an event to select it.

Step 5 In the right-side time picker area, add one or more triggers, as needed.

Note the following:

- By default, a “midnight” trigger may exist (00h:00m); you can delete it if needed.
- Using the [Range option](#), you can add multiple triggers at some repeating interval.
- Triggers apply to *all* trigger events (calendar-side entries).

For more details see [“Adding trigger event times”](#) on page 7-25.

Step 6 Continue to add, edit, rename, or delete trigger events as needed. See [“Right-click menus and other controls”](#) on page 7-25.

Step 7 When trigger events and triggers are like you want, click [Save](#).

Linking schedules

As needed, you link weekly schedules and (if used) trigger schedules to other components to provide scheduling control. For an overview, see [“Schedule component links”](#) on page 7-2.

- [Linking weekly schedules](#)
- [Linking trigger schedules](#)

Linking weekly schedules

You can link the output (Out slot) of a weekly schedule to any component with a like “Status<Type>” input. For example, you might link the output of a [BooleanSchedule](#) to a BooleanWritable point.

- [To link a weekly schedule using the wire sheet](#)
- [To link a weekly schedule using the Nav tree](#)

To link a weekly schedule using the wire sheet

To link a weekly schedule to a component *in the same wire sheet*, do the following:

Step 1 In the wire sheet, mouse over the weekly schedule’s “Out” slot until highlighted.

Step 2 Click and drag the link wire from the weekly schedule to the *bottom* of the target component, and release the mouse button.

A **Link** dialog appears. The source (left) side shows the schedule’s Out slot preselected.

Step 3 In the target (right) side of the **Link** dialog, click on the desired slot of the target component.

Note: By convention, if linking to a priority array input of a writable point, “ln16” is used for schedule level control. However, you can select any available level desired.

Step 4 With both source and target sides selected in the **Link** dialog, click **OK**.

The target point or component is now linked to the weekly schedule.

To link a weekly schedule using the Nav tree

Often, a weekly schedule is in a different container than the target component. In this case, the easiest way to link is using the Nav tree.

Step 1 Expand the Nav tree to locate the source weekly schedule, noting its *name*.

Step 2 Right-click the weekly schedule.
From the **popup** menu select **Link Mark**.

Step 3 Expand the Nav tree to locate the target component.

Step 4 Right-click the target component.
From the **popup** menu select **Link From “<weekly schedule name>”**.

A **Link** dialog appears.

Step 5 In the source (left) side of the **Link** dialog, click the schedule’s “Out” slot.

Step 6 In the target (right) side of the **Link** dialog, click the desired slot of the target component.

Note: By convention, if linking to a priority array input of a writable point, “ln16” is used for schedule level control. However, you can select any available level desired.

- Step 7 With both source and target sides selected in the **Link** dialog, click **OK**.
The target point or component is now linked to the weekly schedule.

Linking trigger schedules

You can link the output (Trigger slot) of a TriggerSchedule to an *action* (or topic) of any component (point or extension) with such a slot. For example, you might link a trigger schedule to the “ResetChangeOf-StateCount” action of a DiscreteTotalizerExt child of BooleanPoint.

Note: Often, you link trigger schedules to point extensions, not directly to a parent container point or component. In this case, before linking you could first composite the target container component, selecting to expose the extension's action up in the parent point. Although optional, this can help reusability and link clarity. For details, see [“About composites”](#) on page 3-23.

Note: Target actions or topics, unlike target property slots, can accept multiple link sources. In some cases, you may wish to link the “Trigger Missed” slot of the trigger schedule to the same target. For details, see [“Trigger Missed slot”](#) on page 7-26.

You can use either of these methods to link trigger schedules:

- [To link a trigger schedule using the wire sheet](#)
- [To link a trigger schedule using the Nav tree](#)

To link a trigger schedule using the wire sheet

To link a trigger schedule to a component *in the same wire sheet*, do the following:

- Step 1 In the wire sheet, mouse over the trigger schedule's “Trigger” slot until highlighted.
- Step 2 Click and drag the link wire from the trigger schedule to the *bottom* of the target component, and release the mouse button.
- A **Link** dialog appears. The source (left) side has the trigger schedule's Trigger slot preselected
- Step 3 In the target (right) side of the **Link** dialog, click on the desired slot of the target component.
- Step 4 With both source and target sides selected in the **Link** dialog, click **OK**.
The target component (point or extension) is now linked to the trigger schedule.

To link a trigger schedule using the Nav tree

Often, a trigger schedule is in a different container than the target component. In this case, the easiest way to link is using the Nav tree.

- Step 1 Expand the Nav tree to locate the source trigger schedule, noting its *name*.
- Step 2 Right-click the trigger schedule.
From the **popup** menu select **Link Mark**.
- Step 3 Expand the Nav tree to locate the target component.
(In some cases, the target may be an extension of a point or other component.)
- Step 4 Right-click the target component.
From the **popup** menu select **Link From “<target schedule name>”**.
A **Link** dialog appears.
- Step 5 In the source (left) side of the **Link** dialog, click the trigger schedule's “Trigger” slot.
- Step 6 In the target (right) side of the **Link** dialog, click the desired *action* of the target component.
- Step 7 With both source and target sides selected in the **Link** dialog, click **OK**.
The target component (point or extension) is now linked to the trigger schedule.

Importing schedules

If the station is part of a *multi-station* NiagaraAX network, you can import NiagaraAX schedule components between stations. You do this using the NiagaraAX driver architecture, specifically, views of the *Schedules extension* under a NiagaraStation (device-level) component.

If the Bacnet driver is used, you can also *import* BACnet Schedules and Calendars (as NiagaraAX schedule components) from a BACnet device. The same basic architecture and methods are used.

Note: Under a BacnetNetwork, you can also export NiagaraAX schedules to a specific BACnet device (to configure existing BACnet Schedules and Calendar objects).

For a brief overview, see [“Schedule exports and imports \(master/slave\)”](#) on page 7-4. For more complete details, refer to the *Drivers Guide*, in sections “About the Schedules extension” and “Station Schedules import/export notes”.

Importing NiagaraAX schedules or Bacnet Schedules

Often, you import a NiagaraAX schedule when *working in a JACE station*, where the source (master) schedule resides in a remote Supervisor station. BACnet Schedules and Calendars are also typically imported into a JACE station (unless a BACnet Supervisor).

To import a NiagaraAX schedule or BACnet Schedule or Calendar

To import a NiagaraAX schedule or BACnet Schedule or Calendar, do the following:

- Step 1 In the Nav tree, expand the **Drivers** folder of the station to receive the imported schedule.
- Step 2 Expand the network, either **NiagaraNetwork** or **BacnetNetwork**.
- Step 3 Expand the device that contains the source schedule (**NiagaraStation** or **BacnetDevice**).
In the Nav tree, the device's children include: Points, Histories, Alarms, and Schedules.
- Step 4 Double-click **Schedules**.
The Niagara or Bacnet Schedule Import Manager displays. Any existing schedules (already imported) from this device are listed.
- Step 5 Click **Discover**.
The manager view splits into two panes and a schedule discovery job is started. When complete, schedules available for import appear in the top pane.
- Step 6 In the top pane, click to select one or more schedules, then click **Add**.
An **Add** dialog allows you to edit the name of the schedule (as it appears in this station) and several other properties. Refer to the section "Schedule Import properties" in the *Drivers Guide* for details.
- Step 7 After editing name and properties to suit, click **OK**.
NiagaraAX schedule components are created for the selected items, reside under that device's Schedules extension. You can link into station logic like any other schedules, but cannot configure these imported schedules (add, delete, or change events or other properties).

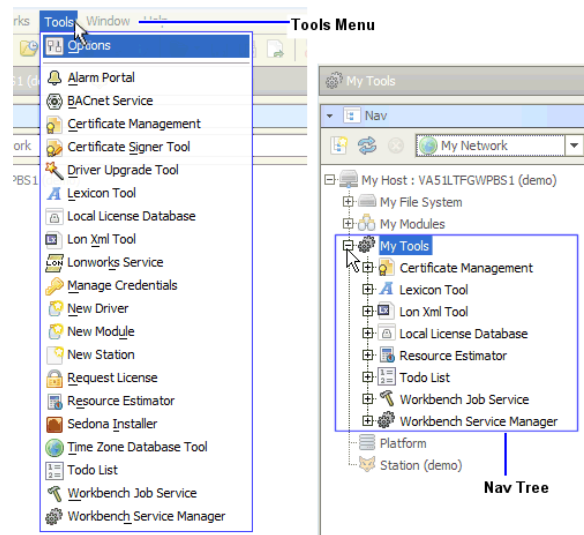
CHAPTER 8

About Workbench tools

This section describes some of the standard “tools” available in Workbench. These tools provide Workbench interfaces 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, as shown in [Figure 8-1](#), navigational links to some of the tools also appear in the nav side bar and in the nav container views.

Note: The nav side bar and nav container views do not display the tool icons until you initially open them using the **Tools** menu.

Figure 8-1 Workbench tools



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 of Workbench. Tools, such as the Alarm Portal, BACnet service, Lonworks service, and the Workbench Job Service are available in Workbench 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”](#) on page 8-3.
- [Bacnet Service](#)
Explains the options that appear in the Bacnet service. Refer to [“Bacnet Service”](#) on page 8-4.

- **Certificate Management**
Explains the Certificate Management view of the SSL Toolset. You use this view to create PKI (Public Key Infrastructure) digital certificates; to create Certificate Signing Requests (CSRs); and to import and export keys and certificates to and from the Workbench key stores. This view appears only if the platform-connected host is licensed for the SSL Toolset and has the necessary modules installed. Refer to the document "*NiagaraAX SSL Connectivity Guide*" for complete details.
- **Certificate Signer Tool**
Explains the options in the Certificate Signer Tool. You use this tool to sign intermediate digital certificates. This tool appears only if the platform-connected host is licensed for the SSL Toolset and has the necessary modules installed. Refer to the document "*NiagaraAX SSL Connectivity 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. Currently, this tool applies if upgrading from the previous **Video Driver** to the newer **Video Framework** driver (older modules at "dev" prefix, new modules with "n" prefix). For more details, refer to the section "Upgrading to Video Framework" in the *NiagaraAX Video Framework Guide*.
- **Lexicon Tool**
Explains the options that appear in the lexicon tool. For general information, see "**Lexicon Tool**" on page 8-5. For detailed information, see the *Lexicon Guide*. For information about the lexicon installer, refer to the "Lexicon Installer" section of the *Platform Guide*.
- **Local License Database**
Explains the Workbench License Manager view. Refer to "**Local License Database**" on page 8-6.
- **Lonworks Service**
Explains the options that appear in the lonworks service manager. Refer to "**Lonworks Service**" on page 8-8.
- **Lon Xml Tool**
Describes the Lon Xml Create view. Refer to "**Lon Xml Tool**" on page 8-7.
- **Manage Credentials**
Explains the options that appear in the credentials manager. Refer to "**Credentials manager**" on page 8-8.
- **New Driver wizard**
Explains the options that appear in the new driver wizard. Refer to "**New Driver wizard**" on page 8-9.
- **New Module wizard**
Explains how to modify the attributes of a new module. Refer to "**New Module wizard**" on page 8-10.
- **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**" on page 8-10.
- **Request License**
Explains the options that appear in the license request dialog box. Refer to "**Request License**" on page 8-12.
- **Resource Estimator**
Explains the options that appear in the resource estimator dialog box. Refer to "**Resource Estimator**" on page 8-12.
- **Sedona Installer**
Provides a wizard to install a "Sedona Framework TXS bundle" distribution, which enables Workbench to work with Sedona powered devices, including integration into NiagaraAX stations.
Note: *If a Sedona bundle is installed, additional Workbench tools (views) related to Sedona will also be available. These tools include a Sedona - New App tool, Sedona Device Simulator, Sedona Jennic - New Wireless Adapter tool, Sedona Jennic - Serial Port Tool, and Sedona Manifest Manager. These tools are covered in Sedona-related documents, including online Workbench help jars that are installed with the Sedona Framework TXS bundle.*
For more details, see the *NiagaraAX Sedona Framework TXS-1.2 Installer Guide*.
- **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**" on page 8-13.
- **Todo List**
Explains the options that appear in the todo list dialog box. Refer to "**Todo List**" on page 8-15.

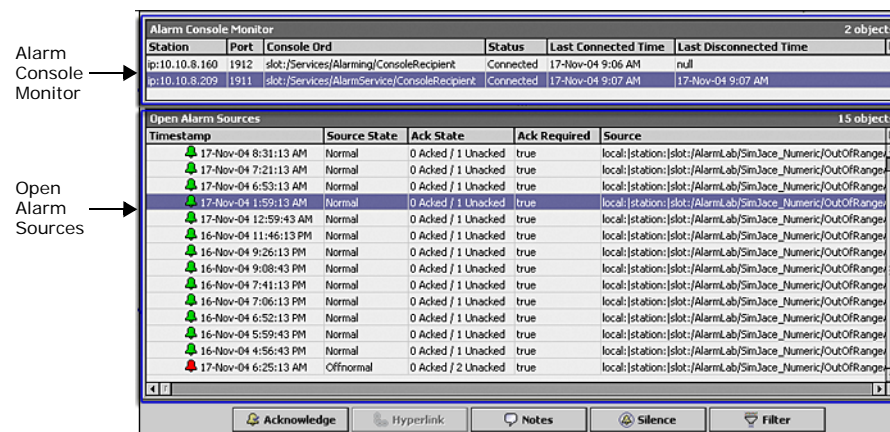
- [Workbench Job Service](#)
Explains the options that appear in the Workbench job service view. Refer to “[Workbench Job Service](#)” on page 8-15.
- [Workbench Service Manager](#)
Explains the options that appear in the service manager screen. Refer to “[Workbench Service Manager](#)” on page 8-15.

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 in [Figure 8-2](#), 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](#)” on page 5-37.

Figure 8-2 Alarm portal



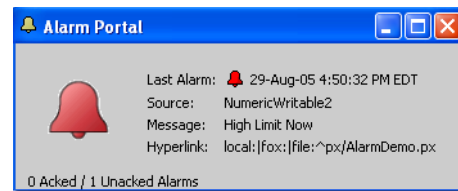
- **Alarm Console Monitor**
This pane displays information about the consoles that are being monitored and contains the following information areas:
 - Title bar
Displays “Alarm Console Monitor” title on the left and the number of consoles that are connected to the alarm portal on the right side of the 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](#)” on page 5-37.
 - 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”](#) on page 5-3) is set to allow hyperlinking. 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”](#) on page 5-36 for more details about alarm record notes).
 - 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”](#) on page 5-37 for more details about the alarm filter dialog box.

The alarm portal tool, when enabled, also places the alarm icon in your system tray and an alarm popup window, as shown in [Figure 8-3](#).

You can set options for the alarm portal in the Tools Options menu, as described in [“Alarm Portal options”](#) on page 2-28.

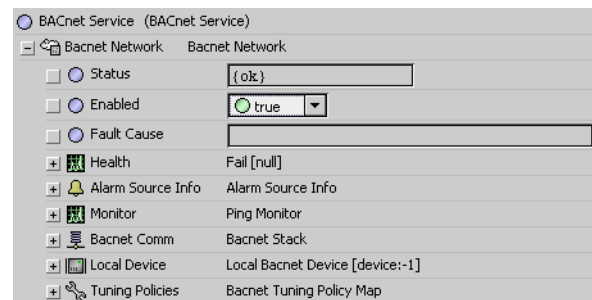
Figure 8-3 Alarm portal popup window



Bacnet Service

Figure 8-4 shows 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 8-4 Bacnet service property sheet view



Note: You can use this Workbench Tools service to configure a local Bacnet device in the same way as 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](#)” on page 1-20. For detailed information on using the Lexicon Tool, see the *NiagaraAX Lexicon Guide*. For details on installing (edited) file-based lexicons in remote JACE platforms, see the *Platform Guide* section “Lexicon Installer”. For details on installing (new/edited) module-based lexicons in remote JACE platforms, see the *Platform Guide* section “Software Manager”.

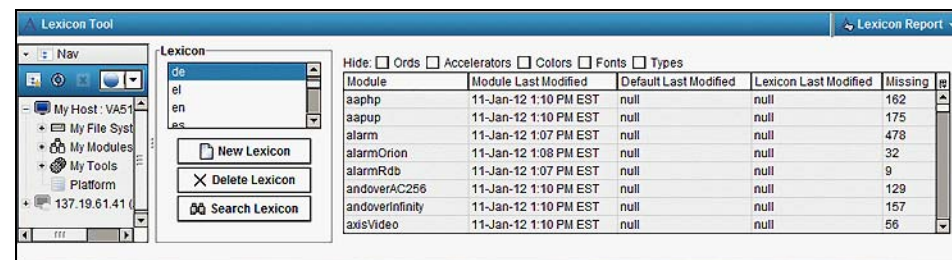
For an overview of the Lexicon Tool views, see the following sections:

- [Lexicon Report view](#)
- [Lexicon Editor view](#)
- [Lexicon Module Builder view](#)

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** box 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, as shown in [Figure 8-5](#).

Figure 8-5 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 *NiagaraAX Lexicon Guide* section “Lexicon components”.

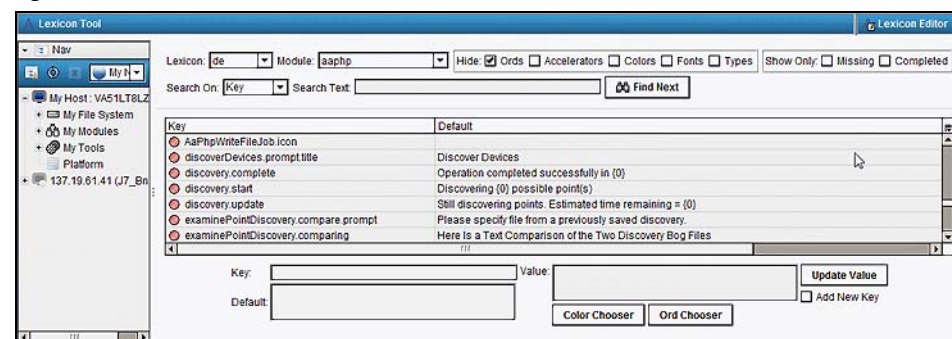
This view lets you add (new) lexicons, delete unwanted lexicons, or search all available lexicon property values that contain a given text. Additionally, you can double-click any module row in the table to launch the [Lexicon Editor view](#), which shows the contents of that lexicon file.

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

Lexicon Editor view

The Lexicon Editor view (shown in [Figure 8-6](#)) 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 8-6 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

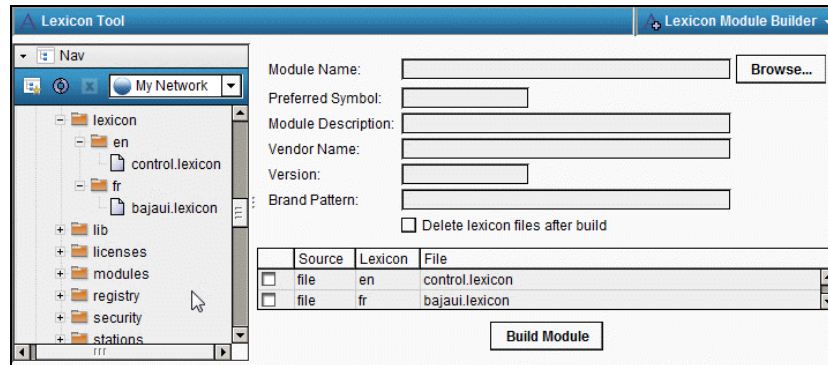
to modify the value of the selected key (or add a value for a new key). Starting in Workbench 3.7 the **Value** field has associated **Color Chooser** and **Ord Chooser** buttons. These options allow you to browse for a specific color or ORD element. Update the table with the **Value** field entry by 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 *NiagaraAX Lexicon Guide*.

Lexicon Module Builder view

Starting in Workbench 3.7, a Lexicon Module Builder view (shown in Figure 8-7) is available. The lexicon module builder 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 8-7 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, as shown in Figure 8-7.

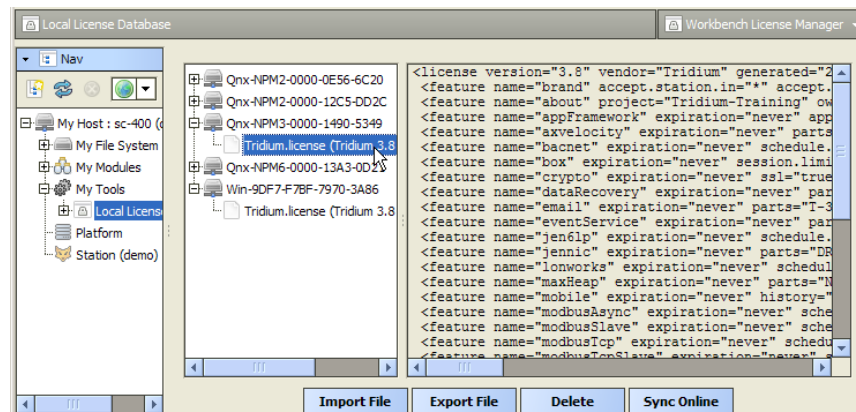
On completion of the build, you will see a confirmation message indicating that the module was constructed and placed in the !modules folder.

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

Local License Database

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

Figure 8-8 Workbench License Manager



As shown in Figure 8-8, 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:

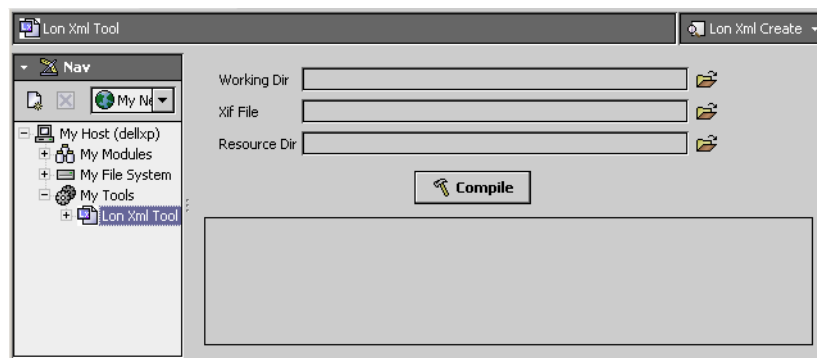
- **Import File**
Always available, this allows you to add license file(s) from a local license file or license archive (.lar) file.
- **Export File**
Always available, this allows you to save all licenses (or any selected licenses) locally, as a license archive file.
- **Delete**
This allows you to 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 NiagaraAX-3.x Platform Guide.

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 (Figure 8-9) provides the necessary input fields.

Figure 8-9 Lon Xml Create tool view



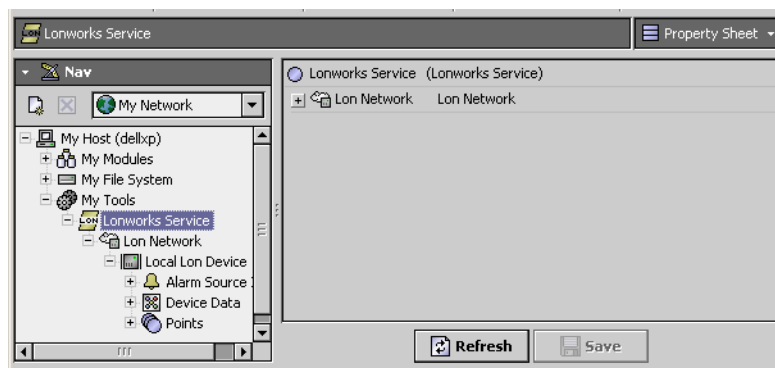
Refer to the *NiagaraAX-3.x Lonworks Guide* for details about using this view and about the following:

- Need for custom Lon Xml files
- Lon Xml file overview
- Lon Xml creation
- Storing .lnml files
- Differential temperatures and lnml 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 8-10).

Figure 8-10 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.

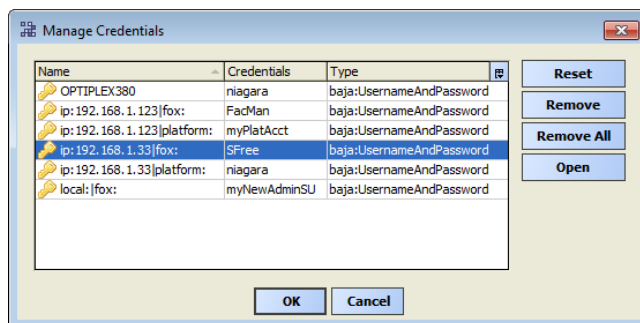
For this reason, do not use this tool to access a Lonworks network already managed by a station, but instead open that station, and access its LonNetwork.

Refer to the *NiagaraAX-3.x Lonworks Guide* for more details.

Credentials manager

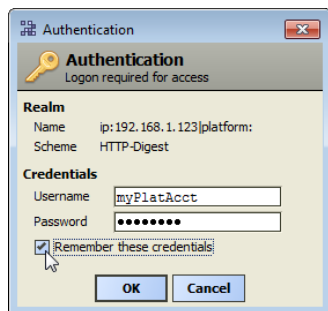
The **Manage Credentials** tool, or “Credentials Manager” provides a dialog box to access and open any previously “remembered” (cached) connections from your Workbench, including both platform and station connections.

Figure 8-11 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, as shown in Figure 8-11. Starting in AX-3.7, any *initial* platform or station connection, this option is *disabled* (unchecked) by default.

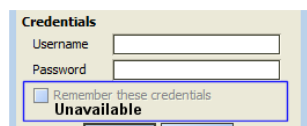
Figure 8-12 Authentication (Remember credentials)



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 (Figure 8-12) 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 (Figure 8-11). This way, that platform or station connection always requires full entry of both user name and password.

Note: Starting in AX-3.7, you can globally disable user credentials caching in Workbench, via **Tools > Options**, in the **General** menu. When "Allow User Credentials Caching" is set to *false*, the "Remember these credentials" checkbox remains unavailable (dimmed) in any Authentication dialog.



For related details, see "General options" on page 2-26.

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 in Figure 8-13.

Figure 8-13 New driver wizard

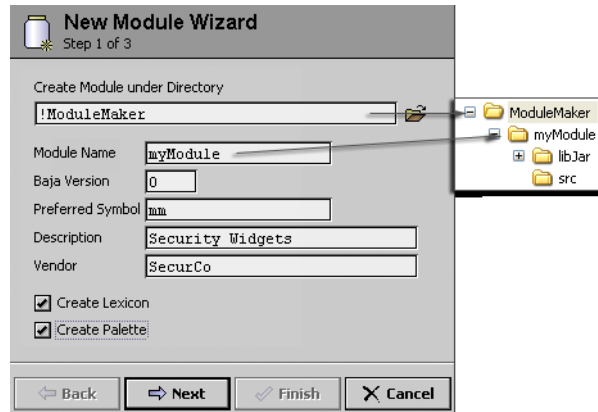


This wizard is provided as a convenient way to perform some of the steps necessary for creating a driver module. The four wizard dialog boxes allow driver developers to specify some basic driver module options and finish with the wizard creating a set of folders and files under a station directory that you assign during the wizard process. Also see the *NiagaraAX-3.x Developer Guide* for more detailed information about building modules.

New Module wizard

The New Module Wizard is available when you select it from the **Tools > New Module** from the main menu. The wizard presents a series of dialog boxes, as shown in Figure 8-14, that help you create and save a new module.

Figure 8-14 New module wizard creates folders under working directory

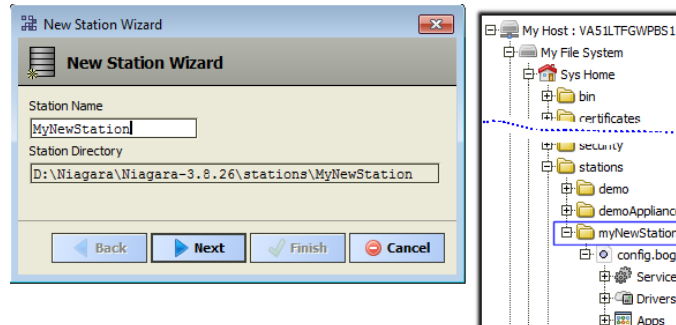


This wizard creates a set of folders and files under a station directory that you assign during the wizard process. After creating these files, you need to finish the process, as described in “[Working with modules](#)” on page 9-24. Also see the *NiagaraAX-3.x Developer Guide* for more detailed information about building modules.

New Station wizard

The New Station wizard is available by selecting **Tools > New Station** from the main menu. The wizard provides two dialogs to help you create and save a new station.

Figure 8-15 Initial New Station Wizard dialog

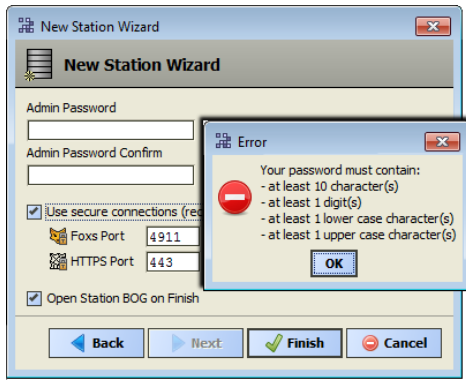
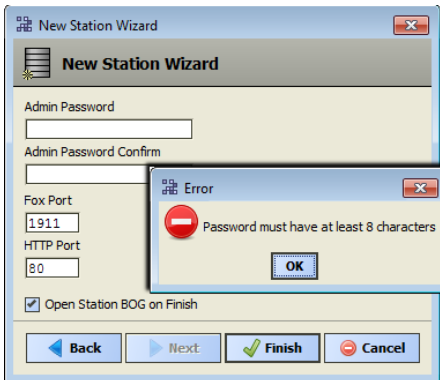


In the first dialog, you enter the name of the station. When the wizard completes, the station folder will have this name, as shown above in Figure 8-15. Fields in this first **New Station** wizard dialog are:

- **Station Name**
The case-sensitive station name, which must begin with a letter. It is best to keep station name short, and use a station “display name” if a longer name with spaces and/or other characters are needed.
- **Station Directory**
Read-only field specifies the location where the station folder is created (under the “Stations” folder).

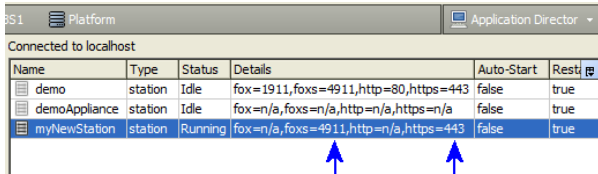
Click **Next** to continue to the next/final wizard dialog, shown in Figure 8-16.

Figure 8-16 Second (final) New Station Wizard dialog (AX-3.8 and AX-3.7)

AX-3.8	AX-3.7u1 and AX-3.7
	
In AX-3.8, Admin Password (user “admin”) must meet minimum “default strong password” requirements, as shown above.	In AX-3.7, Admin Password (user “admin”) has lesser “default strong passwords” needs. Cannot be either all letters or all digits.
By default, the “Use secure connections” checkbox is enabled. “Foxs Only” and “Https Only” will be enabled in the station’s FoxService and Webservice, respectively (for SSL). Ports shown are default Foxs and Https ports. If needed you can uncheck (clear) to create the station with standard (non-SSL) settings.	By default, the station is created with standard (non-SSL) configuration of the FoxService and Webservice. Ports shown are default Fox and Http ports.

Wizard differences between AX-3.8 and AX-3.7 are summarized above, with more details as follows:

- In order to complete the wizard (and create the station), you must enter a password for the “admin” user that meets “default strong password” requirements, differing between AX-3.8 and AX-3.7. If needed you can change this later in the station (regardless if AX-3.8 or AX-3.7).
- In AX-3.8, the New Station wizard defaults to “Use secure connections (recommended)”, such that any station created with the New Station Wizard is pre-configured for “SSL only” Fox client access and Web (HTTP) access. This is different that when using the **New Station** wizard in AX-3.7u1 or earlier, where after creating the new station you must *enable* SSL access.



Name	Type	Status	Details	Auto-Start	Resti
demo	station	Idle	fox=1911,foxs=4911,http=80,https=443	false	true
demoAppliance	station	Idle	fox=n/a,foxs=n/a,http=n/a,https=n/a	false	true
myNewStation	station	Running	fox=n/a,foxs=4911,http=n/a,https=443	false	true

When you start a new AX-3.8 station created with the **New Station** wizard defaults, you can see this in the “station status” area of the **Application Director**, as shown above.

Fields in this second **New Station Wizard** dialog are:

- **Admin Password**
The password for the “built-in” station user named admin. Must meet minimum “strong password” requirements for AX-3.8 or AX-3.7, which differ (see [Figure 8-16](#)).
- **Admin Password Confirm**
Re-enter the same password again (must match).
- **Foxs Port (AX-3.8 default) or Fox Port (AX-3.7, or AX-3.8 option)**
Lets you assign the port to be used for Foxs (Fox SSL) or Fox communications. Default ports are:
 - **Foxs Port:** 4911
 - **Fox Port:** 1911
 After new station creation, this can be changed (if needed) from the station’s Config, Drivers, NiagaraNetwork, FoxService component’s property sheet.
- **HTTPS Port (AX-3.8 default) or HTTP Port (AX-3.7, or AX-3.8 option)**
Lets you assign the port to be monitored by the station’s Webservice for HTTP connections. Default ports are:
 - **HTTPS Port:** 443

- **HTTP Port:** 80

After new station creation, this can be changed (if needed) from the station's Config, Services, Web-Service component's property sheet.

For related topics, see the following:

- [“About built-in users”](#), including [“User admin”](#) on page 6-6.
- [“web-WebService”](#) on page 10-28.
- [“About the FoxService”](#) in the *NiagaraAX Drivers Guide*.
- *NiagaraAX SSL Connectivity Guide*

Request License

The **“Request License”** item on the Workbench **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 [Figure 8-17](#).

Figure 8-17 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 NiagaraAX. 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 the licensing, see *“Appendix A. License Tools and Files”* in the *NiagaraAX-3.x 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 in [Figure 8-18](#).

Figure 8-18 Resource estimator view

Additional Multiplier	Medium (30%)		
Device Networks	0	$\times 100,000 + \text{Multiplier}$	0.000 KRU
Devices	0	$\times 5000 + \text{Multiplier}$	0.000 KRU
Proxy Points	0	$\times 250 + \text{Multiplier}$	0.000 KRU
Program Components	0	$\times 3000 + \text{Multiplier}$	0.000 KRU
Histories	Count * (250 + Size/10)		
Config. 1	Count: 0	Capacity: Record Count	Size: 0 0.000 KRU
Config. 2	Count: 0	Capacity: Record Count	Size: 0 0.000 KRU
Config. 3	Count: 0	Capacity: Record Count	Size: 0 0.000 KRU
Config. 4	Count: 0	Capacity: Record Count	Size: 0 0.000 KRU
Config. 5	Count: 0	Capacity: Record Count	Size: 0 0.000 KRU
AlarmDb Capacity	0	$\times 1$	0.000 KRU
Total			0.000 KRU



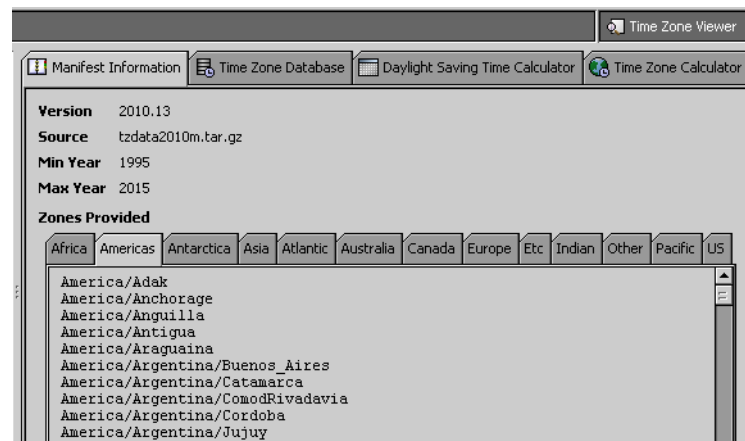
Time Zone Database Tool

The Time Zone Database Tool, available since AX-3.5, 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 the Platform Guide appendix “Time Zones and NiagaraAX”.

Figure 8-19 Time Zone Database Tool in Workbench AX-3.5 or later



As shown in Figure 8-19, the Time Zone Database Tool has four available tabs.

- [Manifest Info](#)
- [Time Zone Database](#)
- [Daylight Savings Time Calculator](#)
- [Time Zone Calculator](#)

Manifest Info

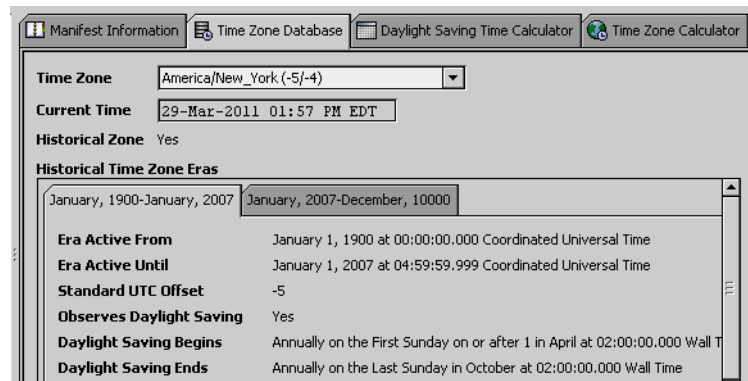
This tab in the [Time Zone Database Tool](#) shows you the `timezone.jar` file version, and which “Olsen Time Zone DB” version the file is based upon. The “Min Year” says how far back historically time zones are guaranteed to behave correctly. Conversely, “Max Year” is how far forward behavior should 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 8-20 Time Zone Database tab in Workbench Time Zone Database Tool



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

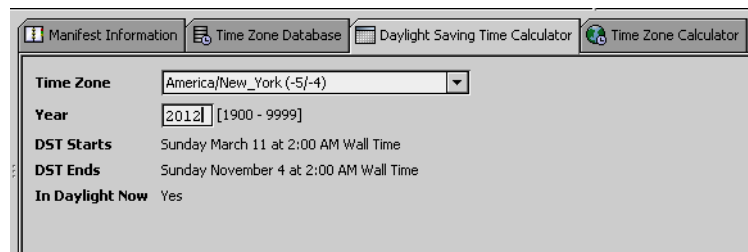
Is this a “historical zone”? (meaning have time zone “rules” changed for this zone at some point?)

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 8-21 Daylight Savings Time Calculator tab in Workbench Time Zone Database Tool



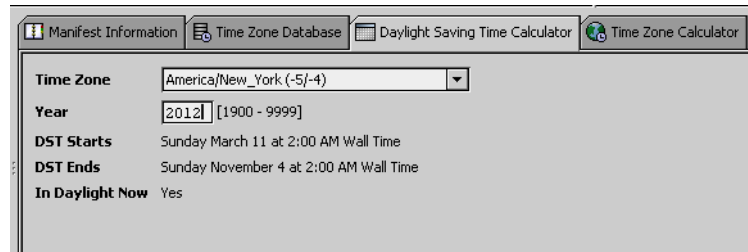
To use, select a time zone and type in a year in the **Year** field.

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 8-22 Time Zone Calculator tab in Workbench Time Zone Database Tool

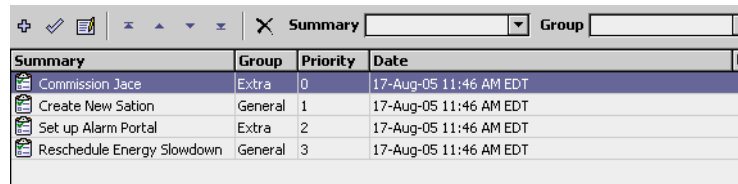


To use, select a source time zone and a target time zone, and specify a source time (initial default is the current time).

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 in Figure 8-23.

Figure 8-23 Todo list view



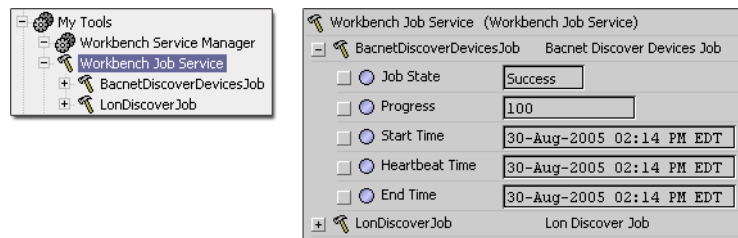
Summary	Group	Priority	Date
Commission Jace	Extra	0	17-Aug-05 11:46 AM EDT
Create New Station	General	1	17-Aug-05 11:46 AM EDT
Set up Alarm Portal	Extra	2	17-Aug-05 11:46 AM EDT
Reschedule Energy Slowdown	General	3	17-Aug-05 11:46 AM EDT

The Todo list is a tabular view with standard table controls and options, as described in “[Table controls and options](#)” on page 2-18. 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 in Figure 8-24, 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 8-24 Job service nav tree and property sheet view

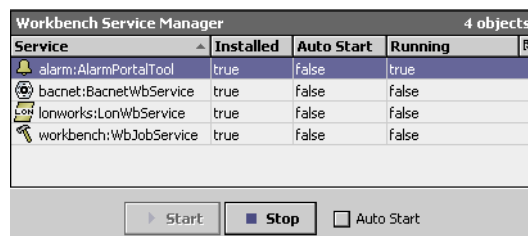


Note: *Workbench* jobs are jobs that are initiated and run under the Workbench - not under a station. Jobs that are run under a station are monitored and displayed in the Station Job Service (under the station Services node in the nav side bar).

Workbench Service Manager

The service manager view is available when you select it from the **Tools** menu. The service manager view, shown in Figure 8-25, is used to manage the life cycle and configuration of all services.

Figure 8-25 Workbench service manager view



Service	Installed	Auto Start	Running
alarm:AlarmPortalTool	true	false	true
bacnet:BacnetWbService	true	false	false
lonworks:LonWbService	true	false	false
workbench:WbJobService	true	false	false

The Workbench service manager is a tabular view with standard table controls and options, as described in “[Table controls and options](#)” on page 2-18. From this view you can Start, Stop, or configure any of the listed services to Auto-Start.

CHAPTER 9

Performing Workbench tasks

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

Common Workbench tasks include the following:

- [Starting and exiting Workbench](#)
- [Getting started with stations](#)
- [Using the side bar pane](#)
- [Using bookmarks](#)
- [Using the help side bar](#)
- [Using the nav side bar \(tree\)](#)
- [Using the palette side bar](#)
- [Using the view pane](#)
- [Editing components](#)
- [Using alarms](#)
- [Creating a Live History Chart](#)
- [Working with modules](#)

Starting and exiting Workbench

Tasks involved with starting and exiting Workbench, include the following:

- [To start Workbench](#)
- [To close a Workbench window](#)
- [To exit Workbench](#)

Note: *Start procedures given here are for the “typical” usage of Workbench. You may have other Windows Start menu shortcuts, Desktop shortcuts, or Windows 8 “Start Tiles” installed for other Niagara applications not mentioned here. Refer to the NiagaraAX 3.8 Windows Installation document for summary information.*

To start Workbench

To start Workbench do one of the following:

- From the Windows **Start** menu, choose either:
 - **Programs > Niagara 3.x.xx > Workbench 3.x.xx**
The **Niagara Workbench** GUI window appears.
 - **Programs > Niagara 3.x.xx > Workbench (Console)**
A **Workbench (Console)** window appears, after which a separate **Niagara Workbench** GUI window appears. The console window can be useful to capture debug messages from the client Workbench application, or to exit in the case of a “hung client connection”.
Note: *If you close the Workbench console window, Workbench also immediately closes—any unsaved changes are lost. Be sure to save changes before closing the associated console window.*
- From a Niagara **Console** window (**Programs > Niagara 3.x.xx > Console**), at the command prompt, enter either:
 - **wb_w**
The **Niagara Workbench** GUI window appears. The console window and Workbench window are independent, meaning if you close or exit one, the other remains open.

- **wb**

A **Workbench (Console)** window appears, after which a separate **Niagara Workbench** GUI window appears. The console window can be useful to capture debug messages from the client Workbench application, or to exit in the case of a “hung client connection”.

***Note:** If you close the Workbench console window, Workbench also immediately closes—any unsaved changes are lost. Be sure to save changes before closing the associated console window.*

Note you can also open a console *within* the Workbench (GUI) window. See “[About the console](#)” on page 2-13.

- If **Desktop Shortcuts** were installed (and/or if AX-3.8 with Windows 8 **Start Screen Tiles** were installed), double-click the following shortcut, or click the following tile:
 - Workbench 3.x .xx
 The **Niagara Workbench** GUI window appears.

To close a Workbench window

***Note:** Closing all Workbench windows will exit the Workbench GUI.*

To close a Workbench window do one of the following:

- From the **File** menu, select the **Close** command.
The Workbench window will close.
- From the **Niagara Workbench** menu (located in the top left corner of the window title bar), select the **Close** command.
The Workbench window will close.
- Click the close icon (located in the top right corner of the window title bar).
The Workbench window will close.

To exit Workbench

You can exit Workbench by using the Exit command (which will close all Workbench windows) or by closing all Workbench windows.

To exit Workbench, do one of the following:

- From the **File** menu, select the **Exit** command.
The Workbench GUI will close.
- Close all Workbench windows.
The Workbench GUI will close when the final Workbench window is closed.

Getting started with stations

In order to communicate with a station, the station must be running and accessible. The following procedures provide information about basic tasks that are required to start, stop, and communicate with stations:

- [To open a platform using Workbench GUI](#)
- [To connect to a platform using Workbench GUI](#)
- [To close a platform using Workbench](#)
- [To disconnect from a platform using Workbench](#)
- [To start a station using Workbench](#)
- [To stop a station using Workbench](#)
- [To open a station](#)
- [To close a station using Workbench](#)
- [To connect to a station using Workbench](#)
- [To disconnect from a station](#)





To open a platform using Workbench GUI

To open a platform, do the following:



- Step 1 From the menu bar, select **File > Open > Open Platform**.
The popup **Open Platform** dialog appears, or if AX-3.8, the **Open Platform with SSL** dialog.

***Note:** The default in AX-3.8 has changed to **Open Platform with SSL**.*

- Step 2 Complete the fields in the **Open Platform** or **Open Platform with SSL** dialog box as follows:

- **Type**
Select either  **Platform SSL Connection** or  **Platform Connection**, as appropriate. If you change, the default **Port** changes between 5011 (SSL) to 3011 (non-SSL).
 - **Host**
Select the **IP** option and type in the IPv4 address of the host platform that you wish to access.
 - **Port**
Verify the port shown, with the default as either 3011 (non-SSL) or 5011 (SSL). If you know the remote host is using a different (non-default) platform daemon port, enter it here.
- Step 3 Click the **OK** button. If you chose **Open Platform** (non-SSL), go to [Step 4](#).
If you chose **Open Platform with SSL**, different results are possible:
- If the Niagara platform is not enabled for SSL, an error results with “Cannot connect. Ensure server is running on specified port”. Retry from [Step 1](#), changing the **Type** to **Platform Connection**.
 - If the Niagara platform is enabled for SSL, but you have not yet installed a certificate for that host, a popup **Identity Verification** dialog with certificate details appears. In most cases, you click **Accept** to proceed to the next step.
 - An **Authentication** popup dialog appears. Go to the next step.
- Step 4 In the popup **Authentication** dialog, enter platform credentials
- Note:** *Be sure to use the platform username and password – not credentials for any station user.*
- **Username**
Enter the username for the platform.
 - **Password**
Enter the password for the platform.
- If the “Remember these credentials option” is available, you can select it, so that Workbench “remembers” these platform credentials. However, this can pose a future security risk; thus this option may be disabled.
- Step 5 Click **OK** after entering credentials.
- If the platform credentials were correct, the platform is connected, and the platform icon appears in the side bar pane and the default Nav Container View appears in the view pane. Where platform icons are:
-  Platform *or*
 -  Platform SSL
- If the platform credentials were not correct, you are prompted to enter them again.
For related details, refer to “Platform overview” in the NiagaraAX *Platform Guide*.

To connect to a platform using Workbench GUI

- Note:** *You cannot connect to a platform unless it is “open”. An open platform may be visible in your Workbench nav side bar, but appear dimmed (be disconnected). If there is no platform icon in the nav side bar, you need to open the platform before connecting (see “[To open a platform using Workbench GUI](#)” on page 9-2). To connect to a disconnected platform, do the following:*
- Step 1 In the nav side bar (refer to “[Using the nav side bar \(tree\)](#)” on page 9-9) right-click on platform icon and select the **Connect** command from the **popup** menu. Or, simply double-click the platform icon. The **Authentication** popup dialog appears.
- Step 2 Enter the credentials information as follows:
- Note:** *Be sure to use the platform username and password – not credentials for any station user.*
- **Username**
Enter the username for the platform.
 - **Password**
Enter the password for the platform.
- If the “Remember these credentials option” is available, you can select it, so that Workbench “remembers” these platform credentials. However, this can pose a future security risk; thus this option may be disabled.
- Step 3 Click **OK** after entering credentials.
- If the platform credentials were correct, the platform is connected, and the platform icon appears in the side bar pane and the default Nav Container View appears in the view pane. Where platform icons are:
-  Platform *or*
 -  Platform SSL
- If the platform credentials were not correct, you are prompted to enter them again.
For related details, refer to “Platform overview” in the NiagaraAX *Platform Guide*.

To close a platform using Workbench

To close a platform, do the following:

- Step 1 In the nav side bar (refer to “[Using the nav side bar \(tree\)](#)” on page 9-9, if necessary) right-click on the platform icon and select the **Close** command from the **popup** menu.
The **Close** dialog box appears.
- Step 2 In the **Close** dialog box, click **OK** to close the platform.
The platform is closed and the platform icon is removed from the nav side bar.

To disconnect from a platform using Workbench


To disconnect from a platform, do the following:

- Step 1 In the nav side bar (refer to “[Using the nav side bar \(tree\)](#)” on page 9-9) right-click on platform icon and select the **Disconnect** command from the **popup** menu.
The popup **Disconnect** confirmation dialog box appears.
- Step 2 In the **Disconnect** dialog box, click **OK** to disconnect.
The platform is disconnected and the platform icon is dimmed.

To start a station using Workbench

Note: Most station–platform functions can be performed from the Application Director platform view of a host; however, you must connect to the platform where a station is hosted before you can perform these tasks.

To start a station using Workbench, do the following:


- Step 1 Connect to the station's platform (see “[To connect to a platform using Workbench GUI](#)” on page 9-3 for details, if necessary).
- Step 2 In the nav side bar, expand the platform node in the nav tree by clicking on the plus sign (+) next to the platform icon in the tree.
The contents of the platform node displays in the tree.
- Step 3 In the nav side bar, double click the  Application Director.
- Step 4 In the **Application Director** view, select the desired station. It becomes highlighted in the table.
- Step 5 In the right-side area of this view, select the following options, as desired:
- Select the **Auto-Start** option if you want the station to automatically start any time the station host computer is re-booted. Clear the **Auto-Start** option to disable auto-start.
 - Select the **Restart on Failure** option if you want the station to automatically attempt a restart any time it fails. Clear the **Restart** option to disable automatic restart.
- Step 6 In the **Application Director** view, do one of the following:
- Select the desired station in the station table and click **Start**.
 - Right click the desired station in the station table and select **Start** from the **popup** menu.
- The station starts.

For details on this platform view, refer to “Application Director” in the NiagaraAX Platform Guide.

To stop a station using Workbench





Note: Most station–platform functions can be performed from the Application Director platform view of a host; however, you must connect to the platform where a station is hosted before you can perform these tasks.

To stop a station using Workbench, do the following:

- Step 1 Connect to the station's platform (see “[To connect to a platform using Workbench GUI](#)” on page 9-3 for details, if necessary).
- Step 2 In the nav side bar, expand the platform node in the nav tree by clicking on the plus sign (+) next to the platform icon in the tree.
The contents of the platform node displays in the tree.
- Step 3 In the nav side bar, double click the  Application Director.
- Step 4 In the **Application Director** view, select the desired station. It becomes highlighted in the table.
- Step 5 Do one of the following:
- With the desired station selected station table, click **Stop**.
 - Right click the desired station and select **Stop** from the **popup** menu.
- The station stops.

To open a station

To open a station, do the following:

- Step 1 From the **File** menu, select **Open > Open Station (Fox)**.
The popup **Open Station** dialog appears, or if AX-3.8, the **Open Station with SSL** dialog.
- Note:** *The default in AX-3.8 has changed to **Open Station with SSL**.*
- Step 2 Complete the fields in the **Open Station** or **Open Station with SSL** dialog box as follows:
- **Type**
Select either  **Station SSL Connection** or  **Station Connection**, as appropriate. If you change, the default **Port** changes between 4911 (SSL) to 1911 (non-SSL).
 - **Host**
Select the **IP** option and type in the IPv4 address of the host platform that you wish to access.
 - **Port**
Verify the port shown, with the default as either 1911 (non-SSL) or 4911 (SSL). If you know the remote host is using a different (non-default) platform daemon port, enter it here.
- Step 3 Click the **OK** button. If you chose **Open Station** (non-SSL), go to [Step 4](#).
If you chose **Open Station with SSL**, different results are possible:
- If the Niagara station's FoxService is not enabled for SSL, an error results with "Cannot connect. Ensure server is running on specified port". Retry from [Step 1](#), changing the **Type** to **Station Connection**.
 - If the Niagara station's FoxService is enabled for SSL, but you have not yet installed a certificate for it, a popup **Identity Verification** dialog with certificate details appears. In most cases, you click **Accept** to proceed to the next step.
 - An **Authentication** popup dialog appears. Go to the next step.
- Step 4 In the popup **Authentication** dialog, enter your station user credentials
- Note:** *Be sure to use your station username and password – not credentials for the platform.*
- **Username**
Enter your station user name.
 - **Password**
Enter the password for this user.
- If the "Remember these credentials option" is available, you can select it, so that Workbench "remembers" these user credentials. However, this can pose a future security risk; thus this option may be disabled.
- Step 5 Click **OK** after entering credentials.
- If the station user credentials were correct, the station is connected, and the Fox connection icon appears in the side bar pane. Where Fox connection icons are:
-  **Station** or
 -  **Station SSL**
- The view pane shows the **Station Summary** view, listing primary components and details on the host.
- If the station user credentials were not correct, you are prompted to enter them again.
- For related details, refer to the "[About Security](#)" section "[Users and security](#)" on page 6-4.

To close a station using Workbench



To close a station, do the following:

- Step 1 In the nav side bar (refer to "[Using the nav side bar \(tree\)](#)" on page 9-9, if necessary) right-click on the station icon and select the **Close** command from the **popup** menu.
The **Close** dialog box appears.
- Step 2 In the **Close** dialog box, click **OK** to close the station.
The station is closed and the station icon is removed from the nav side bar.

To connect to a station using Workbench

To connect to a station, do the following:

- Note:** *You cannot connect to a station unless it is "open". An open station may be visible in your Workbench nav side bar, but appear dimmed (be disconnected). If there is no station icon in the nav side bar, you need to open the station before connecting. See "[To open a station](#)" on page 9-5.*
- To connect to a disconnected station, do the following:

- Step 1 In the nav side bar (refer to [“Using the nav side bar \(tree\)”](#) on page 9-9) right-click the `station` icon and select the **Connect** command from the **popup** menu. Or, simply double-click the station icon.
The popup **Authentication** dialog box appears.
- Step 2 Enter your station user credentials
- Note:** *Be sure to use your station username and password – not credentials for the platform.*
- **Username**
Enter your station user name.
 - **Password**
Enter the password for this user.
- If the “Remember these credentials option” is available, you can select it, so that Workbench “remembers” these user credentials. However, this can pose a future security risk; thus this option may be disabled.
- Step 3 Click **OK** after entering credentials.
- If the station user credentials were correct, the station is connected, and the Fox connection icon appears in the side bar pane. Where Fox connection icons are:
-  Station *or*
 -  Station SSL
- The view pane shows the **Station Summary** view, listing primary components and details on the host.
- If the station user credentials were not correct, you are prompted to enter them again.
- For related details, refer to the [“About Security”](#) section [“Users and security”](#) on page 6-4.

To disconnect from a station

To disconnect from a station, do the following:

- Step 1 In the nav side bar (refer to [“Using the nav side bar \(tree\)”](#) on page 9-9) right-click the `station` icon and select the **Disconnect** command from the **popup** menu.
The popup **Disconnect** confirmation dialog appears.
- Step 2 In the **Disconnect** dialog box, click **OK** to disconnect.
The station is disconnected and the `station` icon is dimmed.

Using the side bar pane

The side bar pane displays all open side bars. The pane may be shown or hidden by selecting a command from the Windows menu, as described in [“To open the side bar pane”](#) on page 9-6.

Following, are some facts about the behavior of side bars and the side bar pane:

- If you open a side bar, the side bar pane (if currently closed) will automatically open to display that pane.
- When you open Workbench, the side bar pane returns to its last previous state (displayed or hidden).
- When you show the side bar after all side bar panes have been closed (causing the side bar pane to hide) the side bar pane will be empty
- When you close all side bars, the side bar pane will close
- When you open Workbench, it will display the side bars that were open (if any) when the last session was closed.
- When more than one side bar is open in the side bar pane, side bars may be resized vertically, maximized, or restored to their previous size using the title bar controls.
- Side bars always expand to fill the side bar pane horizontally.

To open the side bar pane

To open the side bar pane, do the following:

- From the menu bar, select **Window > Side Bars > Show Side Bar** command.
The side bar will appear.

To open the bookmarks side bar

To open the bookmarks side bar, do the following:

- From the menu bar, select **Window > Side Bars > Bookmarks**.
The bookmarks side bar appears in the side bar pane.

To open the help side bar

To open the help side bar, do the following:

- From the menu bar, select **Window > Side Bars > Help**.
The help side bar appears in the side bar pane.

To open the nav side bar

To open the nav side bar, do the following:

- From the menu bar, select **Window > Side Bars > Nav**.
The nav side bar appears in the side bar pane.

To open the palette side bar

To open the palette side bar, do the following:

- From the menu bar, select **Window > Side Bars > Palette**.
The palette side bar appears in the side bar pane.

To close a side bar

To close the side bar pane, do the following:

- From the side bar title bar **dropdown** menu (left side), select the **Close** command.
The side bar pane closes.

To close the side bar pane

The side bar pane toggles on or off by selecting or deselecting the **Show Side Bar** command, as described below. The side bar pane also closes when all palettes are closed.

To close the side bar pane, do the following:

- From the menu bar, select **Window > Side Bars > Show Side Bar**.
The side bar pane closes.

To vertically resize a side bar

To vertically resize a side bar, do the following:

- Click on the title bar and drag vertically.
The side bar will adjust to any size that fits in the side bar pane.

To restore a side bar to a previous size

To restore a side bar to its previous size, do the following:

- Click on the **restore** icon on the title bar.
The side bar will change to its previous state.

To maximize a side bar:

To maximize a side bar, do the following:

- Click on the **restore** icon on the title bar.
The side bar will expand to its maximum size in the side bar pane.

Using bookmarks

Bookmarks are simply linked “shortcuts” or “favorites” to help you quickly find views in just a couple of clicks. The bookmark side bar provides a convenient place in the side bar pane to add, manage, edit, and remove bookmarks. You can also do these “bookmarking” tasks using the **Bookmarks** menu on the menu bar. For an overview of the bookmark side bar, see “[About the bookmark side bar](#)” on page 2-4.

This section describes the following tasks that are associated with the bookmark feature:

- [To add a bookmark](#)
- [To manage bookmarks](#)
- [To edit a bookmark](#)
- [To remove a bookmark](#)

To add a bookmark

To add a bookmark, do the following steps:

- Step 1 Make sure that the view you want to bookmark is showing in the view pane, and then do one of the following:
- From the menu bar, select **Bookmarks: Add to Bookmarks**.
 - From the **popup** menu in the palette side bar, select **Add to Bookmarks**.
- Step 2 In the **Add to Bookmarks** dialog box, type a name for the bookmark, in the **Name** field.
- Step 3 *Optional:* In the **Add to Bookmarks** dialog box, click the **New Folder** button to add a new folder, if desired.
- The **New Folder** dialog box appears. There you can name your new folder and click **OK** to save it.
- Step 4 In the **Add to Bookmarks** dialog box, select the folder where you want to locate the bookmark and click **OK**.
- The bookmark is added.

To manage bookmarks

To manage your bookmarks, do the following steps:

- Step 1 Do one of the following:
- From the menu bar, select **Bookmarks: Manage Bookmarks**.
 - From the **popup** menu in the palette side bar, select **Manage Bookmarks**.
- Step 2 In the **Manage Bookmarks** dialog box, click on one of the following buttons:
- **New Folder** buttons
To create new folders to organize your bookmarks
 - **New Bookmark** button
To create new bookmarks
 - **Copy to** button
To copy bookmarks to another location
 - **Move to** button
To move bookmarks to another location
 - **Remove** button
To delete the selected bookmark
 - **Edit** button
To rename a selected bookmark
 - **Move Up** or **Move Down** button
To move bookmarks up or down in the bookmark tree view

To edit a bookmark

To edit a bookmark, do the following steps:

- Step 1 In the bookmarks side bar, right-click the bookmark that you want to change and select **Edit** from the **popup** menu.
- The bookmark **Edit** dialog box will appear.
- Step 2 In the **Edit** dialog box, type the desired name for your bookmark in the **Name** field.
- Step 3 In the **Edit** dialog box, type the ORD path or scroll **Name** field.

To remove a bookmark

To remove a bookmark, do the following:

- In the bookmarks side bar, right-click the bookmark that you want to change and select **Remove** from the **popup** menu.
- The bookmark is deleted from the bookmark list.

Using the help side bar

When you open the help side bar for the first time, the help side bar is empty except for the **Load Help** button. The **Load Help** button will reappear any time any module's timestamp is changed or when a new module is added or removed. When you click the **Load Help** button, Workbench searches through all the available modules to create the help directory. The Help Sidebar provides a view of the available Help.

To search for topics in the help side bar:

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

To search for topics in the help side bar, do the following steps:

- Step 1 In help side bar, click on the Search tab. The search display appears.
- Step 2 In the **Find** field, type the word or phrase that you are looking for and click the **Search** button. Topics that match your search word(s) will appear in side bar below the text field.
- Step 3 Double-click on a topic to display the topic in the view pane.

Using the nav side bar (tree)

The nav side bar provides a hierarchical view of the system in a nav—tree presentation. Components that appear in the tree may be acted on in various ways depending the component type:

The following sections describe how to use [Nav side bar actions](#) and the following procedures:

- [To refresh a nav side bar tree node](#)
- [To refresh a nav side bar tree node](#)

Nav side bar actions

- **Expand**
Click the plus sign next to an icon to expand a branch of the tree and show any child nodes.
- **Collapse**
Click the minus sign next to an icon to collapse a branch of the tree.
- **Select**
Click a node in the tree to select it.
- **View**
Double—click the minus sign next to an icon to collapse a branch of the tree and hide its child nodes.
- **Use popup menu**
Right—click the desired node to use the context—sensitive **popup** menu. Refer to [“About the nav side bar popup menu items”](#) on page A-9 for details about the nav side bar **popup** menu.
- **Use file menu**
With a node selected, choose a menu item from the **File** menu. Refer to [“About the File menu”](#) on page A-2 for details about the **file** menu.

To refresh a nav side bar tree node

Refreshing a nav side bar tree node, updates the view to reflect any changes to the files of that node.

To refresh a nav side bar tree node, do the following steps:

- Step 1 Click the node to select it.
- Step 2 Do one of the following:
 - From the **File** menu, select **Refresh**.
The tree node refreshes.
 - From the **popup** menu, select **Refresh Tree Node**.
The tree node refreshes.

To Go Into a nav side bar tree node

The **Go Into** command allows you to re-root the nav tree at any arbitrary node. This will make that node the new root of the tree and give you a new tree in the nav side bar window. You can get back to the original tree by selecting it from the 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.

To refresh a nav side bar tree node, do the following steps:

- Step 1 Click the node to select it.
- Step 2 Do one of the following:
 - From the **File** menu, select **Refresh**.
The tree node refreshes.
 - From the **popup** menu, select **Refresh Tree Node**.

The tree node refreshes.

Using the jobs side bar

The Jobs side bar provides a place to view and manage current jobs. Once a job has completed you are notified via the async notification feature. You can also view job details using the ">>" button.

This section explains how to do the following:

- [To view job details](#)
- [To cancel a job from the jobs side bar](#)
- [To dismiss a job](#)

To view job details

To view job details from the jobs side bar, do the following:

- On the desired job, click the **details** (">>") button.
The job log details dialog box appears.

To cancel a job from the jobs side bar

To cancel a job from the jobs side bar, do the following:

- On the jobs side bar, click the **cancel** button.
The job is cancelled.

To dismiss a job

Once a job has been completed, you are notified via the async notification feature. You may then dismiss the job from the jobs side bar, if desired.

To dismiss a job, do the following:

- From the Jobs side bar click the **close** button on the job that you want to dismiss.
The selected job is closed.

Using the palette side bar

The palette side bar provides a place to open, view, and access components in modules. For an overview of the palette side bar, see [“About the palette side bar”](#) on page 2-8.

This section explains how to do the following:

- [To open a palette](#)
- [To close a palette](#)
- [To create a palette](#)
- [To add a new component to a palette \(PaletteFile.palette\)](#)
- [To add a new component to a palette \(module.palette\)](#)
- [To copy a component to a palette](#)

To open a palette

You can open one or more palettes in the palette side bar. Refer to [“To open the palette side bar”](#) on page 9-7 for details on how to open the palette side bar.

To open a new palette in the palette side bar, follow these steps:

- Step 1 On the palette toolbar click the **open palette** button.
The **Open Palette** dialog box appears.
- Step 2 In the **Open Palette** dialog box select one or more palettes from the list, as desired. The text field at the top of the dialog box allows you to filter the displayed palettes by typing in the name of the desired palette. Also, you can use the **Browse** button to locate palettes in alternate locations.

***Note:** Hold down the Ctrl or Shift key to make multiple selections.*
- Step 3 Click **OK**.
The palette(s) opens in the palette side bar pane.

To close a palette

You can close any of the palettes that are open in the palette side bar. When you close all the palettes, the palette side bar closes.

To close a palette, follow these steps:

- Step 1 From the **Drop-down palette** selector, choose the palette that you want to close.
The selected palette displays in the side bar pane.
- Step 2 Click the **Close palette** button.
The palette closes.

To create a palette

Palettes allow you to save copies of your work for future use. Palette files may reside under a station file system or as part of a module. This procedure describes how to create a palette that resides in a station file system. Refer to [“Working with modules”](#) on page 9-24 for information about creating a palette file that is part of a module (jar file).

Note: *For convenience and good organization, it is a good idea to create a “palette” folder on the file system to hold all your custom palette files.*

To create a palette, follow these steps:

- Step 1 In the nav tree, right-click on (or under) the station file system node, in the desired location (see note, above).
The **popup** menu appears.
- Step 2 From the popup menu, select **New > PaletteFile.palette**.
The **File Name** dialog box appears with a default name.
- Step 3 In the File Name dialog box, type the desired name for the new Palette file and click **OK**.
The **File Name** dialog box disappears and the new Palette file appears in the nav tree.

For more information about palette files, refer to [“About palettes”](#) on page 1-12..

To add a new component to a palette (PaletteFile.palette)

You can add a new component to any of your own custom palettes, store the palette files on your file system and view them in your palette side bar. The palette files must reside on the file system, not in the station database. However, the components that you add to your palette file may reside in the station database.

Note: *For convenience and good organization, it is a good idea to create a “palette” folder on the file system to hold all your custom palette files.*

To add a new component to a palette using the wire sheet view, follow these steps:

Note: *To add a new component using a view other than the wire sheet view, refer to the information in [“Working with components”](#) on page 9-15.*

- Step 1 In the nav tree, double-click on the palette file that you want to open.
The default (Wire Sheet) view appears.
- Step 2 Add a component to the wire sheet view using one of the following methods.
 - Right click on the wire sheet view and select a new component from the **New** menu.
 - Drag and drop a component from the nav side bar directly onto the wire sheet.
- Step 3 Select **File > SaveBog** from the main menu.
The palette file is updated to contain the new component.

Note: *If the updated palette is currently being viewed in the palette side bar, you must close and reopen the palette after saving the station (SaveBog).*

To add a new component to a palette (module.palette)

You can add new components to any of your palettes that have `module.palette` files associated with them. A single `module.palette` file may be associated with each module that is stored in the modules directory. You change a `module.palette` file by first making the desired changes to a `module.palette` file that resides in a module “build” directory under your file system and then using the build process to recreate the module that includes the changed palette file.

To add a new component to a palette (module.palette file), follow these steps:

- Step 1 Double click on the desired module.palette file. It should be located in a module directory on your station file system.
The default (Wire Sheet) view appears.
- Step 2 Add a component to the wire sheet view using one of the following methods:
- Right click on the wire sheet view and select a new component from the **New** menu.
 - Drag and drop a component or container from the nav side bar directly onto the wire sheet.
- Step 3 Select File > SaveBog from the main menu.
- Step 4 From the console command line (press F3 to open console window) navigate to the working directory and type the **build** command followed by a space and then the name of the module folder. For example:
`C:\Niagara\Niagara-3.x.xx\workingDirectory> build moduleName`
 Press the Return key to initiate the **build** process. If the build is successful, the updated jar file is built and overwrites the old jar file located in the modules folder. If the build is not successful, no jar file is built and error messages are generated to help you troubleshoot the problem.
- Step 5 Refer to “[To create a new \(.jar\) module](#)” on page 9-24 for more detailed instructions about creating a module.

To copy a component to a palette

You can copy components to any of your own custom palettes, store the palette files on your file system and view them in your palette side bar. The nav files must reside on the file system, not in the station database. However, the components that you add to your palette file may reside in the station database. The components that you add to your palette are copies and not connected to the original components that you are copying.

Note: *For convenience and good organization, it is a good idea to create a “palette” folder on the file system to hold all your custom palette files.*

To copy a component to a palette, using the Wire Sheet view, follow these steps:

- Step 1 In the nav tree, double-click on the file palette file that you want to open.
The default (Wire Sheet) view appears.
- Step 2 From the nav side bar, palette side bar, or other source, drag and drop the desire component to the wire sheet.
The copied component is added to the palette file.
- Step 3 Select File > SaveBog from the main menu.
The palette file is updated to contain the new component.

Note: *If the updated palette is currently being viewed in the palette side bar, you must close and reopen the palette in the palette side bar in order to see the new items.*

To add a side bar preview using the compPreviewWidget property

Palette previews display in the palette when components have images configured either as the default image property or as the image assigned to the compPreviewWidget property. If no preview is associated with a component, in AX-3.5 and later, you can add a compPreviewWidget property to a widget in order to display an image in the preview pane of the palette side bar.

To add a palette preview image to a component, do the following:

- Step 1 Right-click the desired component in the palette side bar and select **Views > Slot Sheet** from the popup menu. The Slot Sheet view displays.
- Step 2 In the slot sheet view, right-click in an empty area of the view and select **Add Slot** from the popup menu. The **Add Slot** dialog box displays.
- Step 3 In the **Add Slot** dialog box, enter the following information:
- **Name**
Type compPreviewWidget in this text field.
 - **Type**
Select bajaui and Label from the two option lists respectively.
- Step 4 Click the **OK** button. The compPreviewWidget slot is added to the component.
- Step 5 Right-click the desired component in the palette side bar and select **Views > Property Sheet** from the popup menu. The Property Sheet view displays.
- Step 6 In the property sheet view, edit properties under the compPreviewWidget to add text and an image, as desired and click the **Save** button.

- Step 7 Click the **OK** button. The preview is configured and should appear in the preview pane the next time you select the component in the palette side bar.
- Step 8 Right-click the desired component in the palette side bar and select **Views > Slot Sheet** from the popup menu. The Slot Sheet view displays.
- Step 9 In the Slot Sheet view, right-click on the compPreviewWidget slot and select **Config Flags** from the popup menu. The **Config Flags** dialog box displays.
- Step 10 In the **Config Flags** dialog box, select the **Hidden** and **Remove on Clone** options and click the **OK** button. The preview is now configured and should display in the preview pane the next time you select the component in the palette side bar.

Using the view pane

The view pane can display any component view that you choose (see [“About views”](#) on page 1-19 for an introduction to views). Different views of components can provide different editing options. Most component editing is done in the view pane, however, many editing functions also involve the nav tree pane as well. The following sections describe some of the basic tools provided in Workbench for editing components. These tools are context-sensitive and may be used in several – but not all – views and with several – but not all – types of components.

For more details about viewing components, refer to [“Viewing a component”](#) on page 9-14

Editing components

Editing components in Workbench typically involves:

- [Selecting a component](#)
- [Viewing a component](#)
- [Working with components](#)
- [Editing component data](#)

The following sections describe these tasks.

Selecting a component

By selecting an item, you are defining the scope of your actions. In other words, the only components that are directly affected by your edits are those that are “selected” during your edits. Of course, other components may be influenced by the change in a component that communicates with it. It is also important to remember the hierarchical nature of the component tree. If you delete or move a component that contains other components, then you are deleting or moving all items that are contained in that container component.

When items are selected, they appear in one of the following ways:

- In the nav tree view, the component text is changed to a white on purple highlight.
- In the wire sheet view, selected components have a red border with red handles.
- In the wire sheet view, selected links turn red.

For more information about views, refer to [“About views”](#) on page 1-19.

To select a component

To select a component, do one of the following:

- In the nav tree view or the wire sheet view, left-click the desired component.
The component is selected.
- In the nav tree view or the wire sheet view, right-click the desired component.
The component (or link) is selected.
- In the wire sheet view, drag around the desired component.
The component (or link) is selected.

To select multiple components in the nav tree

To select more than one component in the nav tree do the following:

- Step 1 Select a single component or link, as described in [“To select a component”](#).
- Step 2 Hold down the Shift key or Ctrl key and select additional items.
Multiple components (or links) are selected.

- Step 3 *Optional:* In the wire sheet view, drag around the group of elements that you want to select. Hold down the Shift key or Ctrl key and select additional items to select or deselect them.
Multiple components (or links) are selected.

To select a multiple components in the wire sheet

To select more than one component in the wire sheet do the following:

- Step 1 Select a single component or link, as described in [“To select a component”](#).
Step 2 Hold down the Shift key or Ctrl key and select additional items.
Multiple components (or links) are selected.

Viewing a component

Each component that you select (in a view or in the nav tree) has a “default” view that appears in the view pane. You can select a different view of the component using the view selector or the **popup** (right-click) menu. The last view of an object becomes its default view during a Workbench session. If Workbench is closed and restarted, default views are reset to the original default views.

For details on selecting component views, see [“To select a component view”](#) on page 9-14.

In addition to having different views, you can edit the way views display information. For details about customizing a view, refer to the following procedures:

[“To reorder components \(in a component container\)”](#) on page 9-14

[“To create a composite view”](#) on page 9-15

To select a component view

To select a component view, do one of the following:

- Double-click on a component in the nav side bar.
This will display the default view of the component.
- Right-click a component and choose a view from the **Views** submenu.
- With a component selected, choose a view from the **view selector** menu

To reorder components (in a component container)

You can reorder the components that are contained in a “container” component using the property sheet view, the nav tree view, or the wire sheet view.

To reorder components, do the following:

- Step 1 Right-click the desired component.
The **popup** menu appears.
- Step 2 From the **popup** menu, select **Reorder**.
The **Reorder** dialog box appears.
- Step 3 In the **Reorder** dialog box, use the **sort** buttons, as follows:
- **Move Up** button
Select one or more items in the component list (use the Ctrl or Shift key to select multiples) and click this button to move selected item(s) up (one position per click) in the list.
 - **Move Down** button
Select one or more items in the component list (use the Ctrl or Shift key to select multiples) and click this button to move selected item(s) down (one position per click) in the list.
 - **Sort by Name** button
Click this button to sort all items in the list in alphabetical order, by name. Each click toggles the sort order between ascending and descending sort order.
 - **Sort by Type** button
Click this button to sort all items in the list in order by component type. Each click toggles the sort order between ascending and descending sort order.
 - **Reset** button
Click this button to set the components in the list back to the order that they were in when the Reorder dialog box was opened.

The components are sorted according to button selected.

To create a composite view

Composite views are available in the wire sheet view and in the property sheet view. Creating a composite view of a component allows you to expose one or more child slots (that you designate) at the parent–view level. For more information about composite views, refer to “[About composites](#)” on page 3-23.

To create a composite view, do the following:

Note: *The component that you select in the following step must have a child component to be a legitimate parent component. If no valid child component is available, then there cannot be a composite.*

- Step 1 In the wire sheet view or in the property sheet view, right-click on the desired parent object. appears.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Composite** command.
The **Composite Editor** appears.
- Step 3 In the **Composite Editor**, expand the tree to find and select the desired points to add to the parent component. Select multiple points (if desired) by holding the Ctrl or Shift keys when selecting.
- Step 4 With the desired point(s) selected, click the **Add** button to add one or more points to the right column of the **Composite Editor**.
- Step 5 *Optional:* Use the Composite Editor control buttons to do the following, if desired:
- **Reverse**
Reverse the chosen side (“in” or “out”) of a point that is being composited.
 - **Rename**
Provide a custom name for the new slot created by composited point.
 - **Remove**
Removes the composited item (slot) from right side of the Composite Editor.
 - **Move Up**
Sort all selected points by moving them up in the Composite Editor (list (one position per click)).
 - **Move Down**
Sort all selected points by moving them down in the Composite Editor list (one position per click).
- Step 6 Click the **OK** button.
The component composites are set.

Working with components

The following procedures describe basic tasks that are used when working with components:

- [To add a new component \(property sheet, nav side bar, or wire sheet view\)](#)
- [To delete a component \(property sheet, nav side bar, or wire sheet view\)](#)
- [To rename a component \(property sheet, nav side bar, or wire sheet view, slot sheet\)](#)
- [To duplicate a component \(property sheet, nav side bar, or wire sheet view\)](#)
- [To paste a component \(property sheet, nav side bar, or wire sheet view\)](#)
- [To drag a component to the property sheet, nav side bar, or wire sheet \(left-click method\)](#)
- [To drag a component to the property sheet, nav side bar, or wire sheet \(right-click method\)](#)

To add a new component (property sheet, nav side bar, or wire sheet view)

You can add (create) a new component to a container using the property sheet view, the nav tree view or the wire sheet view. To add a slot refer to “[To add a new slot](#)” on page 9-19.

To add a new component, do the following:

- Step 1 Right-click in the container where you want to place the new component. This can be a component in the nav tree view, the wire sheet view or the property sheet view.
A **popup** menu appears.
- Step 2 From the **popup** menu, choose the **New** command and select the desired type of component:
- **Folder**
 - **IconFolder**
 - **TextBlock**
 - **BooleanWritable**
 - **NumericWritable**
 - **EnumWritable**
 - **StringWritable**
- The **Name** dialog box appears.

- Step 3 Type the desired name for the new component in the text field and click the **OK** button.
The new component appears in the property sheet.

To delete a component (property sheet, nav side bar, or wire sheet view)

You can delete a component from a container, using the property sheet view, the nav tree view or the wire sheet view. To delete a component using the slot sheet view, refer to To delete a slot refer to [“To delete a slot”](#) on page 9-20.

To delete a component, do the following:

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

- Step 1 Right-click the desired component.
A **popup** menu appears.
- Step 2 From the **popup** menu, select the **Delete** command.
The component is deleted.

To rename a component (property sheet, nav side bar, or wire sheet view, slot sheet)

You can rename a component in the property sheet view, the nav tree view, the wire sheet view, or the slot sheet view.

To rename a component, do of the following:

- Step 1 In the desired view, right-click the component to rename.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Rename** command.
The **Rename** dialog box appears.
- Step 3 In the **Rename** dialog box, type the desired name in the text field and click the **OK** button.
The component is renamed.

To duplicate a component (property sheet, nav side bar, or wire sheet view)

You can duplicate a component from the property sheet view, the nav tree view, or the wire sheet view.

To duplicate a component, do of the following:

- Step 1 In the desired view, right-click the component to duplicate.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Duplicate** command.
The **Name** dialog box appears.
- Step 3 In the **Name** dialog box, type the desired name in the text field (or use the default name) and click the **OK** button.
The component is duplicated.

To paste a component (property sheet, nav side bar, or wire sheet view)

You can add a component to a container by pasting it to a container displayed the property sheet view, the nav tree view, or the wire sheet view.

To add a component by *pasting* it, do the following:

- Step 1 In the desired view, select the desired component and use the **Copy** (to duplicate) or **Cut** (to remove) command to copy the component to the clipboard.
Copy and **Cut** commands are available on the **popup** (right-click) menu, the **Edit** menu, or from the toolbar.
- Step 2 Right-click in the container where you want to place the new component. This can be a component in the nav tree view, the wire sheet view or the property sheet view.
The **popup** menu appears.
- Step 3 From the **popup** menu, select the paste command.
The **Name** dialog box appears.
- Step 4 Type the desired name for the new component in the text field and click the **OK** button.
The new component is added to the targeted container.

To drag a component to the property sheet, nav side bar, or wire sheet (left-click method)

You can drag a component to the displayed property sheet view, the nav tree view, or the wire sheet view, by using a right-click drag or by using a normal, left-click drag.

Note: A left-click drag always creates a copy of the original.

To drag a component using the left-click drag method, do the following:

- Step 1 In the desired view, left-click the component to copy and drag it to the target container. When you release the mouse button, the **Name** dialog box appears.
- Step 2 Type the desired name for the new component in the text field and click the **OK** button. The new component is added to the targeted container.

To drag a component to the property sheet, nav side bar, or wire sheet (right-click method)

You can drag a component to the displayed property sheet by using a right-click drag or by using a normal, left-click drag.

Note: A right-click drag always prompts you to choose a **copy**, **move**, or **cancel** command.

To drag a component to a property sheet by using the right-click drag method, do the following:

- Step 1 In the desired view, right-click the component that you want to move (or copy) and drag it to the property sheet. When you release the right mouse button, a **popup** menu appears.
- Step 2 From the **popup** menu, choose one of the following commands:
- **Copy** – this command makes a duplicate of the component to paste onto the property sheet. The **Name** dialog box appears.
 - **Move** – this command removes the original component and prepares to paste in onto the property sheet. The **Name** dialog box appears.
 - **Cancel** – this command will stop the process, and the **popup** menu will disappear.
- Step 3 Type the desired name for the new component in the text field and click the **OK** button. The new component appears in the property sheet.

Editing component data

Many component editing functions in Workbench work the same, or similarly, in all component views. However, some editing functions are restricted to particular component views. For information about the standard edit commands (such as cut, copy, and paste, refer to “[Types of edit commands](#)” on page A-13. The following sections describe some of the basic component editing procedures.

- [Using the property sheet](#)
- [Using the wire sheet](#)
- [Using the slot sheet](#)
- [Using the link sheet](#)

Using the property sheet

The property sheet view is displayed in the view pane by selecting **Property Sheet** from the **view selector** menu (see “[About the view selector](#)” on page 2-15) or by using one of the other methods described in “[About the view pane](#)” on page 2-12.

For information about creating new components, deleting components, duplicating components, and so on, refer to “[Editing components](#)” on page 9-13.

To add an extension to a component property sheet view:

For more information about extensions, refer to “[Extension process overview](#)” on page 3-12 or “[About point extensions](#)” on page 3-12

Note: The following procedure may also be performed directly in the nav side bar tree (refer to “[Using the nav side bar \(tree\)](#)” on page 9-9) for more information.

- Step 1 Display the component property sheet view, as follows:
- In the nav pane tree, select the desired component.
 - From the right-click **popup** menu (or from the view selector), select the component property sheet view.

- Step 2 Open the desired extensions palette from the palette side bar.
- Step 3 Expand the extensions folder to find the extension that matches your data type and collection method.
- Step 4 Drag and drop or copy and paste the extension onto the bottom of the properties sheet.
The extension is added. Refer to [“Configure history extensions”](#) on page 4-25

To add a component facet:

For more information about facets, refer to [“About point facets”](#) on page 3-7.

- Step 1 In the property sheet view of a component, click the **Edit Facet** (>>) button.
The **Edit Facets** dialog box appears.
- Step 2 In the **Edit Facets** dialog box, click the **Add** button.
The **Add...** dialog box appears.
- Step 3 From the **Key:** options menu, select a parameter. Duplicates are not allowed.
- Step 4 From the **Type:** options menu, select a parameter.
- Step 5 Click the **OK** button.
The **Add...** dialog box closes and the **Edit Facets** dialog box displays the new information.
- Step 6 From the **Edit Facets** dialog box, click the **OK** button.
The new facet is added.
- Step 7 In the property sheet, click the **Save** button to your save changes.

To remove a component facet:

For more information about facets, refer to [“About point facets”](#) on page 3-7.

- Step 1 In the property sheet view of a component, click the **Edit Facet** (>>) button.
The **Edit Facets** dialog box appears.
- Step 2 In the **Edit Facets** dialog box, click the **Remove** button.
The **Remove...** dialog box appears.
- Step 3 From the **Remove...** dialog box, choose a parameter(s) that you want to delete by selecting the associated check box(es).
- Step 4 Click the **OK** button.
The **Edit Facets** dialog box disappears and deleted parameter information does not appear.
- Step 5 From the **Edit Facets** dialog box, click the **OK**.
The new facet is removed.
- Step 6 In the property sheet, click the **Save** button to save changes.

Using the wire sheet

The following are common tasks associated with the wire sheet

- [To pin a slot](#)
- [To paste a component onto the wire sheet](#)
- [To drag a component to the wire sheet \(left-click method\)](#)
- [To drag a component to the wire sheet \(left-click method\)](#)
- [To drag a component to the wire sheet \(right-click method\)](#)

To pin a slot

By default, some slots are visible in the wire sheet view of the slot. However, you can have any – or all slots visible on the wire sheet by “pinning” open the slot(s).

To pin a slot:

- Step 1 In the wire sheet view, right-click on the desired component.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Pin Slots** command.
The **Pin Slots** dialog box appears.
- Step 3 In the **Pin Slots** dialog box, click to the left of all slots that you want to pin open.
A pin icon appears next to each icon you click.
- Step 4 Click the **OK** button.

The slots are pinned and appear on the wire sheet with a knob next to the slot name.

To paste a component onto the wire sheet

You can add a component to the displayed property sheet by pasting it onto the property sheet or by dragging it onto the property sheet.

To add a component to a property sheet by *pasting* it to the property sheet, do the following:

- Step 1 Select the desired component and use the **Copy** (to duplicate) or **Cut** (to remove) command to copy the component to the clipboard.
Copy and **Cut** commands are available on the **popup** (right-click) menu, the Edit menu, or from the toolbar.
- Step 2 Right-click on the property sheet.
The **popup** menu appears.
- Step 3 From the **popup** menu, select the paste command.
The **Name** dialog box appears.
- Step 4 Type the desired name for the new component in the text field and click the **OK** button.
The new component appears in the property sheet.

To drag a component to the wire sheet (left-click method)

You can drag a component to the displayed property sheet by using a right-click drag or by using a normal, left-click drag.

To drag a component to a property sheet by using the *left-click* drag method, do the following:

- Step 1 Select the desired component using a left-click and drag it to the property sheet.
When you release the mouse button, the **Name** dialog box appears.
- Step 2 Type the desired name for the new component in the text field and click the **OK** button.
The new component appears in the property sheet.

To drag a component to the wire sheet (right-click method)

You can drag a component to the displayed property sheet by using a right-click drag or by using a normal, left-click drag.

To drag a component to a property sheet by using the *right-click* drag method, do the following:

- Step 1 Right-click the desired component and drag it to the property sheet.
When you release the right mouse button, a **popup** menu appears.
- Step 2 From the **popup** menu, choose one of the following commands:
- **Copy** – this command makes a duplicate of the component to paste onto the property sheet.
The **Name** dialog box appears.
 - **Move** – this command removes the original component and prepares to paste in onto the property sheet.
The **Name** dialog box appears.
 - **Cancel** – this command will stop the process, and the **popup** menu will disappear.
- Step 3 Type the desired name for the new component in the text field and click the **OK** button.
The new component appears in the property sheet.

Using the slot sheet

The following are common tasks associated with the slot sheet

- [To add a new slot](#)
- [To delete a slot](#)

To add a new slot

To add a new slot, perform the following steps:

- Step 1 In the slot sheet view, right-click in the pane.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Add Slot** command.
The **Add Slot** dialog box appears.

- Step 3 In the **Add Slot** dialog box, enter the following information:
- Name: type in a name for the slot.
 - Type: Select the component property type from the drop-down option lists.
 - Choose the desired slot options from the Flags field.
- Step 4 Click the **OK** button.
The slot is added to the slot sheet.

To delete a slot

To delete a slot, perform the following steps:

- Step 1 In the slot sheet view, right-click on the slot that you want to delete.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Delete** command.
The slot is deleted.

Using the link sheet

The following are common tasks associated with the link sheet

- [To delete a link using the link sheet view](#)
- [To edit a link using the link sheet view](#)

To delete a link using the link sheet view

To delete a link using the link sheet view, perform the following steps:

- Step 1 In the link sheet view, right-click on the link that you want to delete.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Delete** command.
The link is deleted.

To edit a link using the link sheet view

To edit a link using the link sheet view, perform the following steps:

- Step 1 In the link sheet view, right-click on the link that you want to edit.
The **popup** menu appears.
- Step 2 From the **popup** menu, select the **Edit** command.
The **edit** dialog box appears.
- Step 3 In the **edit** dialog box, edit the following fields, as desired
- Source Ord
Type in (or browse to) the ORD for the desire source of the link.
 - Source Slot Name
Type in a name for the Source.
 - Target Slot Name
Type in a name for the Target Slot
 - Enabled
Choose **True** or **False** from the Enable option list. **True** enables the link and **False** disables the link.
- Step 4 In the **edit** dialog box, click the **OK** button.
The **edit** dialog box disappears and the edits are saved.

Using alarms

The following are common tasks associated with setting up and working with alarms in NiagaraAX:

- [Creating alarms](#)
- [Routing alarms](#)
- [Managing alarms](#)

Creating alarms

To create an “alarmable” control point, you must add an appropriate alarm extension to a control point and configure it to define your alarm condition requirements:

To set up alarm conditions for a control point

Note: Just as there are different types of control points (refer to “Types of control points” on page 3-1), there are different types of control point extensions (refer to “Types of point extensions” on page 3-12). When adding an alarm extension to a control point, you must add the proper type of extension to match the point, as described in “Types of alarm extensions” on page 3-15.

To set up alarm conditions for a control point, do the following:

- Step 1 Add an alarm extension to the control point property sheet as described in “To add an extension to a component property sheet view:” on page 9-17. The alarm extension must match the control point type (for example, BooleanChangeOfStateAlarmExt fits with a Boolean Writable point – and a EnumChangeOfStateAlarmExt would not).
- Step 2 Configure the alarm extension properties as desired. Refer to the “About alarm extension properties” on page 5-3 for details about the alarm extensions properties.

Note: The *Alarm Enable* property options are not selected by default. You must select an *AlarmEnable* option in addition to setting your alarm extension properties.

Routing alarms

The alarm service handles the routing of alarms by using alarm classes to coordinate the transfer of messages between alarm sources and alarm recipients. Refer to “Types of alarm recipients” on page 5-36 for more information about alarm recipients.

To add an alarm class to the alarm service

For more information about alarm service, refer to “About the alarm service” on page 5-7. For more information about alarm class, refer to “About alarm class” on page 5-32.

To add an alarm class to the alarm service, do the following:

- Step 1 Right-click the AlarmService node in the nav side bar.
The **popup** menu appears.
- Step 2 From the **popup** menu, select Views > WireSheet.
The alarm service Wire Sheet view appears.
- Step 3 From the alarm palette, drag an AlarmClass component onto the wiresheet.
The **Name** dialog box appears.
- Step 4 Type the desired name for the new alarm class in the text field and click the **OK** button.
The new alarm class appears in the wiresheet and is ready for configuration.

To add a console recipient to the alarm service

For more information about alarm service, refer to “About the alarm service” on page 5-7. For more information about console recipient, refer to “About the console recipient” on page 5-36.

To add a console recipient to the alarm service, do the following:

- Step 1 Right-click the AlarmService node in the nav side bar.
The **popup** menu appears.
- Step 2 From the **popup** menu, select Views > WireSheet.
The alarm service Wiresheet view appears.
- Step 3 From the alarm palette, drag a ConsoleRecipient component onto the wiresheet.
The **Name** dialog box appears.
- Step 4 Type the desired name for the new console recipient in the text field and click the **OK** button.
The new console recipient appears in the wiresheet and is ready for configuration.

To add a station recipient to the alarm service

For more information about alarm service, refer to “About the alarm service” on page 5-7. For more information about station recipient, refer to “About the station recipient” on page 5-41.

To add a station recipient to the alarm service, do the following:

- Step 1 Right-click the AlarmService node in the nav side bar.
The **popup** menu appears.
- Step 2 From the **popup** menu, select Views > WireSheet.

- The alarm service Wiresheet view appears.
- Step 3 From the alarm palette, drag a StationRecipient component onto the wiresheet.
The **Name** dialog box appears.
- Step 4 Type the desired name for the new station recipient in the text field and click the **OK** button.
The new station recipient appears in the wiresheet and is ready for configuration.

To add an email recipient to the alarm service

For more information about alarm service, refer to “[About the alarm service](#)” on page 5-7. For more information about email recipient, refer to “[About the Email Recipient \(email-EmailRecipient\)](#)” on page 5-43.

To add an email recipient to the alarm service, do the following:

- Step 1 Right-click the AlarmService node in the nav side bar.
The **popup** menu appears.
- Step 2 From the **popup** menu, select Views > WireSheet.
The alarm service Wiresheet view appears.
- Step 3 From the email palette, drag an EmailRecipient component onto the wiresheet.
The **Name** dialog box appears.
- Step 4 Type the desired name for the new email recipient in the text field and click the **OK** button.
The new email recipient appears in the wiresheet and is ready for configuration.

Managing alarms

To acknowledge alarms from the alarm console view

Note: Alarms are not removed from the alarm console until both the following conditions exist:

- alarm acknowledged
- alarm source is in a normal (not alarm) state

You can acknowledge alarms from the alarm portal as well as from the alarm console. To acknowledge alarms from the alarm portal, refer to “[To acknowledge alarms from the alarm portal](#)” on page 9-22.

To acknowledge alarms from the alarm console view, do the following:

- Step 1 Right-click on the ConsoleRecipient node in nav side bar pane.
The **popup** menu displays.
- Step 2 From the **popup** menu, select Views > Alarm Console.
The alarm console view appears with all alarm sources displayed in a tabular view.
- Step 3 In the alarm console view, select alarm sources that you want to acknowledge. Select multiple alarms, if desired, using the Shift or Ctrl key.

Note: Each record that appears in the alarm console table represents one alarm source and one or more alarms from that source. You may acknowledge either the latest (most recent) alarm or acknowledge all alarms that are reported from that source by choosing either the “Acknowledge” command or the “Acknowledge All” command, as described in the following step.

- Step 4 Acknowledge selected alarm(s) by doing one of the following:
- From the **Alarm** menu, select **Acknowledge** (to acknowledge the latest alarm only)
The latest alarm from each selected choice is acknowledged.
OR
 - From the **Alarm** menu, select **Acknowledge All** (to acknowledge all alarms reported from that source). Alternatively, you may click the **Acknowledge All** button (located at the bottom of the dialog box).
All alarms from each selected alarm source are acknowledged.

To acknowledge alarms from the alarm portal

The alarm portal must be running and configured to include all desired alarm consoles before it can be used to acknowledge alarms. Refer to “[Alarm portal](#)” on page 8-3 for details about the alarm portal. Refer to “[Alarm Portal options](#)” on page 2-28 for details about alarm portal display options.

Note: Alarms are not removed from the alarm console (or portal) until both the following conditions exist:

- alarm acknowledged
- alarm source is in a normal (not alarm) state

You can acknowledge alarms from the alarm console as well as from the alarm portal. To acknowledge alarms directly from the alarm console, refer to [“To acknowledge alarms from the alarm console view”](#) on page 9-22.

To acknowledge alarms from the alarm portal, do the following:

- Step 1 From the **Tools** menu, select **Alarm Portal**.
The **Alarm Portal** view displays in the view pane.
- Step 2 If the desired Alarm Console does not appear in the Alarm Console Monitor table (top portion of the Alarm Portal view) right-click in the Alarm Console Monitor area.
The **popup** menu, appears.
- Step 3 Select Add Alarm Console.
The **Add Alarm Console** wizard appears.
- Step 4 Complete the **Add Alarm Console** wizard by entering the following information:
- Host address information, as needed (IP or dialup)
 - Credentials information (username and password)
 - Choose the desired console, by Ord (if more than one is present at the host address)
- Step 5 Click the **Finish** button to complete the Add Alarm Console wizard.
The alarm console appears in the **Alarm Console Monitor** table (top portion of the Alarm Portal view). Any alarms will appear in the **Open Alarm Sources** table (bottom portion of the Alarm Portal view).
- Step 6 In the **Open Alarm Sources** table, select alarm sources that you want to acknowledge. Select multiple alarms, if desired, using the Shift or Ctrl key.
- Note:** *Each record that appears in the **Open Alarm Sources** table represents one alarm source and one or more alarms from that source. You may acknowledge either the latest (most recent) alarm or acknowledge all alarms that are reported from that source by choosing either the “Acknowledge” command or the “Acknowledge All” command, as described in the following step.*
- Step 7 Acknowledge selected alarm(s) by doing one of the following:
- From the **Alarm** menu, select **Acknowledge** (to acknowledge the latest alarm only)
The latest alarm from each selected choice is acknowledged.
OR
 - From the **Alarm** menu, select **Acknowledge All** (to acknowledge all alarms reported from that source). Alternatively, you may click the **Acknowledge All** button (located at the bottom of the dialog box) or use the **popup** menu to acknowledge the alarm(s).
All alarms from each selected alarm source are acknowledged.

To delete alarm records

The Alarm Database Maintenance view provides three ways to delete alarm records from the alarm database. To delete old alarm records, do the following:

- Step 1 In the Alarm Database Maintenance view, select one of the following three options:
- **Clear Old Records option**
to clear alarm records *before* a certain date and time. The **Before** field is provided to allow you to set the date and time for removing old records. The Before field is not available when you select either of the other options.
 - **Clear All Before Selected Record option**
to delete all records that have a timestamp earlier than the timestamp of the record that you select in the Alarm History pane table. The selected record is not deleted.
 - **Clear All Records options**
to delete all records that appear in the Alarm History pane table, regardless of the date.
- Step 2 Click the **Run Maintenance** button to initiate the delete action.
The **Confirm Clear** dialog box displays to clarify that you are about to delete records and that the operation cannot be “undone”.
- Step 3 If the information in the **Confirm Clear** dialog box confirms that you are deleting the desired alarm records, click the **Yes** button (otherwise click the **No** button).
The alarm records are deleted and removed from the Alarm History table in the Alarm History pane.

Creating a Live History Chart

The Live History Chart view is available directly from the history extension component and also from a view on the HistoryPointList component. The history extension view displays the single point in a live chart plot, while the HistoryPointList component, when configured, displays all List Points that it contains.

The following procedures describe how to setup these views.

- [Live History Chart view from a History Extension](#)
- [Live History Chart view from a HistoryPointsList Component](#)

Live History Chart view from a History Extension

Note: Use this direct "history extension method" to display a single live charting point. If you need to view several points plotted simultaneously, then use the HistoryPointList component and the steps in the section, *Live_Chart_view_from_a_HistoryPointsList_Component*.

To display a Live Chart view in workbench using only a history extension, do the following:

- Step 1 Right-click the desired history extension component in the workbench nav tree. The popup menu displays.
- Step 2 From the popup menu, select Views>Live History Chart. The Live History Chart view displays.

Live History Chart view from a HistoryPointsList Component

To display a Live Chart view in workbench using the HistoryPointsList component, do the following:

- Step 1 From the History palette, add a HistoryPointList component to your station in an appropriate location (you can add the component anywhere in the station).
- Step 2 For each desired data plot, add a HistoryPointListItem to the HistoryPointList using one of the following methods:
 - Drag and drop the component from the History palette, then link the history extension and configure the component using the property sheet view.
 - Right click on the HistoryPointList container, choose Actions>AddItem from the popup menu. Using the List Item dialog box, *link* the history extension and configure the component.
- Step 3 Double-click on the HistoryPointList component to display the Live History Chart view. When the Live History Chart view displays, select or clear selections in the Item List View pane, as desired.

Working with modules

You can create and customize your own (.jar) modules so that you have access to individual custom components or complete folders of components and applications that you have designed and may use again.

Note: Starting in AX-3.7, non-developers can also create and customize "synthetic" (.sjar) modules. With synthetic modules and types, there is no source code, no compiling. So, no developer experience is needed. For details, refer to the engineering notes document NiagaraAX Synthetic Modules.

For basic information about modules, see ["About modules"](#) on page 1-6.

Creating a new (.jar) module

To create a new (.jar) module

You can create your own custom modules. This allows you to save copies of your work for future use in the compressed (jar) module form. Palette and lexicon files may be customized and included as part of the modules. Typically, if you choose to create a palette and a lexicon file, you select these options in the New Module Wizard, which creates an empty "module.lexicon" and "module.palette" file that you edit, as desired.

Note: For convenience, it is a good idea to create a "working" folder under the station directory (as described in step 1, below) to hold all your custom files source material. You can leave this folder in your station and use it to hold source files for any modules that you create.

To create a new module using Workbench, follow these steps:

- Step 1 Connect to the station that you want to work with and, in the nav side bar, expand the **My File System** > **Sys Home** nodes. The station installation directory appears under this node.

- Step 2 Right-click on the **Sys Home** node in the tree and select New > New Folder from the popup menu to create a “working folder”. Give the folder a generic name, such as “ModuleSource” or “workingModule”.
- Step 3 Use the **New Module Wizard** to create the directory and files required for a new module. Refer to [“New Module wizard”](#) on page 8-10 for more information about using the **New Module Wizard**. The wizard creates a new folder under a directory that you specify (typically, you want to specify the “working directory” that you create in step 1, above).
- Step 4 In the nav side bar, create a new folder named “src” under the newly created module folder. In the nav tree you should now have a folder structure similar to the following:
My File System/Sys Home/<workingDirectory>/<newModuleName>/src
- Step 5 Add items to the src folder, as desired. Copy and paste components or files from your file system or from the station database. You can add subfolders under the src folder to organize data, as desired, or you can put everything directly into the src folder.
- Step 6 In the <newModuleName> folder, double-click the “build.xml” file. It opens in Workbench text file editor.
- Step 7 In the “build.xml” file, after the last <dependency/> element, type in one or more <resources/> elements to define the location of your source files.
For example, if you want to include all the “gif” files that are in an “images” folder under your “src” folder, you type the following line in the “build.xml” file:

```
<resources name="/images/*.gif"/>
```


The following listing is an example of how your “build.xml” file might look.

```
<!-- Module Build File -->
<module
name = "newModuleName"
bajaVersion = "0"
preferredSymbol = "nM"
description = "My new graphic files"
vendor = "Acme"
>
<!-- Dependencies -->
<dependency name="baja" vendor="Tridium" vendorVersion="3.4" bajaVersion="0" />
<resources name="/images/*.*/"/>
</module>
```
- Step 8 Save and close the “build.xml” file.
- Step 9 From the console command line in Workbench (press F3 to open console window) navigate to the working directory and type the **build** command followed by a space and then the name of the module folder. For example:
C:\Niagara\Niagara-3.x.xx\workingDirectory> **build moduleName**
Press the Return key to initiate the **build** process. If the build is successful, the new jar file is built and located in the modules folder. If the build is not successful, no jar file is built and error messages are generated to help you troubleshoot the problem.
- Step 10 You must restart workbench to see the new module in the modules directory.
Note: *If you chose to create a palette during the build process (using the New Module Wizard) an empty palette is available with the new module. Refer to [“To add a new component to a palette \(module.palette\)”](#) on page 9-11 for instructions about adding components to the palette.*

CHAPTER 10

Component Guides

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 *KitControl* 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

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


- Refer to [“About alarm class”](#) on page 5-32 for more details.

alarm-AlarmClassFolder

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

- Refer to [“About alarm class”](#) on page 5-32 for related details.


alarm-AlarmConsoleOptions

 The AlarmConsoleOptions component allows you to configure the parameters of the alarm console. The **Options** dialog box displays when you select the Workbench **Tools > Options** menu item.

- Refer to [“Alarm Console options”](#) on page 2-27 for details about the individual alarm console options.
- Refer to [About the alarm console](#) for details about the alarm console.

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


alarm-AlarmPortalOptions

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

- Refer to [“Alarm Portal options”](#) on page 2-28 for details about the individual alarm console options.
- Refer to [About the alarm console](#) for details about the alarm console.


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

alarm-AlarmService

 The AlarmService uses AlarmClasses to route all alarm messages between AlarmSources and AlarmRecipients. Each station contains a single AlarmService. The AlarmService is available in the alarm palette.


- Refer to [“About the alarm service”](#) on page 5-7 for more details.

alarm-AlarmSourceExt

 AlarmSourceExt is the abstract superclass of all Baja control alarming algorithms. The AlarmSourceExt is available in the alarm module. Some common alarm source extensions include OutOfRangeAlarmExt, BooleanChangeOfStateAlarmExt, BooleanCommandFailureAlarmExt, EnumChangeOfStateAlarmExt and EnumCommandFailureAlarmExt.

Refer to [“About alarm extensions and components”](#) on page 5-3 for more details.


alarm-AlarmSourceInfo

 AlarmSourceInfo is a container slot on any network component, and also on each child device component. Properties under this slot are used to populate the alarm record when the network or device does not respond to the network’s monitor ping. These properties work in the same fashion as those in an alarm extension for any control point. See [“About alarm extension properties”](#) on page 5-3 for AlarmSourceInfo property descriptions.

For more details, refer to these topics in the *Drivers Guide*:

- [“About network Alarm Source Info”](#)
- [“Device Alarm Source Info”](#)

alarm-BooleanChangeOfStateAlarmExt


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

See the following links for more information:

- [“Types of alarm extensions”](#) on page 3-15

- [“About alarm extensions and components”](#) on page 5-3
- [“About alarm extension properties”](#) on page 5-3
- [“About alarm extension manager”](#) on page 5-26


alarm-BooleanCommandFailureAlarmExt

 BooleanCommandFailureAlgorithm implements command failure alarm detection algorithm for Boolean objects as described in BACnet. If feedback and output values of the point are not equal for timeDelay duration, an offnormal alarm is generated. This alarm extension is available in the Extensions folder of the alarm palette.

See the following links for more information:

- [“Types of alarm extensions”](#) on page 3-15
- [“About alarm extensions and components”](#) on page 5-3
- [“About alarm extension properties”](#) on page 5-3
- [“About alarm extension manager”](#) on page 5-26


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.


See the following links for more information:

- [“About the console recipient”](#) on page 5-36
- [“Types of alarm recipients”](#) on page 5-36
- [“About the alarm service”](#) on page 5-7

alarm-FaultAlgorithm

 FaultAlgorithm is the super-class of all Fault detection mechanisms defined by Niagara, 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. See [“Types of alarm extensions”](#) on page 3-15, and also [“About alarm extensions and components”](#) on page 5-3.


alarm-EnumChangeOfStateAlarmExt

 EnumChangeOfStateAlarmExt 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 generate an alarm. This alarm extension is available in the Extensions folder of the alarm palette.

See the following links for more information:

- [“Types of alarm extensions”](#) on page 3-15
- [“About alarm extensions and components”](#) on page 5-3
- [“About alarm extension properties”](#) on page 5-3
- [“About alarm extension manager”](#) on page 5-26


alarm-EnumCommandFailureAlarmExt

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

See the following links for more information:

- [“Types of alarm extensions”](#) on page 3-15
- [“About alarm extensions and components”](#) on page 5-3
- [“About alarm extension properties”](#) on page 5-3
- [“About alarm extension manager”](#) on page 5-26

alarm-LinePrinterRecipient


 The LinePrinterRecipient component provides the capability to print alarms to line printers that are attached to a station running on a Windows platform, or to such remote (networked) printers known to its Windows OS. Alerts may be generated if the printing of an alarm fails, but the lineprinter recipient does not print alarms that it generates itself.

See the following links for more information:

- [“About the lineprinter recipient”](#) on page 5-45
- [“Types of alarm recipients”](#) on page 5-36


- [“About the alarm service”](#) on page 5-7

alarm-MemoryAlarmService


 MemoryAlarmService 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 alarm service. The MemoryAlarmService is available in the alarm palette.

- Refer to [“About memory alarm service”](#) on page 5-8 for more details.


alarm-OffnormalAlgorithm

 OffnormalAlgorithm is a super-class for all algorithms that check for “off normal” conditions, and contains properties that specify “alarm” conditions. An OffnormalAlgorithm is under each *type* of alarm extension, along with a FaultAlgorithm container. See [“Types of alarm extensions”](#) on page 3-15, and also [“About alarm extensions and components”](#) on page 5-3.

alarm-OutOfRangeAlarmExt

 OutOfRangeAlarmExt 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. See [“Types of alarm extensions”](#) on page 3-15, and also [“About alarm extensions and components”](#) on page 5-3.

alarm-PrinterRecipient

 (AX-3.7 and later) The PrinterRecipient 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 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.


The new PrinterRecipient provides additional font and style settings, and works well with the “multi-line” alarm message text properties of alarm extension components, introduced starting in AX-3.7.

Alerts may be generated if the printing of an alarm fails, but the printer recipient does not print alarms that it generates itself.

See the following links for more information:

- [“About the printer recipient”](#) on page 5-46
- [“Types of alarm recipients”](#) on page 5-36
- [“About the alarm service”](#) on page 5-7


alarm-StationRecipient

 Recipient for another station. This class receives alarms and formats them for another station. The StationRecipient is available in the alarm palette.


See the following links for more information:

- [“About the station recipient”](#) on page 5-41
- [“Types of alarm recipients”](#) on page 5-36
- [“About the alarm service”](#) on page 5-7

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”](#) on page 3-15, and also [“About alarm extensions and components”](#) on page 5-3.

alarm-StringChangeOfValueAlarmExt

 StringChangeOfValueAlarmExt provides the ability to alarm upon inclusion *or* exclusion of a particular string value, or more accurately, “regular expression” (regex) of a point’s Out slot (string type). By default, matching is case sensitive, but this is configurable using the “Case Sensitive” property in the extension’s “Offnormal Algorithm” and “Fault Algorithm” container slots.

By default, other configuration property values under Offnormal Algorithm and Fault Algorithm are:

- Expression — a value of: `. *`
This is the regex value for “any text”.
- Normal On Match — `true`

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

This alarm extension is available in the “Extensions” folder of the AX-3.6 and later alarm palette.

See the following sections for more StringChangeOfValueAlarmExt related details:

- [“Example”](#)
- [“Regex notes”](#)

Also see these sections for general alarm extension information:

- [“Types of alarm extensions”](#) on page 3-15
- [“About alarm extensions and components”](#) on page 5-3
- [“About alarm extension properties”](#) on page 5-3
- [“About alarm extension manager”](#) on page 5-26

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”.

Regex 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” (regex) 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 regex 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 NiagaraAX 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

Note: *In the 2013 “update releases” of NiagaraAX, including AX-3.7u1, password storage changes were made to increase security. These changes affect the backup dist content, and more importantly how backups can be restored. However, in AX-3.8 changes were made that simplify backup dist usage, although there are considerations about JACE platform credentials (which are no longer stored in a backup .dist file.) For details, refer to the NiagaraAX 2013 Security Updates document.*

 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, png, and so forth.

Also, the backup dist contains the zipped "nre-config" for that host (including license and certificates files), as well as pointers to the installed "nre-core," OS, and JVM, each by version. Essentially, a backup dist provides a "configuration snapshot" of the entire JACE platform in zipped "dist" file format. This allows for a complete image restoration.



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*.


Note: *In update releases (including AX-3.7u1), in this "replicated station" use case it is recommended that you modify the original saved backup dist file, before installing it in multiple hosts. Otherwise, station security is compromised. However in AX-3.8, improvements were made such that edits to an AX-3.8 backup .dist file are unnecessary. For related details, see the sections "Station archiving changes" and "Making modifications to archived station files" in the NiagaraAX 2013 Security Updates document.*

Also see "Restoring a backup dist" in the *NiagaraAX 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 (0-100) 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”](#) on page 10-5
- [“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](#)” on page 6-2.

baja-CategoryService

 The CategoryService is the station container for all [Category](#)(ies), which represent logical groupings to which you can assign components, files, and histories. 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. For more CategoryService details, see “[CategoryService](#)” on page 6-38.



baja-Component

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


Using Containers Containers allow you to logically group components. The current container is the component that contains components in the display window. A container may be selected as the current container by one of the following methods:

- Double-click the component in the Nav tree.
- Right-click the component in the Nav tree (which brings up a menu) and select a view.
- Right-click the component in a wire sheet (which brings up a menu) and select a view.


Container components include the following:

-  Component can be used as a general container for components. It allows you to place components and links in a container.
-  Page is a special component used to create a map of name/Ref pairs as dynamic slots.


baja-DataFile

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

baja-Directory

 Directory is used to represent directories in File space implementations.

baja-FileSystem

 FileSystem is a File space for the local machine's file system.

baja-Folder

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

baja-Format

Format (or “BFormat”) is used to format objects into strings using a standardized formatting pattern language. The format string is normal text with embedded scripts denoted by the % percent character (use %% to insert a real %). A script is one or more calls chained together using the . dot operator. Calls are mapped to methods using reflections. Given 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(bajai: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 *NiagaraAX 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” on page 9-10).

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](#)" on page 1-6, including subsections "[About module characteristics](#)" on page 1-7, as well as "[Working with modules](#)" on page 9-24, including subsections "[Creating a new \(.jar\) module](#)" on page 9-24.


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](#)" on page 1-6, and also the engineering notes document *NiagaraAX Synthetic 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](#)" on page 6-8 and "[About permission levels](#)" on page 6-10. 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 `bajoui.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 *NiagaraAX Developer Guide* for developer information. The Station is available in a fox connection to a NiagaraAX 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 (0-100) 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 (.jar) that allows the creation of memory-resident modules and types programmatically at station runtime. Synthetic modules differ from “standard” .jar (non-synthetic) modules in a number of ways, and have a special default “Synthetic Module File View” for editing contents.

For details, refer to the *NiagaraAX Synthetic Modules* engineering notes document.

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”](#) on page 6-4
- [“User”](#) on page 6-30 (all properties)
- [“UserService security notes”](#) on page 6-11


Also, see [“Security model overview”](#) on page 6-2.

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”](#) on page 6-18 and [“Password reset”](#) on page 6-21. 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 NiagaraNetwork. For more details, see [“Network users”](#) on page 6-7 and [“About User prototypes”](#) on page 6-36.

baja-UserService

 UserService is the service you use to add, edit, delete, and otherwise manage users of the station. The [UserManager](#) is its primary view. Also available is a [PermissionsBrowser](#) view, in which you can review and change permissions for any user’s assigned [Category](#)(ies). See [“Users and security”](#) on page 6-4 and [“UserService”](#) on page 6-24 for more details. 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.

Note: *To facilitate user management, the UserService component has a unique permissions scheme. See [“UserService security notes”](#) on page 6-11 for details.*

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”](#) on page 6-18 and [“Password history \(unique passwords\)”](#) on page 6-23. 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 NiagaraAX. For a general explanation about Baja virtual components, see the *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 NiagaraAX drivers. For a general explanation, see the *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”](#) on page 3-1 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”](#) on page 3-1, and [“About writable points”](#) on page 3-21 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”](#) on page 3-1 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”](#) on page 3-1, and [“About writable points”](#) on page 3-21 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”](#) on page 3-1 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”](#) on page 3-1, and [“About writable points”](#) on page 3-21 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”](#) on page 3-1 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”](#) on page 3-1, and [“About writable points”](#) on page 3-21 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 NiagaraAX release level. It also applies to any NiagaraAX platform running a pre-AX-3.7 release that require SSL capability. For more details, refer to the engineering notes document *NiagaraAX CryptoService (SSL)*.

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. For related details, see the NiagaraAX SSL Connectivity Guide.

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”](#) on page 5-15.

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 is available starting in NiagaraAX-3.5. 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.
- **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 (available starting in NiagaraAX-3.5) 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](#) 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 starting in NiagaraAX-3.5. 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 (NiagaraAX-3.5 and later) displays a value to indicate how many emails did not successfully send.
 - **Last Discard**
This read-only property (NiagaraAX-3.5 and later) displays a date and time value to indicate when the last email did failed to send.
 - **Last Discard Cause**
This read-only property (NiagaraAX-3.5 and later) 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](#)
- [PxFFile](#)
- [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”](#) on page 1-12.


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 flr module

- [FLR](#)

flr-FlrService


 The FloatingLicenseRepository (FLR) component provides the service for setting up a floating license server in NiagaraAX. The component has one action “Reload”. This action reloads or “reads” the license pack file (LPF) into the FLR server application. This component is found in the flr palette.

For details, refer to the *NiagaraAX Floating License Repository (FLR) - Engineering Notes* document.

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”](#) on page 6-12.


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”](#) on page 3-16 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”](#) on page 3-16 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” on page 3-16 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” on page 3-16 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” on page 4-30
- “About the History Group property sheet view” on page 4-31
- “About the History Group Manager view” on page 4-31

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” on page 4-30
- “About the History Group property sheet view” on page 4-31
- “About the History Group Manager view” on page 4-31

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-History-PointListItem” 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 lifecycle, 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 History-Service. The HistoryService is available in the history module.

See the [“About the history service”](#) on page 4-2 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”](#) on page 4-32.


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”](#) on page 3-16 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”](#) on page 3-16 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”](#) on page 3-16 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”](#) on page 3-16 for more information about this extension.


Components in net module

- [InternetAddress](#)
- [HttpProxyServer](#)

net-InternetAddress

InternetAddress models an Internet address which is a composite of a hostname (or raw IP address) and a port number.

net-HttpProxyServer

 HttpProxyService (available starting in NiagaraAX-3.4) 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

 Available in NiagaraAX-3.5 and later, the On Call Service is a customization of the alarm service that allows you to send alarm notifications to users based on a priority list (“Contact List”). This service 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 component is available in the onCall module. Also see “[About the On Call Service](#)” on page 5-8.

Actions Actions include **Execute**.


onCall-OnCallRecipient

 The On Call Recipient (AX-3.5 and later) manages the transfer of alarms between the alarm service and On Call Contacts. For example, a station may send alarm notifications by email or cell phone to one or more contacts if they are specified in the currently assigned On Call Contact List. The On Call

Recipient component is linked to an Alarm Class component and provides a place to specify the scheduling details and other routing options. The OnCallRecipient is available in the onCall module. Also see [“About the On Call Recipient”](#) on page 5-42.

Actions Actions include **Route Alarm**.

onCall-OnCallSchedule

 The OnCallSchedule (On Call List Schedule) is used to set times for making On Call Lists active. The OnCallSchedule component is available in the onCall module.

Actions Actions include **Cleanup**.

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 NiagaraAX modules from [Program](#) components. This provides a mechanism to version and provision Program modules just like any other NiagaraAX 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”](#) on page 7-7 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”](#) on page 7-5 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”](#) on page 7-5 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”](#) on page 7-5 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](#)” on page 7-5 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 preconfigured 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

 WebService 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.

Note: In AX-3.8, several WebService properties in a new station (via **New Station** tool in Workbench) have different defaults than in previous NiagaraAX releases. See “[New station WebService defaults in AX-3.8](#)” on page 10-30. In addition, any AX-3.8 station that provides “Web Workbench” to browser clients requires those browser client PCs to have Java configured with “Unlimited Strength Policy Files”. For details refer to “Additional AX-3.8 client-side Java installation” in the NiagaraAX 2013 Security Updates document.

Properties of the WebService determine the port usage and authentication methods of the station's web server, among other things. Properties include:

- **Status**
Read-only property that displays the status of the WebService.
- **Fault Cause**
If status is fault, provides an error message that indicates the reason.
- **Enabled**
By default `true` (enabled); if necessary it can be set to `false` (disabled).
Note: To increase security in a scenario where the WebService is not used, for example in a JACE station where browser access is not needed, set this to `false` to disable this service.
- **Http Port**
TCP port the service listens on for HTTP client connections, where port 80 is the default.
- **Http Enabled**
Boolean to determine if HTTP client requests are processed, it is `true` by default (before AX-3.8). If `false`, and HTTPS is enabled, HTTP requests are redirected to HTTPS. See “[New station WebService defaults in AX-3.8](#)” on page 10-30.
- **Https Port**
TCP port the service listens on for HTTPS (secure) client connections, where port 443 is the default.

- **Https Enabled**
Boolean to determine if HTTPS client requests are processed, it is `false` by default (before AX-3.8). See [“New station WebService defaults in AX-3.8”](#) on page 10-30.
 - Starting in AX-3.7, any “Hotspot JVM” NiagaraAX host can be configured for secure (SSL/TLS) connections for browser (HTTPS) access, as well as secure Fox connections (Foxs) and platform connections (platformssl). Complete details are in the *NiagaraAX SSL Connectivity Guide*.
 - Any “IBM J9” host (earlier JACE models) can provide SSL connections to the station via browser access, regardless of NiagaraAX release level. The initial station login screen is always encrypted (secure), as well as all Hx access of the station. However, if browser access uses a Workbench profile (WbApplet), data transmitted over Fox is not secured with an SSL socket. A different architecture is used, where station requires the **CryptoService**, with the `crypto` module installed. For details, refer to the *NiagaraAX CryptoService (SSL)* engineering notes document.
- **Https Only**
If both HTTP and HTTPS are enabled, all HTTP requests are redirected to the HTTPS port. See [“New station WebService defaults in AX-3.8”](#) on page 10-30.
- **Https Min Protocol**
For “Hotspot JVM” SSL only. Specifies the security protocol to use, where the default is either standard protocols (SSLv3+TLSv1). Other choices are one or the other: SSLv3 or TLSv1.
- **Https Cert**
For “Hotspot JVM” SSL only. Specifies the server certificate to use for HTTPS connections, which may be different (or the same) as the one used for secure platform and secure fox connections.
- **Authentication Scheme**
The authentication scheme used for HTTP requests, where choices are:
 - Cookie Digest — (default) The most secure (and recommended) authentication method. However, in some scenarios another authentication method may be necessary.
 - Cookie — Applies if the station uses the **LdapUserService** or **ActiveDirectoryService** instead of the standard **UserService**. Also applies if using “domain-wide cookies authentication”, in which case the WebService’s property Single Sign On Enabled (new in AX-3.7u1) also should be set to `true`. For related details, see [“Domain-wide cookies authentication”](#) on page 6-15.
 - Basic — Typically not used, except if the client did not support one of the other (preferred) authentication schemes.
- **Gzip Enabled**
Default is `true`, to enable gzip compression for basic text types: html, xml, css, js, and so on. Gzip support is new starting in AX-3.7. If set to `false`, gzip compression is not used.
- **Log File Enabled**
Default is `false`. If set to `true`, a log file of HTTP transactional messages from client connections is created in the station’s specified log file directory.
- **Log File Format**
File format for log files (if enabled). Choices include:
 - NCSA Common Log Format (default)
 - NCSA Extended Log Format
 - W3C Extended Log Format
- **Log File Directory**
Default is `file: ^httpd`. The folder in the station’s file space in which log files are stored. Log file names use a `YYMMDD.log` (date) convention, such as `130501.log` for a file created May 1, 2013.
- **Log File Policy**
Determines when a new log file is started, whether Daily (the default) or else Weekly, Monthly, or Limited Size.
- **Log Maximum Size**
Specifies the maximum size of a log file, in MB, where 100MB is the default.
- **Auto Login Enabled**
Boolean to determine if auto-login can occur using cookies, it is `false` by default. Only valid if the authentication scheme is set to Cookie.
- **Single Sign On Enabled**
Boolean to determine if “domain-wide cookies authentication” can occur using cookies, it is `false` by default. Only valid if the authentication scheme is set to Cookie. Other WebService modification is necessary for this feature, see [“Domain-wide cookies authentication”](#) on page 6-15.
- **Login Template**
Specifies the template for the browser login page. If null (default), the default login template is used.

- **Tunneling Enabled**
Boolean to specify if this station provides HTTP tunneling to other stations (default is `false`). If set to `true` (and the host is licensed for tunneling), the station can act as a proxy and redirect HTTP requests/responses where appropriate. Note you may also need to enable Fox tunneling for full support. For more details, refer to the *Fox Tunneling and HTTP Tunneling* engineering notes document
- **Proxy Authentication When Tunneling**
Boolean to specify if authentication is required at this (proxy) station before rerouting HTTP tunnel requests to other stations (default is `false`). If set to `true` authentication is required, otherwise this proxy station simply reroutes all HTTP tunnel requests without requiring additional authentication.
Note: *This is required only when using Cookie Digest on the target station when:*
 - A non-default Fox port is used on the target station, *or*
 - The target station has not been upgraded to AX-3.7u1 or later.
 - The target station has a different password.
- **Cookie Digest Session Timeout**
Applicable only if authentication type is Cookie Digest. Specifies the maximum amount of time, in minutes, without activity during a browser session, whereafter the session is considered timed out—the user must log in again).
The default is 5 minutes. If set to 0, there is no timeout (but typically *not recommended*, as this could be a security risk).
- **Client Environments**
(New starting in AX-3.7) Container for Mobile Client Environment (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 "About the Mobile Client Environment (mobile) property" in the *NiagaraAX Mobile Guide*.

New station WebService defaults in AX-3.8 By default in AX-3.8, any new station created in Workbench using the **New Station** tool (wizard) is created for SSL access only, including both its **WebService** and **FoxService** (the latter a container under its **NiagaraNetwork** component).

A "Use secure connections (recommended)" checkbox in the **New Station** wizard enables this. If left checked, the station's **WebService** has these properties set as follows:

- Http Enabled: `false`
- Https Port: 443 (or whatever Https Port was specified in the New Station wizard).
- Https Enabled: `true`
- Https Only: `true`

Sometimes when using the AX-3.8 **New Station** wizard you may wish to *uncheck* secure connections, which makes a **WebService** (and **FoxService**) with *different* values for the properties above. You might do this when making a station to be installed in a "J9 JVM" JACE (JACE-2/4/5 series), which cannot use Foxs (Fox SSL), but *could* use SSL for the **WebService**, if you also configured it with the station-based **CryptoService**.

In any case, after creating a new station, you can always re-edit its **WebService** (and **FoxService**) properties *offline* if needed—before installing it in the target host platform. For related details "[New Station wizard](#)" on page 8-10.

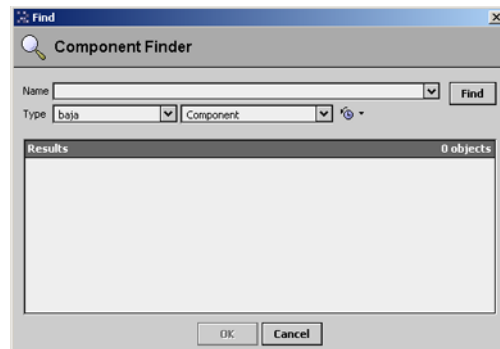
Components in workbench module

- [ComponentFinder](#)
- [KioskService](#)
- [WbFieldEditorBinding](#)
- [WbViewBinding](#)
- [WsOptions](#)


workbench-ComponentFinder

The ComponentFinder dialog provides a means of finding Components.

Figure 10-1 Sample Window




workbench-KioskService

 workbench-KioskService is used to enable the Kiosk mode. Kiosk mode provides a way to run a NiagaraAX 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 WbField-Editor (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 ([Property-Sheet](#), [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.

CHAPTER 11

Plugin Guides

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 wutil module](#)
- [Plugins in workbench module](#)

Plugins in alarm module

- [alarm-AlarmConsole](#)
- [alarm-AlarmDbMaintenance](#)
- [alarm-AlarmDbView](#)
- [alarm-AlarmClassSummary](#)
- [alarm-AlarmExtManager](#)
- [alarm-AlarmInstructionsManager](#)
- [alarm-AlarmPortal](#)






alarm-AlarmConsole



 The Alarm Console allows you to view and acknowledge Alarms. It can be started by Right-clicking the alarmRecipient in the [AlarmService](#) and selecting **Views > Alarm Console**.


Viewing Alarms Multiple Alarms from a single source are visualized as a single row in the alarm console. Color coded icons provide a visual indication for unacknowledged/inAlarm, acknowledged/inAlarm, and unacknowledged/not inAlarm.


-  **alarm**
-  **redAlarm**
-  **greenAlarm**
-  **orangeAlarm**


Double-clicking a row in the Table shows the history of the alarms associated with that point. You can sort the Alarms in order of any column by pressing the **column bar** (once for ascending, twice for descending). Available commands include:

-  [Acknowledge](#)
-  [Hyperlink](#)
-  [Notes](#)
-  [Silence](#)
-  [Filter](#)

Acknowledging Alarms Select an Alarm by Left-clicking it. In order to acknowledge an Alarm, press the  **Alarm Acknowledge** toolbar button or select **Alarm > Acknowledge** from the Menu. You can select multiple Alarms and acknowledge them simultaneously. In order to silence Alarms, press the  **Alarm Silence** toolbar button or select **Alarm > Silence** from the Menu.

Acknowledge  **Acknowledge** allows you to acknowledge the currently selected Alarm(s).





Hyperlink  **Hyperlink** allows you to hyperlink to Alarm URL.

Silence  **Silence** allows you to silence the currently selected Alarm(s).

Filter  **Filter** allows you to filter alarms.





Alarm Report Dialog From the Alarm console, you can view the Alarm Report to see all alarms on the point. Select an Alarm and Double-click it to see the Alarm Report.


You can sort the Alarms in order of any column by pressing the **column bar** (once for ascending, twice for descending). Available commands include:

-  [Acknowledge](#)
-  [Hyperlink](#)
-  [Notes](#)
-  [Filter](#)


Viewing Alarm Record From the Alarm Report Dialog, you can view the Alarm record to see all information on the Alarm. Select an Alarm and Double-click it to see the Alarm record.

Available commands include:

-  [Acknowledge](#)
-  [Hyperlink](#)
-  [Notes](#)
-  [Filter](#)

 **Notes** **Notes** allows you to add user notes to Alarm records.


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** allows you to manage Alarm extensions.






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 (Wb Web Profile) — !backups**
A subdirectory “backups” under the Niagara subfolder of your installed Java 2 runtime environment (Java plugin).
For example: C:\Program Files\Java\j2re11.4.2_05\niagara\backups.
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

- [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 Slot 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.

Table 11-1 *Flags*

Flag	Char	Applies	Description
readonly	r	P	The <code>readonly</code> flag is used to indicate slots which are not accessible to users.
transient	t	P	Transient properties do not get persisted when saving a object graph to the file system. Transient properties are usually also readonly, unless they are designed to be a linkable input slot.
hidden	h	P,A,T	Hidden slots are designed to be invisible to the user, and exist only for Java developers. User interfaces should rarely display hidden slots.
summary	s	P	Summary properties are the focal points of any given BComponent. This flag is used by user interface tools to indicate primary properties for display. This might be as a columns in a Table, or as a glyph in a graphical programming tool.
async	a	A	By default Action are invoked synchronously on the callers thread. By using the <code>async</code> flag on an Action, invocations are coalesced and executed asynchronously at some point in the near future on the engine's thread.
noExecute	x	P	No execute properties prevents start/stop from recursing on properties with this flag set.
defaultOnClone	d	P	Specifies that when an object is cloned via the <code>newCopy()</code> method these properties retain their default value, not the clone source's value.
confirmRequired	c	A	When the Action is invoked by a user, a confirmation 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.

Flag	Char	Applies	Description
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 a 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

- Q 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

- Q 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

- Q The HistoryGroupManager provides a tabular view of all history groups. It is the default view of the [HistoryGroupings](#) component and provides information about the groups contained in the History-Group component.

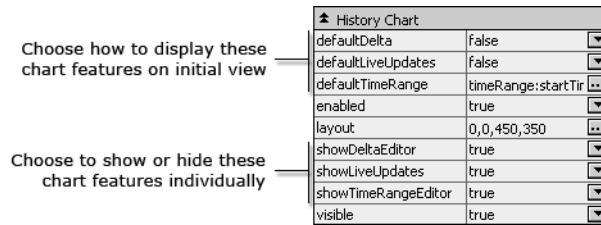
Refer to the following sections for related information:



- [“About history grouping”](#) on page 4-30
- [“About the History Group property sheet view”](#) on page 4-31
- [“About the History Group Manager view”](#) on page 4-31

history-HistoryChart


- Q 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, as shown in [Figure 11-1](#).


Figure 11-1 History Chart display properties are configurable in the Px page view

- **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 [HistoryService](#) 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”](#) on page A-8
- [“About the history extension manager view”](#) on page 4-3
- [“About the history extension manager popup menu items”](#) on page A-11

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](#)” on page 4-12 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](#)” on page 2-13. 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://bajau/doc/style.css' type='text/css'/>`
- meta - *not supported*
- object - *not supported*
- ol - ordered list
- p - paragraph Ex: `<p class='note'>Note</p>`
- pre - preformatted text
- span - *not supported*
- style - *not supported*
- table - table (Attributes `align`, `border`, `bordercolor`, `cellpadding`, `cellspacing` and `width`)
- tbody - *not supported*
- td - table data cell supports `ALIGN` (left, right, and center), `BGCOLOR`, `COLSPAN`, `ROWSPAN` and `WIDTH`
- tfoot - *not supported*
- th - table header cell supports `ALIGN` (left, right, and center), `BGCOLOR`, `COLSPAN`, `ROWSPAN` and `WIDTH`
- thead - *not supported*
- title - document title
- tr - table row supports `ALIGN` (left, right, and center) and `BGCOLOR`
- tt - teletype or monospaced text style
- ul - unordered list

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 “
- &lquo; - 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://bajau/doc/style.css' type='text/css' />
</head>
```

See the default stylesheet used for Niagara developer documentation: <module://bajau/doc/style.css> or the CSS stylesheet used for this document at <docbook.css>.

Pseudo-classes and Pseudo-elements Anchor pseudo-classes include:


- A:link unvisited links *unsupported*
- A:visited visited links *unsupported*
- A:active active links *unsupported*

CSS1 Properties Stylesheet elements supported include:

- background - The 'background' property is a shorthand property for setting the individual background properties (i.e., 'background-color', 'background-image', 'background-repeat', 'background-attachment' and 'background-position') at the same place in the style sheet.
- background-attachment *unsupported*
- background-color - This property sets the background Color of an element.
- background-image *unsupported*
- background-position *unsupported*
- background-repeat *unsupported*
- border *unsupported*
- border-bottom *unsupported*
- border-bottom-width *unsupported*
- border-color *unsupported*
- border-left *unsupported*
- border-left-width *unsupported*
- border-right *unsupported*
- border-right-width *unsupported*
- border-style *unsupported*
- border-top *unsupported*
- border-top-width *unsupported*
- border-width *unsupported*
- clear *unsupported*
- color - This property describes the text Color of an element (often referred to as the foreground Color).
- display *unsupported*
- float *unsupported*
- font *unsupported*
- font-family *unsupported*
- font-size *unsupported*
- font-style *unsupported*
- font-variant *unsupported*
- font-weight *unsupported*
- height *unsupported*

- letter-spacing *unsupported*
- line-height *unsupported*
- list-style-image *unsupported*
- list-style-position *unsupported*
- list-style-type *unsupported*
- margin *unsupported*
- margin-bottom *unsupported*
- margin-left *unsupported*
- margin-right *unsupported*
- margin-top *unsupported*
- padding *unsupported*
- padding-bottom *unsupported*
- padding-left *unsupported*
- padding-right *unsupported*
- padding-top *unsupported*
- text-align
- text-decoration *unsupported*
- text-indent *unsupported*
- text-transform *unsupported*
- white-space *unsupported*
- width *unsupported*
- word-spacing *unsupported*

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](#)” on page 5-10.

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](#)” on page 5-12.


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](#)” on page 5-13.

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](#)” on page 2-13. 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**, a component available starting in AX-3.5. It lets you create a NiagaraAX *module* from one or more **Program** components, such that they may be versioned and provisioned just like other NiagaraAX 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](#)” on page 2-13. 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](#)” on page 7-19.

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](#)” on page 7-13.

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](#)” on page 7-7.

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”](#) on page 7-23.

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](#)” on page 2-13. 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 `bajau: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](#)
- [wbutil-CategoryManager](#)
- [wbutil-CategorySheet](#)
- [wbutil-PermissionsBrowser](#)
- [wbutil-ResourceEstimator](#)
- [wbutil-ToDoList](#)
- [wbutil-UserManager](#)

wbutil-CategoryBrowser


 CategoryBrowser is the default view of a station's CategoryService. Use it to assign objects in the station into categories. This view presents an expandable tree table where the table columns are the *available categories*, and rows are the categorizable objects.

- Yellow rows are objects explicitly assigned into a category.
- Dimmed rows represent objects *inheriting* their parent's categories.

Note: For any object except the three root objects: station (components), file, and history, you can check to inherit its parent's categories. For root objects, you must select at least one category.


For any component, you can alternately use its [CategorySheet](#) view to assign it categories. See ["Categories and security"](#) on page 6-3 for details on how categories affect security, and ["Category Browser"](#) on page 6-40 for more details on this CategoryService view.

wbutil-CategoryManager


 CategoryManager is a view on the CategoryService that allows you to create, enable and delete categories. Categories are used by the security model, as well as for future features. You can then use the [CategoryBrowser](#) view to centrally assign objects to categories. Or, at the individual component level, use a component's [CategorySheet](#) view to assign it to categories.

You can assign an object to many categories at the same time. Each object stores its own categories. See [“Categories and security”](#) on page 6-3 for details on how categories affect security, and [“Category Manager”](#) on page 6-39 for more details on this CategoryService view.

wbutil-CategorySheet

 CategorySheet is used to assign a component into one or more categories (or configure it to inherit categories). Every component has a CategorySheet view. See [“Categories and security”](#) on page 6-3 for details on how categories affect security, and [“Category Sheet”](#) on page 6-41 for more details on this component view.

wbutil-PermissionsBrowser


 PermissionsBrowser is an available view on the UserService, for reviewing the permissions any user has on each object. It provides an expandable tree table (similar to the [CategoryBrowser](#) view of the CategoryService). In the PermissionsBrowser, columns represent station *user accounts*, and rows are the objects in the station, with each table cell showing user permissions.

- Yellow rows are objects explicitly assigned with permissions.
- Dimmed rows represent objects *inheriting* their permissions.

Double-click a cell to bring up the permissions dialog for that user (unless user has super user rights). This allows you to globally change that user's permission levels for any category in the station.


See [“Permissions and security”](#) on page 6-8 for details on how permissions affect security, and [“Permissions Browser”](#) on page 6-34 for more details on this UserService view.

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.


For more information, refer to [“Resource Estimator”](#) on page 8-12.

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.

For more information, refer to [“Todo List”](#) on page 8-15.

wbutil-UserManager


 The UserManager is the primary view of the UserService. You use it to add, edit, and delete users for accessing the station. See [“Users and security”](#) on page 6-4 for details on how users affect security, and [“User Manager”](#) on page 6-28 for more details on this UserService view.

For an overview on the station security model, see [“Security model overview”](#) on page 6-2.

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 control:ControlPoint
```








The Table options menu (see “[Table controls and options](#)” on page 2-18, 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](#)” on page 2-13.

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](#)” on page 2-13. 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 shows all the user visible Properties of the Component. This view also lets you change any Properties that you have authority to change. It can be used on a Component of a running Station or a Component in a Bog File.

In order to see Properties of Components in the **PropertySheet**, press the + sign to expand the Component.


When you want to enter a URL, you can Copy the Component or URL and Paste it into the **Property-Sheet** using the Paste shortcut Ctrl + V.


When a Property has been changed, its symbol will become red until you press  **Save**.

PropertySheet Menus The Workbench main menu functions are available. If you Right-click any Component in the PropertySheet, you can choose from the following:

- Views - show the view menu for 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
 -  • Rename
 -  • Reorder
 -  • [Config Flags](#)

PropertySheet Toolbar The Workbench toolbar contains navigation and editing buttons as described in [“About the toolbar”](#) on page 2-13.

 **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 FileOrd-Chooser.

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  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”](#) on page 2-13. 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 displayName to delete or rename Name Map and displayName_xx to delete or rename Name Map (xx) where xx is the Language Code.


 **Config Facets** [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 document *NiagaraAX Synthetic Modules* 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”](#) on page 2-13.

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 `bajau: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.

APPENDIX A

References

In this appendix (introduction)

This appendix contains the following reference information:

- [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](#)” on page 2-13 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](#)” on page 2-13.

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”](#) on page A-2
- **Edit menu**
See [“About the Edit menu”](#) on page A-4
- **Search menu**
See [“About the Search menu”](#) on page A-5
- **Bookmarks menu**
See [“About the Bookmarks menu”](#) on page A-6
- **Tools menu**
See [“About the Tools menu”](#) on page A-6
- **Window menu**
See [“About the Window menu”](#) on page A-6
- **Px Editor menu**
See [“About the Px Editor menu”](#) on page A-7
- **History Ext Manager menu**
See [“About the History Ext Manager menu”](#) on page A-8
- **Help menu**
See [“About the Help menu”](#) on page A-8




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 > Open > Open 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 is 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.palette, 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 access 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  Delete 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.bog|bog:|slot:/`.
- **Manage Credentials**
 Manage Credentials is available in the main **Tools** menu by selecting **Manage Credentials...** You can **Reset** or **Remove** selected Credentials or **Remove All** Credentials. You can also **Open** selected Credentials.
- **Request License**
Request License is available in the main **Tools** menu by selecting **Request License...** You can request a license by submitting the form.

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” on page 2-2](#).
 - **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” on page 2-4](#).

-  **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”](#) on page 2-5.
-  **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”](#) on page 2-9.
- **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”](#) on page 2-6.
- **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”](#) on page 2-8.
- **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”](#) on page A-16 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:

- **Toggle View/Edit Mode**
This command toggles the active view between Px Editor (for editing) and Px View (view only). If there are unsaved changes in your Px file, you are prompted to save before switching from Px Editor to Px View.
- **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.

- **Set Target Media**
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 (HTML Px Media or Workbench Px Media). See [“About presentation media”](#) on page 1-14 for more details about media types.








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”](#) on page A-9.
- **Wire sheet**
For descriptions of the wire sheet popup menu items, refer to [“About the wire sheet popup menu items”](#) on page A-10
- **Property sheet**
For descriptions of the property sheet popup menu items, refer to [“About the property sheet popup menu items”](#) on page A-10.


- **Px Editor**
For descriptions of the Px Editor popup menu items, refer to [“About the Px Editor popup menu items”](#) on page A-10.
- **History extension manager**
For descriptions of the History extension manager popup menu items, refer to [“About the history extension manager popup menu items”](#) on page A-11.
- **Todo list**
For descriptions of the Todo list popup menu items, refer to [“About the Todo list popup menu items”](#) on page A-11.
- **Point Manager**
For descriptions of the Point Manager popup menu items, refer to [“About the point manager popup menu items”](#) on page A-12.

About the nav side bar popup menu items

 The nav sidebar provides a tree-type hierarchical view of the system. For an overview of the nav side bar, refer to [“About the nav side bar”](#) on page 2-6.

The nav side bar menu is the popup menu that displays command options when you right-click on a component item in the nav side bar. The nav side bar menu has the following options:

- **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.
-  **Cut**
See [“Types of edit commands”](#) on page A-13.
-  **Copy**
See [“Types of edit commands”](#) on page A-13.
-  **Paste**
See [“Types of edit commands”](#) on page A-13.
-  **Duplicate**
See [“Types of edit commands”](#) on page A-13.
-  **Delete**
See [“Types of edit commands”](#) on page A-13.
-  **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 reordering 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.
-  **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.

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"](#). Following are wire sheet specific popup menu options:

- **Arrange**
Provides options for aligning components on the wire sheet in order to make them easier to view:
 - **Arrange All**
Redistributes the layout of all components on the wire sheet.
 - **Arrange Selection**
Redistributes the layout of all selected components on the wire sheet.
- **Select all**
Selects all components on the active wire sheet.

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"](#). Following are wire sheet specific popup menu options:


- **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

The Px Editor popup menu displays when you right-click on an object in the Px Editor view. The menu commands are dimmed or available, depending on the type of object that you select.





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

- **New** (refer to ["About the Edit menu"](#) on page A-4)
-  **Cut** Ctrl + X (refer to ["About the Edit menu"](#) on page A-4)
-  **Copy** Ctrl + C (refer to ["About the Edit menu"](#) on page A-4)
-  **Paste** Ctrl + V (refer to ["About the Edit menu"](#) on page A-4)
-  **Duplicate** Ctrl + D (refer to ["About the Edit menu"](#) on page A-4)
-  **Delete** Delete (refer to ["About the Edit menu"](#) on page A-4)
-  **Edit Properties...** (refer to ["About the properties side bar"](#) on page 2-11)
-  **Align > Left**
This command is available when you select two or more objects in the Px Editor. Choose the **Left** command to align the left edges of two or more objects along a vertical line.
-  **Align > 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.
-  **Align > Top**
This command is available when you select two or more objects in the Px Editor. Choose the **Top** command to align the top edges of two or more objects along a horizontal line.
-  **Align > Bottom**
This command is available when you select two or more objects in the Px Editor. Choose the **Bottom** command to align the bottom edges of two or more objects along a horizontal line.
-  **Reorder > To Top**
This command is available when you select one or more objects in the Px Editor. Choose the **To Top** command to move the selected object to the top position (inside its parent object) in the Widget Tree. This command will place the object in front (with respect to the view pane, or z-axis) of all other objects in the parent object, but will not move the object out of its parent object.
- **Reorder > Up**
This command is available when you select one or more objects in the Px Editor. Choose the **Up** command to move the selected object one position higher in the Widget Tree and forward one position in the view pane. This command will not move the object out of its parent object.
- **Reorder > Down**
This command is available when you select one or more objects in the Px Editor. Choose the **Down** command to move the selected object one position lower in the Widget Tree and back one position in the view pane. This command will not move the object out of its parent object.

-  **Reorder > To Bottom**
This command is available when you select one or more objects in the Px Editor. Choose the **To Bottom** command to move the selected object to the bottom position (inside its parent object) in the Widget Tree. This command will place the object behind (with respect to the view pane, or z-axis) all other objects in the parent object, but will not move the object out of its parent object.
- **Border > Add Border**
This command is available when you select one or more objects in the Px Editor. Choose the **Add Border** command to wrap selected object(s) with a border pane. Each selection is wrapped in a separate border pane.
- **Border > Remove Border**
This command is available when you select one or more border panes in the Px Editor. Choose the **Remove Border** command to delete selected border pane(s).

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”](#) on page 4-26 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”](#) on page A-8 for information about the History Ext Manager menu. Refer to [“About the history extension manager view”](#) on page 4-3 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](#)” on page 8-15 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](#)” on page 2-15) 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](#)” on page A-13.
- **Copy**
See “[Types of edit commands](#)” on page A-13.
- **Paste**
See “[Types of edit commands](#)” on page A-13.
- **Duplicate**
See “[Types of edit commands](#)” on page A-13.
- **Delete**
See “[Types of edit commands](#)” on page A-13.
- **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 reordering 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 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”](#) on page 2-13.

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”](#) on page A-14 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”](#) on page A-14 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”](#) on page A-14 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”](#) on page A-15 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”](#) on page A-15 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”](#) on page A-15 for the listing of Todo list toolbar icons.





Standard toolbar icons

For an overview description of the toolbar, refer to [“About the toolbar”](#) on page 2-13. Following are the icons that appear in all views:

-  Back. See [“Back”](#) on page A-3 for details about the Back function.
-  Forward. See [“Forward”](#) on page A-3 for details about the Forward function.
-  Up Level. See [“Up Level”](#) on page A-3 for details about the Up Level function.
-  Recent Ords. See [“Recent Ords”](#) on page A-3 for details about the Recent Ords function.
-  Home. See [“Home”](#) on page A-3 for details about the Home function.
-  Refresh. See [“Refresh”](#) on page A-3 for details about the Refresh function.
-  Left Side Bar. See [“Side Bars”](#) on page A-6 for details about the Side Bars function.
-  Open. See [“About the File menu”](#) on page A-2 for details about all the Open functions.
-  Save Ctrl + S. See [“Save”](#) on page A-3 for details on the Save function.
-  Save Bog. See [“Save Bog”](#) on page A-3 for details on the Save Bog function.
-  Printing Ctrl + P. See [“Printing”](#) on page A-3 for details on the Printing function.
-  Cut Ctrl + X. See [“Cut”](#) on page A-13 for details on the Cut function.
-  Copy Ctrl + C. See [“Copy”](#) on page A-13 for details on the Copy function.
-  Paste Ctrl + V. See [“Paste”](#) on page A-13 for details on the Paste function.
-  Duplicate Ctrl + D. See [“Duplicate”](#) on page A-13 for details on the Duplicate function.
-  Delete Delete. See [“Delete”](#) on page A-13 for details on the Delete function.
-  Undo Ctrl + Z. See [“Undo”](#) on page A-13 for details on the Undo function.
-  Redo Ctrl + Alt + Z. See [“Redo”](#) on page A-13 for details on the Printing function.






Slot Sheet toolbar icons






For an overview description of the toolbar, refer to [“About the toolbar”](#) on page 2-13. 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”](#) on page 2-13. 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”](#) on page 2-10).
 - Widget Tree
Shows or hides the Widget Tree side bar (refer to [“About the widget tree side bar”](#) on page 2-10).
 - Properties
Shows or hides the Properties side bar (refer to [“About the properties side bar”](#) on page 2-11).
-  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](#)” on page 4-3. 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](#)” on page 4-15. 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](#)” on page 8-15. 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”](#) on page A-16:
- **nre commands**
These commands are used at the command line and may be typed in using the “nre” prefix. See [“nre \(station\) commands”](#) on page A-17.
- **wb commands**
These commands are used at the command line and may be typed in using the “wb” prefix. See [“wb \(Workbench\) commands”](#) on page A-17.
- **plat commands**
These commands are used at the command line and may be typed in using the “plat” prefix. See [“plat \(platform\) commands”](#) on page A-17:

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 to **set** to display all the environment variables.
 - Type **set <prefix>** to display all the environment variables that start with the specified prefix.
 - Type to **set <name>=** to remove the “named” variable.
 - Type to **set <name>=<value>** to set the “named” variable to the “value” specified.
- **which**
This command resolves a filename in a path. Type to **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.

APPENDIX B


Glossary

 **action** Action is a slot that defines a behavior that can be invoked on a component.

admin This term indicates one of two levels of permissions (admin and operator) available in NiagaraAX. Admin and Operator levels are also specified by the terms read, write, and invoke. So you may have permission-levels of: operator Read, operator Write admired, domineered, and so on.

administrator This term typically describes a system or application user that can perform certain high-level functions that other users cannot perform.


agent An agent is a special object type that provides services for other object types. Agents are registered on their target types via the module manifest and queried via the registry interface.

 **alarm** Alarms provide the ability to be notified when a special event occurs. See [AlarmConsole](#) or [AlarmPortal](#) for more information.

animate In NiagaraAX-3.x, animation allows you to change, or “update” graphics based on data values that come from object sources that are connected (or “bound”) to them. In the PxEditor, right-click a property and select animate. For more details, see the *NiagaraAX Graphics Guide* section “Animating Graphics”.

API API is a Application Programming Interface. It defines how software engineers access the capabilities of software like the Niagara Framework.


applet A small Java program that can be embedded in an HTML page. Applets differ from full-fledged Java applications in that they are not allowed to access certain resources on the local computer, such as files and serial devices (modems, printers, etc.), and are prohibited from communicating with most other computers across a network. The common rule is that an applet can only make an Internet connection to the computer from which the applet was sent. See also `BAppletView.bajadoc` for more technical information.


 **archive** Archive is a history that is stored in a different station from where it originated.

audit The Audit keeps a history of changes made by users. If enabled, it registers itself as the Auditor for the system when the service is started. One of the important aspects of security is the ability to analyze what has happened after the fact. See [history-AuditHistoryService](#) for more information.


BACnet BACnet refers to data communication protocol for Building Automation and Control Networks. Developed under the auspices of the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE), BACnet is an American national standard, a European pre-standard, and an ISO global standard. The protocol is supported and maintained by ASHRAE Standing Standard Project Committee 135. See the BACnet Guide for more information on BACnet in the Niagara Framework.


Baja Baja is a term coined from Building Automation Java Architecture. The core framework built by Tridium is designed to be published as an open standard. This standard is being developed through Sun's Java Community Process as JSR 60. See Baja vs. Niagara in the Developer Guide for more information.

 **Bajadoc** Bajadoc is the Baja reference documentation. Baja reference documentation includes both Java API details as well as Baja slot documentation. See [help-BajadocViewer](#) for more information.

 **binding** A Binding is used to bind a Widget to a data source. Bindings are bound to one Object using an Ord. Subclasses are registered as agents on the type of objects to which they can be bound to. For more details, see the *NiagaraAX Graphics Guide* section “Data binding”.

bog file A Bog file contains baja components. It can be a complete database or any collection of components. A bog file is a special file that can describe components in a database. All views can be used on components in a bog file just as if they were in a station. See “Bog Files” in the *NiagaraAX Developer Guide* for more information.


 **bookmark** Bookmark identifies a single bookmark. See [Bookmarks](#) and [About the bookmark side bar](#) for more information.

 **BooleanPoint** BooleanPoint defines a read only control point with only an output element. See also, “[Types of control points](#)” on page 3-1.

BQL BQL is an acronym for Baja Query Language. BQL has a select statement similar to the select statement of Structured Query Language (SQL). It is typically used to provide programmable access to information in Niagara. See “BQL” in the *NiagaraAX Developer Guide* for more details.


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

build The build tool is used to build the Niagara Framework itself. You may utilize this build tool to manage your own Niagara modules. See “build.html” in the Developer Guide for more information.


 **chart** A chart is a graphical representation of data. The [HistoryChart](#) is a view that allows you to view and configure charts of histories. See “[Types of history views](#)” on page 4-10 for more information.


client The client part of a client-server architecture. Typically, a client is an application that runs on one computer and relies on a server to perform some operations. For example, an email client is an application that enables you to send and receive e-mail.


collection A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. See [Collection.bajadoc](#) for more technical information.

 **color** Color stores a color specification via an alpha, red, green, and blue component. See [BColor.bajadoc](#) for more information.

command Command typically refers to an Action in the Niagara Framework. See “[About point actions](#)” on page 3-3 for more information on point commands.

 **component** A component is the primary building block of the Niagara Framework. See also, “[About components](#)” on page 1-9.

 **composite** A composite allows you to expose child slots of a component as slots on the parent. See “[About composites](#)” on page 3-23 for more information.


 **connect** Connect refers to entering a valid address for a system to establish a connection, a user name and password. A connection creates an active session for use of the system.

console Console provides a simple runtime console for managing and debugging the station. See “[About the console](#)” on page 2-13 for more information.

container Containers allow you to logically group components. The current container is the component that contains the components in the display window. See [baja-component Containers](#) for more information.

context Context is used in operations that may perform differently depending on their context. Examples include operations that may require auditing or localization.


control The control module provides normalized components for representing control points. All control points subclass from the ControlPoint base class. Control points are typically used with the driver framework to read and write points in external devices. See “Control” in the *NiagaraAX Developer Guide* for more information.


 **control point** In the narrowest terms, control points refer to the 8 point types found in the Baja control palette under the “Points” folder. See “[About control points](#)” on page 3-1.

In broader terms, control points include other components found in both the control and kitControl palettes. Most of these are "classed" from the 8 basic point types. See [“Other control components”](#) on page 3-2.

ControlPoint is the base class for all point types in the Baja control architecture. A ControlPoint typically maps to one value being read or written via a driver. All ControlPoints have a StatusValue property called "out". ControlPoints inherit from BooleanPoint, EnumPoint, NumericPoint and StringPoint.


If the predefined proxyExt is not a NullProxyExt then the point is considered a proxy point which means that it is a local representation of a point which actually exists in an external device. The driver framework is used to maintain synchronization.

 **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.


 **CSS** Cascading style sheet.

CSV The CSV ("Comma Separated Values") file format is used to exchange data between applications. CSV is one format option that may be used for exporting data from tables in some NiagaraAX views (for example, history tables).

daemon Daemon typically refers to the Niagara platform daemon, the server process required by a Niagara host to run a station. There is also a separate "dialup daemon" to handle dialup modem functions.

 **database** Database is used to store information. Examples of databases include the registry or a station database.

debug Debug refers to testing or correcting software. The NiagaraAX Program Debugger provides debug capability for program components.

 **delete links** This action, available from the workbench toolbar removes the selected links.

dependencies All modules include zero or one dependencies element. This element contains zero or more dependency elements which enumerate the module's dependencies. Dependencies must be resolved by the framework before the module can be successfully used. Each dependency has one required attribute. The name attribute specifies the globally unique name of the dependent module. A dependency may also specify a bajaVersion and/or a vendor version. If the bajaVersion attribute is declared then it specifies the lowest bajaVersion of the dependent module required. It is assumed that higher versions of a module are backward compatible, thus any version greater than the one specified in a dependency is considered usable. Likewise the vendor and vendorVersion may be specified to declare a dependency on a specific implementation of a module. The vendor attribute may be specified without the vendorVersion attribute, but not vice versa. The required embed attribute specifies whether the dependency should be enforced on embedded Niagara devices.


The [ModuleSpaceView](#) supports the ability to right click a module and display all its dependencies. These dependencies are as declared by the module's manifest. To compute the dependencies needed to use a specific API a new Dependencies command has been added to the [BajadocViewer Menus](#). This command will compute all the classes and modules required to use a specific class in the API.


device extensions DeviceExt is the abstract base class for device extensions which provide feature integrations between the device's native protocol and model and the NiagaraAX framework-normalized model. See BDeviceExt.bajadoc for more information.

directory Directory is the file type used to represent directories in "file space" implementations. See BDirectory.bajadoc for more information. See also, [“Types of space”](#) and [“Types of files”](#) on page 1-18.

disabled Status disabled means a user manually disabled the component using the enabled property (enabled = false). Disabled always propagates down. For example if you set the network to "disabled", that forces all child devices disabled, which then forces all proxy points under each device disabled. See also, [“About point status”](#) on page 3-18.

disconnect You may logoff a connection (for example, a platform, station, or application) by right-clicking the object in the tree and select **Disconnect**. This allows you to close a session without removing it from the tree. The "disconnected" object appears dimmed in the nav tree. Right-click and select **Connect** to reestablish a connection to the object.

 **discover** Discover allows you to find items that are defined using a driver's framework. Refer to the *Drivers Guide* sections “About Device Discover, Add, and Match (Learn Process)” and “About Point Discover, Add, and Match (Learn Process)” for related examples.

 **domain** Security domains may be assigned to any component or set of components in the system. For example, security domains may be used to assign components into application domains such as hvac, fire, and lighting or security domains can be used to assign components into customer groups such as tenant 1, tenant 2, etc.





down This term indicates a communication error at the network or device level. “Down” always propagates downward in a relational hierarchy. Down is controlled by the `pingFail` and `pingOk` methods. The reason why a device is down is always stored in the `health.lastFailCause` property.

download Download refers to passing data down to a device. You may download a station from its supervisor.

driver This term is used for software that provides a device-specific (or protocol-specific) framework for integration with another system. In NiagaraAX drivers have “manager” views that provide a consistent look and feel for configuring and learning of devices and points. The cornerstone of the driver tools is the ability to support batch edits and Learns. See also, “Driver Framework” in the *NiagaraAX Developer Guide*.

element The elements define the hierarchical structure of a document. The majority of elements have opening and closing tags. Within these tags, text or even the whole documents can be found. There are empty elements which contains only opening tags without any content.

An element may also model one single piece of information in the baja control architecture. The most common types of elements are:

-  • BooleanElement
-  • NumericElement
-  • EnumElement
-  • StringElement


All elements regardless of their value type share common attributes. Every element has a status property which provides a consistent mechanism to convey point status. Each element also contains a priority level for informational purposes. If the priority is `PriorityLevel.none`, this is not a prioritized element.

Elements are basic building blocks from which larger, more sophisticated control points are created. A control point contains one or more input elements and one or more output elements. Elements can also be linked between Baja Objects to move data.

enumeration Enumeration represents a value data item which has a fixed set of possible values. This set of values is called the Enumeration's Range. The standard enumerations include:


- alarm:AlarmRecordTypes
- alarm:AlarmTransition
- baja:Month
- baja:Weekday
- control:AlarmState
- control:AlarmType
- control:CollectionState
- control:DisableAction
- control:LoopAction
- control:NotifyType
- control:PriorityLevel
- control:Reliability
- control:TotalizationInterval
- driver:ArchiveMode
- driver:ArchiveResult
- driver:ArchiveState
- driver:PollFrequency
- driver:ProxyState

See [Enum Range Dialog](#) or `BEnum.bajadoc` for more information.

 **EnumPoint** EnumPoint defines a read only control point with only an output element. See also, “[Types of control points](#)” on page 3-1.


export Export refers to providing data from the system for external use. 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. The Niagara Framework also exports its Database to XML for backup and conversion.

extension Extensions allow plug-in functionality to support features such as alarming and historical data collection. See also, “[About point extensions](#)” on page 3-12.

 **facets** Facets contain meta data – or additional data about an object. For example, “units of measurement” is a type of facet. See also, “[About point facets](#)” on page 3-7.

fault Faults always propagate downward in a hierarchy relationship. The framework performs “fatal-Fault” checking such as making sure a device is under the right network type. Fatal faults are also the mechanism used to perform licensing checks. Developers can also manage fault condition using the `configFail()` and `configOk()` methods. The reason a component is in fault is always stored in the `faultCause` property.


feature A term that refers to a line in a license file that specifies a capability (“feature”) for the associated framework. A feature is uniquely identified by a vendor and feature name. It contains an expiration and a set of optional properties. See also, “*Workbench license manager*” in the *NiagaraAX-3.x Platform Guide*.


 **file** This term (and icon) represents a file object in the file system of a session. See also, “[Types of files](#)” on page 1-18.


file scheme 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 types:


- Authority Absolute: “//hostname/dir1/dir2”
- Local Absolute: “/dir1/dir2”
- Sys Absolute: “lib/system.properties”
- Station Absolute: “^config.bog”
- Relative: “myfile.txt”
- Relative with Backup: “../myfile.txt”


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


 **filter** Filter is used to select specified items. An example is the [Alarm Console Filter](#).


 **find** Find allows you to search for the selected string. The shortcut is F5. See Search Find for more information.

 **find files** Find Files allows you to find the all occurrences in the files. See Search Find Files for more information.

 **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).

 **flag** 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.

 **folder** This object is a folder container in the file system of a session. See also, [Folder](#).

format Format 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(bajauri:dialog.error)%"`

 **Fox** Fox is a proprietary protocol which is used for all network communication between Stations as well as between the workbench and stations. Fox is a multiplexed peer to peer protocol which sits onto of a TCP connection.

fox scheme 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 the port is not specified, then the default 1911 port is assumed.


FROM FROM is the mechanism to choose the extent. This may be a history or a class in a station. See [ExtentBuilder](#) for more information.


glyph A Glyph is a visual representation.


GotoFile In some views, this command allows you to go to a file that has been indexed with the file indexer. The shortcut is Ctrl + F7 (hold down Ctrl and press F7). See [Search Goto File](#) for more information.


GotoLine In some views, this command allows you to go to a line number in the file. The shortcut is Ctrl + G (hold down Ctrl and press G). See [Search Goto Line](#) for more information.


handle scheme The "h" Scheme is used to resolve a component by its handle. Handles are unique String identifiers for components within a componentSpace. Handles provide a way to persistently identify a component independent of any renames which modify a component's slot path.

 **Help** Help refers to the electronic documentation available. It includes electronic manuals as well as context specific help.

 **history** History is an ordered Collection of timestamped records. Each history is identified by a unique id. Histories can be periodically archived to a remote history database (archive). A history database is a set of histories. History: is also used as a Scheme in Ords to refer to collected histories. See also, ["About Histories"](#) on page 4-1.


 **Home** This term refers to a default page that appears as soon as you login to a station or application. In Niagara you can use navigation files ("nav" files) to create a unique navigation tree, including a home page, for each user or user-type. For related details, see the *NiagaraAX Graphics Guide* section "About the Nav file".

 **host** This is a term for a hardware system (or platform) that provides the operating environment for Niagara application. In a navigation tree, the host node is used to depict the platform, which is the first level of the navigation tree. Hosts always represent a physical piece of hardware. Localhost is a term used to indicate the local machine.

 **HTTP** HTTP is short for HyperText Transfer Protocol, the underlying protocol used by the World Wide Web (World Wide Web). HTTP defines how messages are formatted and transmitted, and what actions Web Servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web Server directing it to fetch and transmit the requested Web Page. The other main standard that controls how the World Wide Web works is HTML, which covers how Web Pages are formatted and displayed.


HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement World Wide Web sites that react intelligently to user input. This shortcoming of HTTP is addressed in a number of technologies, including Java, JavaScript and cookies.

icon An icon is a graphical representation. Icons are used in the workbench toolbar and nav tree.


 **image** Image is a representation of a raster image.

import Import refers to receiving data from an external source. The Niagara Framework imports its Database from XML for backup and conversion.


ip scheme The "ip" Scheme is used to identify a BIPHost instance. Ords starting with "ip" are always absolute and ignore any base which may be specified. The body of a "ip" query is a DNS hostname or an IP address of the format "dd.dd.dd.dd".

 **Java** Java is a high-level programming language developed by Sun Microsystems. Java source code files (files with a .java extension) are compiled into a format called bytecode (files with a .class extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs), exist for most operating systems, including UNIX, the Macintosh OS, and Microsoft Windows. Bytecode can also be converted directly into machine language instructions by a just-in-time compiler (JIT).


Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java Applets and can be downloaded from a Web Server and run on your computer by a Java-compatible World Wide Web browser. For more information, see <http://java.com> and <http://java.sun.com/>.

 **JavaScript** JavaScript refers to the standard scripting language used in HTML documents. Microsoft Internet Explorer support is discussed at <http://msdn.microsoft.com/scripting/jscript/default.htm>. Mozilla support is discussed at <http://www.mozilla.org/js/index.html>. See <http://www.ecma-international.org/publications/standards/Ecma-262.htm> for more information on the official Standard ECMA-262 ECMAScript Language Specification.

language code Language Code refers to ISO 639 language code -- two letter, lower-case preferred. See <http://www.loc.gov/standards/iso639-2/langcodes.html> for the list.

 **learn** Learn allows you to discover and add items that are defined using the driver framework. Refer to the *Drivers Guide* sections "About Device Discover, Add, and Match (Learn Process)" and "About Point Discover, Add, and Match (Learn Process)" for related information.

lexicon A lexicon is used to provide alternative text for the workbench interface. For example, the text in menus, actions, and other things can be changed by using a custom lexicon entry. Lexicons also provides support for non-English languages. See also, "[About lexicons](#)" on page 1-20 and "Notes on English (en) lexicon usage" in the *NiagaraAX Lexicon Guide*.


 **links** Links are the relationships defined between components. Links provide the means to pass data between components. See "Links" in the *Developer Guide* for more information.

local Local is a scheme used in Ords. It is both a host scheme and a session scheme. It represents Objects found within the local VM. See also, "[About schemes](#)" on page 1-17.


log Log is used to log events. 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.

logoff Logoff refers to removing a user from an active session so that a password will have to be entered to logon for use of the system.

logon Logon is the act of entering a valid address, user name and password. This creates an active session for use of the system.

 **match** Match is a feature provided in manager views that provide a "Discover" function. After discovery, the match feature makes it easier to properly configure and add objects (for example, points) to a "discovered" network.

MIME The Multipurpose Internet Mail Extensions (MIME) provide a mechanism to determine file type in a standardized way.


 **module** The first step in understanding the Niagara architecture is to grasp the concept of modules. Modules are the smallest unit of deployment and versioning in the Niagara architecture. A module is:

- A JAR file compliant with PKZIP compression
- Contains a XML manifest in meta-inf/module.xml

- Is independently versioned and deployable
- States its dependencies on other modules and their versions


See “[About modules](#)” on page 1-6, or modules in the *Developer Guide* for more information.


module scheme 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. See also, “[About schemes](#)” on page 1-17.

 **monitor** Monitor refers to observing. The Ping Monitor periodically calls ping on all the “pingable” objects to monitor network and device health. PingMonitor provides built-in support to generate alarms when pingables are down. Refer to the *Drivers Guide* section “About Monitor” for more details.

Mozilla Mozilla is a browser used to access Web Pages from a Web Server. See <http://www.mozilla.org/index.html> for more information.

name map Name Map defines a new Display Name for a Language Code. You can [Add Name Map](#) to provide a new Display Name for a Language Code. A blank Language Code is used for all languages. You Right-click on the Property displayName to delete or rename Name Map and displayName_xx to delete or rename Name Map (xx) where xx is the Language Code. Bajadoc is available at BNameMap.bajadoc.


 **nav file** Nav File stores XML nav markup. It allows you to define a navigation tree. You may use the [NavFileEditor](#) to modify this file. Bajadoc is available at BNavFile.bajadoc. For more details, see the *NiagaraAX Graphics Guide* section “About the Nav file”.


 **network** Network is a collection of devices. In WorkbenchAX, the Device Manager view provides a summary of an attached network. See “About Network architecture” in the *Drivers Guide* for more information.

Niagara Framework The Niagara Framework is a system designed to manage and control information. Its primary application is for control systems because of its powerful and flexible integration capabilities. The system is made up of Stations that run the components of the Niagara Framework and views that provide the ability to view and command these components. See “Baja vs. Niagara” in the *NiagaraAX Developer Guide* for more information.


normalization In NiagaraAX, this term is used to indicate “data normalization”. This refers to the process of making data and features from various different communications protocols work together so that they can be integrated. The Niagara framework provides a way to normalize data across various protocols so that it may be viewed and controlled from a single user interface.







null The null flag indicates that the associated value should not be used. It represents a “don’t care” condition. It is used in combination with priority arrays so that points may take or release control of their level in a priority array based on their current state. It is also used in math, logic and other application blocks that take a variable number of inputs. Any input with the null flag set is ignored.

 **NumericPoint** NumericPoint defines a read only control point with only an output element. Bajadoc is available at BNumericPoint.bajadoc.

 **object** An Object is the base class required for all objects which conform to the Baja model. See BObject.bajadoc for more technical information. See Object Model in the Developer Guide for more information on the model.


offline Offline refers to accessing a Bog File file directly when the Station is not running.

open  Open allows you to open any of the following:

-  • Ctrl + L
-  • Ctrl + O
-  •
-  •
-  • Ctrl + Shift + O
-  • Open Daemon (Niagarad)...

See the File menu reference for more information.

operator This term indicates one of two levels of permissions (admin and operator) available in NiagaraAX. Admin and Operator levels are further specified by the terms read, write, and invoke. So you may have permission-levels of: operatorRead, operatorWrite adminRead, adminWrite, and so on.


 **options** Options allow you to customize the workbench for the way you use it. It can be selected from the main Menu by selecting **Tools > Options**. See [“Types of Workbench options”](#) on page 2-25 for more information.

ORD Ord is an "Object Resolution Descriptor". An ORD is Baja's universal identification system for integrating heterogeneous naming systems into a single string. An ORD is composed of one or more queries. Each query has a Scheme which identifies how to parse and resolve the query into an Object.

```
ord := query (ws "|" ws query)* query := scheme ws ":" ws body scheme := alpha
(alpha | digit)* body := bodyChar (bodyChar)* alpha := "a"-"z" | "A"-"Z" digit
:= "0"-"9" bodyChar := 0x32 - 0x127 except "|" ws := (space)*
```


Also see “Naming” in the *NiagaraAX Developer Guide*, and [“About ORDs”](#) on page 1-16.


overridden Overridden indicates that the user has manually set the value of a component.

 **palette** Palettes allow you to save copies of your work for future use and are commonly associated with modules. Palette files may reside under a station file system or as part of a module. Also see [“About palettes”](#) on page 1-12 and [“To create a palette”](#) on page 9-11.


pathbar In a workbench view, the part of the locator bar that shows your current path (not including the view selector) is referred to as the pathbar. The pathbar automatically updates each time a new view is selected to display the Ord of each view. It also allows you to select part of the Ord, by clicking on it, to select a destination. See [“About the locator bar”](#) on page 2-14 for more information.


permissions Permissions define what rights a user has within each of the categories in a station. There are two Niagara permission levels: operator and admin. Within both levels, separate options exist for read access, write access, and invoke (action) access. See also, [“Permissions and security”](#) on page 6-8.


 **pin slots** This function provides the capability to make Properties of a component visible in the Wiresheet view of the component. See also, [“Pin Slots”](#) on page 11-15.

 **platform** Platform is the Niagara Platform Definition. See also, *NiagaraAX Platform Guide*.

plugin A This is the term for a visualization (or view) for a component. A WbPlugin is a widget designed to provide plugin functionality in the Workbench tool environment.

 **point extension** Point extensions allow control point behavior to be extended in a consistent manner. Each property of a ControlPoint that subclasses from a PointExtension is considered an extension on the point. Extensions allow plug-in functionality such as alarming and historical data collection via special hooks that a ControlPoint provides to PointExtension.


 **print** Print provides automatic generation of a PDF for a view. Most standard widgets including labels, point widget, tables, Charts, and most layout panes are handled. Access to the PDF visualization is through the Print command in both Workbench and browser views.

 **profile** Profiles provide NiagaraAX software engineers with the ability to customize both the desktop Workbench and the Web Workbench interface. In addition, the type of profile used to login to the station can result in certain views being inaccessible even if the user has permissions to access the view. For more details, see the *NiagaraAX Graphics Guide* section “About profiles”.

properties Properties are the visible data associated with a component. They provide the primary means of interacting with components. Most components properties are visible in the component property sheet view.


proxy Proxy is used locally to represent a remote object. See [“About the proxy extension”](#) on page 3-13 for more information.


ProxyExt ProxyExt is the point extension supported in ControlPoints which are proxies for point data in an external system. See [“About the proxy extension”](#) on page 3-13.

 **Px file** A Px file contains Baja components. It can be a complete Database or any Collection of components. A Px File is a special file that can provide a visualization for components in a Database. The PxEditor is a view used to create and modify Px files. All views can be used on components in a Px just as if they were in a Station. For details, see the *NiagaraAX Graphics Guide* section “About Px files”.


query Query is part of an "Object Resolution Descriptor". An ORD is composed of one or more queries. Each query has a scheme which identifies how to parse and resolve the query into an object. A query is made up of a scheme and a body. Also see, “Naming” in the *NiagaraAX Developer Guide* [“About ORDs”](#) on page 1-16.


range For an enum type component, this is a range of ordinal/name pairs. Also, see [Enum Range Dialog](#) or “[Effect of facets on point actions](#)” on page 3-9.


 **redo** This toolbar button or menu item causes an Action to be redone. It is only available after an undo. The shortcut is Ctrl + Alt + Z (hold down Ctrl and Alt and press Z).


 **refresh** This toolbar button or menu item refreshes a page after its source has been edited, or if you encounter a reason to reload the view.

registry The registry is a term for a small Database built by Niagara Framework whenever it detects that a module has been added, changed, or removed. The registry may be used to interrogate the system for summary information about the modules, types, Ord Schemes, and Agents installed without loading modules into memory.

 **rename** Rename allows you to change the name of a component. It is available on a component's popup menu.

 **reorder** This toolbar and popup menu command is used to open Reorder dialog box when viewing a component's slot sheet. A reorder command is also available in the Point Manager view and in the Px Editor.

 **replace** This search function feature allows you to replace the next occurrence in the currently opened file.

 **replace files** Replace Files allows you to replace the all occurrences in the files.

restart This command restarts the selected station.

scheme Scheme is part of an "Object Resolution Descriptor". The scheme identifies how to parse and resolve the query into an Object. The scheme "local" is both a Host scheme and a session scheme. It represents objects found within the local VM. Some common schemes include:

- ip Scheme
- fox Scheme
- file Scheme
- module Scheme
- station Scheme
- slot Scheme
- handle Scheme
- spy Scheme
- bql Scheme


See “Naming” in the Developer Guide and “[About ORDs](#)” on page 1-16.

security Security in the Niagara framework covers some broad topics:

- Authentication: Logging in and verifying a user
- Encryption: When and how to use cryptography to secure data
- Permissions: Configuring and verifying user Permissions on objects
- Auditing: Logging user actions to create an audit trail


The heart of the Niagara security architecture is embodied by the [UserService](#). The UserService contains a Table of SecurityDomains (see [Security model overview](#)), Profiles (see [Security model overview](#)), and [Users](#). Domains are used to build groups of objects with common security needs. Profiles are used to define [Permissions](#) across zero or more Domains. Users are assigned to a Profile and used to represent an entity (human or machine) requiring authentication and security checks. See Security in the Developer Guide for more information.

SELECT This is the primary statement used to access information in a Database. See [workbench:ProjectionBuilder](#) for more information.

 **selection** The selection is the set of currently selected items. Most operations in the system act on the selected items.

servlet A Java program that runs as part of a network service, typically an HTTP Web Server and responds to requests from clients. The most common use for a servlet is to extend a Web Server by generating World Wide Web content dynamically. For example, a client may need information from a database; a servlet can be written that receives the request, gets and processes the data as needed by the client and then returns the result to the client. Applets are also written in Java but run inside the JVM of an HTML browser on the client. Servlets and Applets allow the server and client to be extended in a mod-

ular way by dynamically loading code which communicates with the main program via a standard programming interface. See also `HttpServlet.bajadoc` for more technical information.

 **sidebar** Sidebar allows you to choose whether to view sidebars by selecting **Window > Side Bars > Show Side Bar** from the main menu. You can also select which sidebars you would like to view. See [“About side bars”](#) on page 2-3 for more information.

slot Slots are the building blocks for defining Niagara components. There are three types of slots:

- Property
- Action
- Topic

Also see [“About slots”](#) on page 1-9.


slot scheme The "slot" Scheme is used to resolve a Value within a Complex by walking down a path of slot names. Slot queries always parse into a SlotPath. Also see [“About slots”](#) on page 1-9 and [“About ORDs”](#) on page 1-16.

space Space defines a group of Objects which share common strategies for loading, caching, lifecycle, naming, and navigation. Also see [“Types of space”](#) on page 1-18.

spy scheme The "spy" scheme is used to navigate spy pages. The `javax.baja.spy` APIs provide a framework for making diagnostics information easily available. Also see [“Types of schemes”](#) on page 1-17.

SQL SQL is an acronym for Structured Query Language. SQL is an American National Standards Institute (ANSI) standard for accessing Database systems.

stale Stale indicates that a situation has occurred which renders data untrustworthy. How stale is determined is a driver specific issue. However the common case is that a period of time has elapsed without a successful read. `TuningPolicy` provides a `staleTime` property for configuring this period of time.

 **station** Station is the component which represents a station in the Baja framework. See [“About stations”](#) on page 1-15, or “Station” in the *Developer Guide* for more information.

station scheme The "station" Scheme is used to resolve the componentSpace of a station Database. Also see [“Types of schemes”](#) on page 1-17

status Status provides a bit mask for various standardized status flags in the Baja control architecture. Plus it provides for arbitrary extensions using facets. Also see, [“About point status”](#) on page 3-18

statuscolors StatusColors are configurable by the baja lexicon and accessible by the Status API. You may use the **Lexicon Editor** to change these values. Entries include alarm, disabled, fault, down, overridden and stale :

```
Status.alarm.bg=#FFFF0000 Status.alarm.fg=#FFFFFFF
Status.disabled.bg=#FFD88AFF Status.disabled.fg=#FF000000
Status.fault.bg=#FFFFAA26 Status.fault.fg=#FF000000 Status.down.bg=#FFFFFFF0
Status.down.fg=#FF000000 Status.overridden.bg=#FF86BEFF
Status.overridden.fg=#FF000000 Status.stale.bg=#FF00FF00
Status.stale.fg=#FF000000
```

The WireSheet options allow you to choose whether to [Show Status Colors](#). Also see, [“Types of status flags”](#) on page 3-18.

StatusFormat StatusFormat is used to provide a simple pattern language for formatting status values. Patterns are strings with script commands embedded using the % character:

```
"Value=%value% Status=%status%"
```

Examples:

- %value% = value using facets
- %value(precision=2)% = value using 2 decimal places
- %value(units=null)% = value using facets with no units
- %value(units=null|precision=i:2)% = combination of previous two
- %status% = status fully specified
- {%status.flags%} %status.facets% = same as %status%
- %status.flags% = status flags, but not facets
- %status.facets% = status facets, but not flags
- %status.activeLevel% = status active level
- %value% {%status.flags%} @ %status.activeLevel%

The BNF grammar for the pattern language:

```
script := valScript | statusScript valScript := "value" [ "(" valFacets ")" ]
valFacets := valFacet [ "|" valFacet ] * valFacet := valAddFacet | valRemoveFacet
valAddFacet := name "=" value valRemoveFacet := name "=null" value :=
DataUtil.marshal & unmarshal statusScript := "status" [ "." statusSubset ]
statusSubset := statusFlags | statusFacets | statusFacetKey statusFlags :=
"status.flags" statusFacets := "status.facets" statusFacetKey := "status."
name
```

Also see, the Engineering Note document, *BFormat (Baja Format) Property Usage*.

- **StringPoint** StringPoint defines a read only control point with only an output element. Bajadoc is available at BStringPoint.bajadoc. See also, “[Types of control points](#)” on page 3-1.

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

- 📁 **tab** Tab allows you to view multiple views in the same Window. See “[Creating tabs in the view pane](#)” on page 2-21 for more information.
- 📊 **table** Table displays a grid of rows and columns. See also, “[Table controls and options](#)” on page 2-18.
- 🏷️ **tag** An element bounded by the marks "<" and ">" is a tag. Tags are used to mark the semantic or the structure of a document. HTML and XML use tags. A sample is the tag <title> to mark the beginning of a title. See [HTML Tags](#) in the [WbHtmlView](#) for more information on tags supported by the HtmlView plugin.

toolbar A Toolbar is a row of button used to provide quick access to commonly used features. In the Niagara Framework there are two rows of buttons on the toolbar. The first are the common buttons that are always available. The second are the buttons based on the current view, mode and Selection. See also, “[About the toolbar](#)” on page 2-13

- **topic** Topic defines a slot which indicates an event that is fired on a component. Also see “[About slots](#)” on page 1-9
- ↶ **undo** This reverses the last Action as if it had not been performed. In Niagara it is only available for some actions. The shortcut is Ctrl + Z (hold down Ctrl and press Z).
- 📶 **up level** Up Level allows you to go to the next level up. The shortcut is Alt + Up. See “[Types of toolbar icons](#)” on page A-13 for more information.

upload Upload refers to receiving data from a device. In Niagara you may upload a station to its supervisor so that you have a backup of its database.

URI Uniform Resource Identifiers (URIs, aka URLs) are short strings that identify resources in the World Wide Web: documents, images, downloadable files, services, electronic mailboxes, and other resources. They make resources available under a variety of naming schemes and access methods such as HTTP, FTP, and Internet mail addressable in the same simple way.

URI is an extensible concept: there are a number of existing addressing schemes, and more may be incorporated over time.

URL Uniform Resource Locator (URL) is the standard way of addressing over the Internet. The Niagara Framework uses this convention for many of its resource locations.


- 👤 **user** This term represents is a valid name. It is used to logon to the system with a valid password.


view There are many ways to view your system. One way is directly in the tree. In addition, you can right-click on an item and select one of its views from the popup menu. Plugins also provide views of components. See “[About the view selector](#)” on page 2-15 and Viewing components for more information.

Web server A Web Server is a computer that delivers (serves up) Web Pages. Every Web server has an IP address and possibly a domain name. For example, if you enter the URL <http://www.niagaraframework.com/> in your browser, this sends a request to the server whose domain name is niagaraframework.com. The server then fetches the page named index.html and sends it to your browser. Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet. There are many Web server software applications, including public domain software from NCSA and Apache, and commercial packages from Microsoft, Netscape and others.


WHERE The Where clause is used to specify that only certain rows of the Table are displayed, based on the criteria described in that Where clause. See [workbench:QualifierBuilder](#) for more information.

widget Widget is a class used to create light weight user interface components which can seamlessly be used in both the bajaui toolkit and in a 1.1 Applet environment.

 **window** A Window is a graphical area used to present a view of the system. This can refer to the primary window with Locator and tree or user created windows. See [“Creating Additional Windows”](#) on page 2-20 for more information.

 **wizard** Wizards are used to guide the user through common tasks. Some common wizards include:

- New Station Wizard
- Commissioning Wizard
- New Module Wizard

 **Workbench** This is the name for Niagara's Integrated Development Environment (IDE) for non-programmers.

XML eXtended Markup Language (XML) is a standard for tagging data. It is a subset of the Standard Generalized Markup Language (SGML) like HTML. See XML in the Developer Guide for more information.

