

Technical Document

NiagaraAX Mercury Driver Guide

November 6, 2013



NiagaraAX Mercury Driver Guide

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 notice

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

Copyright and patent notice

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

Copyright © 2009-2013 Tridium, Inc. All rights reserved.

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

CONTENTS

- About this guideiii
- Related documents.....iii
- Document change logiii
- Chapter 1 Getting started 1
 - Quick start1
 - Plan system configuration1
 - Mercury requirements.....2
 - Tested versions.....3
 - Mercury driver modules3
 - Install modules in Workbench.....3
 - Run the Commissioning wizard.....3
- Chapter 2 Setting up the station5
 - Add a network5
 - Discover and add Mercury panel(s)6
 - Manually add a Mercury Panel.....7
 - Edit device properties8
 - Discover and add Mercury points.....9
 - Manually add points11
 - About updating point status.....11
 - Set up asynchronous event notification12
 - Manage asynchronous notification for all panels.....13
 - Set up point polling13
- Chapter 3 Configuring alarms and acknowledgments 15
 - About Niagara configuration15
 - Set up alarm and acknowledgment points.....15
- Appendix A Mercury driver troubleshooting..... 17
 - Conditions17
 - Error messages17
- Appendix B Mercury driver reference 19
 - About the Mercury driver19
 - The Mercury panel.....19
 - How the Mercury driver works19
 - Mercury driver terminology.....22
 - Driver dialogs22
 - Mercury driver components.....25
 - MercuryNetwork properties.....25

| | |
|--|-----------|
| MercuryDevice properties | 26 |
| Mercury Communicator properties..... | 26 |
| Device ID properties..... | 28 |
| Mercury points properties..... | 29 |
| Event Listener..... | 31 |
| MercuryDeviceFolder | 32 |
| MercuryPointFolde | 32 |
| Mercury Alarm Extensions | 32 |
| Mercury driver component views (plugins)..... | 34 |
| Mercury Device Manager plugin | 35 |
| Mercury Point Manager plugin | 35 |
| Index | 37 |

PREFACE

About this guide

The NiagaraAX Mercury driver allows you to integrate Mercury devices (panels) and one or more NiagaraAX stations into a complete building automation system.

This guide covers driver installation and NiagaraAX station configuration.

- To begin the installation process, see [Quick start, page 1](#)
- For an introduction to how the NiagaraAX Mercury driver works, see [About the Mercury driver, page 19](#)

Related documents

As part of a NiagaraAX system, this driver shares features and functions in common with other drivers. The common documentation can be found in several locations.

- *NiagaraAX Drivers Guide* (module://docDrivers/doc/index.html)
- *JACE NiagaraAX Install & Startup Guide* (module://docJaceStartup/doc/index.html)
- *NiagaraAX Platform Guide* (module://docPlatform/doc/index.html)

Document change log

Changes to the topics contained in this document

Updates (changes/additions) to this document are listed below.

- Edited images and made style changes, August 2013
- Initial release: May 2013

CHAPTER 1 GETTING STARTED

TOPICS COVERED IN THIS CHAPTER

- **Quick start**
- **Plan system configuration**
- **Mercury requirements**
- **Tested versions**
- **Mercury driver modules**
- **Run the Commissioning wizard**

Quick start

The basic steps to configure a Niagara station for communication with a Mercury panel involves setting up Workbench, commissioning each JACE and configuring input and output proxy points.

- Step 1. Plan the configuration (alarms to communicate to the Mercury panel, door events to report to the Niagara station, and resulting values to set).
- Step 2. Set up virtual points in the Mercury panel(s) to support alarm communication with the Niagara station.
- Step 3. Copy mercury.jar and psia.jar and relevant SSL jars to the Niagara **Modules** folder in Workbench.
- Step 4. Commission each JACE.
- Step 5. Set up SSL security in Workbench, each station and each Mercury panel.
- Step 6. Discover Mercury devices.
- Step 7. Discover portal, input and output points.
- Step 8. Set up alarm and acknowledgement ords.
- Step 9. Set up other components.

Plan system configuration

Both the Niagara station and Mercury panel(s) require configuration. Many options are available. Naming conventions need to be considered and simple system documentation designed.

Panel and station configuration

Implementing the Mercury driver involves coordinating the Mercury panel configuration with the Niagara station configuration of the Mercury driver. Included is any integration with other station control logic. For example, if an input type Mercury proxy point representing an access area transition is set to “true” during an unscheduled (off) time, you may want to override a zone’s air handler (HVAC) and lighting control points to an occupied (on) state during this period.

The way you configure the Mercury driver can affect system access of both:

- a NiagaraAX station user, providing Mercury access control information
- a Mercury access control system user, providing NiagaraAX control system information and related alarms

Best practices

Before you begin, make a list of the configuration you will need, including:

- The number of Mercury panels and portals (this will give you an idea of how long device discovery might take)
- What Niagara device alarms to report to the Mercury security system
- Any other device control and reporting requirements (for example, you could use portal status to control Niagara devices, or even use the NiagaraAX station to connect a Mercury input with a Mercury output)
- Design naming conventions that clearly describe connections and which points are virtual ones.
- Write down the configuration in the form of a diagram.

Mercury requirements

Requirements include the version of Workbench supported, platform compatibility for SSL, and licensing requirements. Each Mercury panel must be configured so that device and point discovery work successfully.

Systems integrator requirements

The procedures in this document assume that you:

- Know how to install and configure the Mercury Panel.
- Are NiagaraAX certified and experienced at configuring stations.

Platform prerequisites

The Mercury driver requires a NiagaraAX platform that supports Hot Spot VM (virtual machine) from Sun Microsystems. It does not support the JACE 2, 4, and 5.

Version of Workbench

The Mercury driver requires Workbench 3.7 or later.

Security requirement

This driver requires either CryptoService or full SSL security. If your network is not yet secure, the form of security you need depends on your version of NiagaraAX and the model number of the JACE you are using. For more information about security, see *NiagaraAX CryptoService (SSL)* (module://docEngNotes/doc/docEn_CryptoService.html) and the *NiagaraAX SSL Connectivity Guide* (module://docSSL/doc/index.html)

.

Licensing requirements

- License to use the NiagaraAX Mercury driver.
- License for CryptoService or full SSL. The same license covers either security feature.

Mercury Panel pre-configuration

- Make sure that the firmware in the Mercury panel is up-to-date before you begin.
- Pre-configuration of a Mercury panel is required to configure virtual points for discovery in the NiagaraAX station.

Tested versions

The Mercury Driver has been tested with the EP1501 Mercury panel, version 1.17.4 (352).

Mercury driver modules

Mercury integration with a NiagaraAX system requires two Niagara modules and one or more SSL modules depending on the platform and NiagaraAX version.

Mercury modules

Two modules are required for the Mercury driver:

- psia.jar
- mercury.jar

Install modules in Workbench

The latest Mercury .jar files may or may not be present in the Workbench **Modules** folder. It is important to work with the latest modules.

Prerequisites:

- A version of Workbench that supports the driver must be installed on the PC or laptop computer. For driver requirements, see [Mercury requirements, page 2](#). If you are installing Workbench for the first time, enable the check box option labeled: **This instance of Workbench will be used as an installation tool**. This option installs the needed distribution files (file extension: .dist) used to commission various models of remote JACE platforms. The distribution files are located under the **Niagara\sw** folder in various revision-named sub-folders. For more information about the distribution files, see *About your software database* (module://docPlatform/doc/aWbSoftwareDatabase.html) in the *Platform Guide* (module://docPlatform/doc/index.html).
- Access to Niagara Central to retrieve the modules .jar files if you do not already have the latest modules.

If Workbench is already installed, use this procedure to manage the driver modules.

- Step 1. Check the Niagara\version\modules folder (where *version* is the version of NiagaraAX you are using).
- Step 2. If needed, download the latest module .jar files from Niagara Central and save them in the Niagara\version\modules folder.

Run the Commissioning wizard

The Commissioning Wizard provides a checklist for performing essential and often one-time platform tasks to set up a new JACE or upgrade the core NiagaraAX software in an existing JACE.

Prerequisites: Before you begin:

- Open the JACE platform.
 - Confirm that you are connected to the internet.
- Step 1. To launch the wizard, right-click the platform in the Nav tree and click **Commissioning Wizard**.
By default all commissioning steps are selected. For a new JACE you should accept all default selections.
 - Step 2. In the **Commissioning** dialog, click to include or omit steps and click **Next**.

Step 3. In the **Licensing** dialog, accept the default and click **Next**.

A minimum of one license is required.

The system downloads the appropriate licenses from the licensing server.

Step 4. To install the module .jar files, accept the default selection on the **Module Content Filter Level** dialog and click **Next**.

Step 5. In the **Station Installation** dialog, if you have a specific database on your PC with stations that are ready to install, select the **Station** from the drop-down list, fill in a **New Name** for the station, select the option for when to start the station, and click **Next**.

Step 6. In the next **Station Installation** dialog, select which station files to copy.

Copy only the “config.bog” station database file copies only the station configuration (components), and not any supporting folders/files, such as px files, html files, and so forth.

Step 7. In the **Software Installation** dialog, click the selection box next to the module name to select additional software modules that are required for your installation and click **Next**.

Before this dialog appears, a popup “Rebuilding software list” briefly appears as the dependencies of the JACE are compared against the available software modules in your Workbench PC’s software database.

NOTE: Do not select modules unless you are sure that they are needed. You can manage software modules later using the Software Manager.

Step 8. In the **Distribution File Installation** dialog, click **Next**.

This dialog lists dependent files for the JACE platform that need to be installed.

Step 9. Make selections on the each of the next three dialogs and click **Next** for each.

- The **TCP/IP Configuration** dialog configures TCP/IP settings.

NOTE: Appropriate DNS and Gateway IP addresses must be set to permit connection with the DRAS server. Otherwise, the AdrDevices will not be able to establish communication with the external server.

- The **Platform Daemon Authentication** dialog defines platform credentials.
- The **Lexicon Installation** dialog selects the language to be used by the JACE.

Step 10. Review the options you selected.

- Click **Back** to make changes.
- Click **Finish** to begin the installation process.

When completed, the wizard reboots the JACE and displays a final dialog. Click **Close** to complete the process.

For more information about the Commissioning Wizard and the options available to configure a JACE, see the *JACE NiagaraAX Install & Startup Guide* (module://docJaceStartup/doc/index.html).

CHAPTER 2 SETTING UP THE STATION

TOPICS COVERED IN THIS CHAPTER

- Add a network
- Discover and add Mercury panel(s)
- Manually add a Mercury Panel
- Edit device properties
- Discover and add Mercury points
- Manually add points
- About updating point status

Add a network

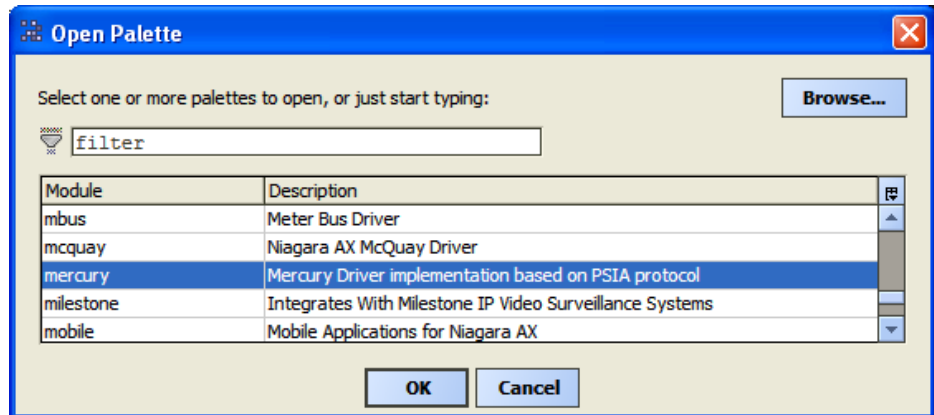
Adding the network is the first step to configure the station.

Prerequisites:

- The driver modules must be available in the Niagara\version\modules folder (where *version* is the version of Workbench).

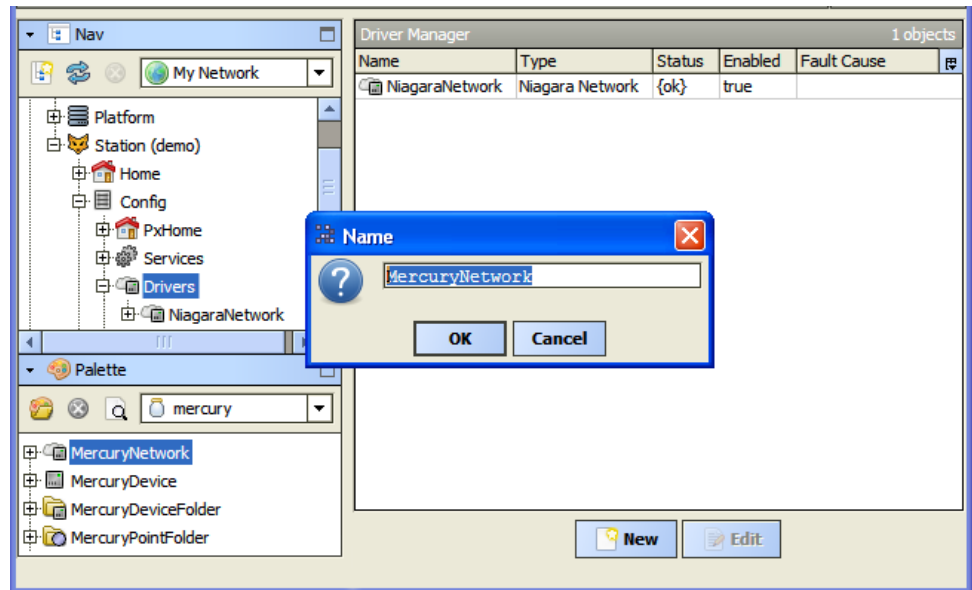
Step 1. In the Nav tree, expand the station and double-click the **Drivers** node.

Step 2. In the **Driver Manager** view, open the driver palette in the side bar.



Step 3. Select the **Module** and click **OK**.

Step 4. Drag and drop or copy a network component from the palette to the **Driver Manager** view pane.



Step 5. Name the network and click **OK**.

Step 6. Confirm that network **Status** is {ok}.

You are ready to configure the station using driver features.

Discover and add Mercury panel(s)

Panel discovery works for any “Hotspot” JACE (JACE-6 or JACE-7 series) or any Windows-based host. For a JACE-2, you must add each Mercury Security panel manually.

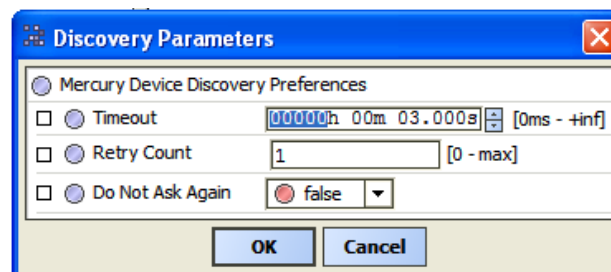
Prerequisites:

- The JACE station must be open.

Device discovery in NiagaraAX is a two-step process. First, you discover device candidates for inclusion in the station database. Next, you drag and drop them from the Device Manager **Discovered** pane to the **Database** pane. This creates device components in the network.

Step 1. Double-click the **MercuryNetwork** node under **Station > Config > Drivers** in the Nav tree.

Step 2. Click **Discover**.

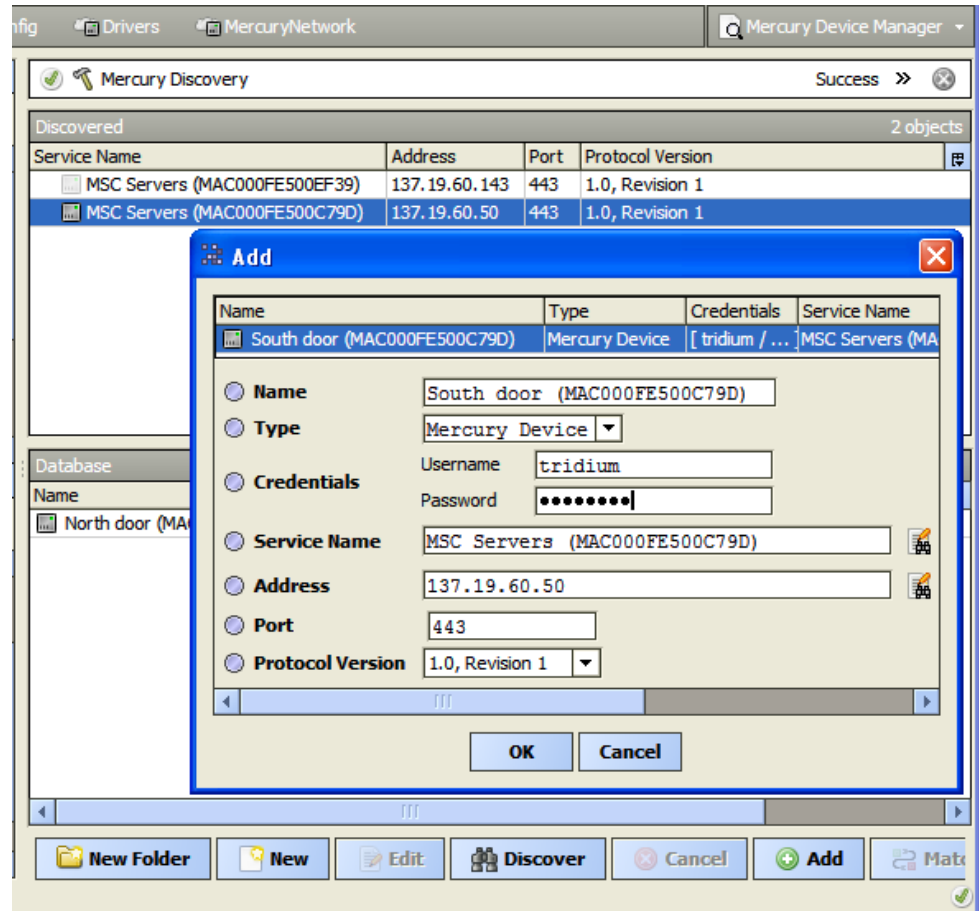


Step 3. In the **Discovery Parameters** dialog, choose your preferences.

NOTE: There’s no progress bar during discovery because there is no way of knowing how many devices are on the network. Set **Timeout** to a long enough time for the system to find all devices.

The device property sheet contains these properties.

During the discovery process, the station sends a DNS request to a well-known multi-cast DNS IP address. This is a broadcast inquiry for all devices that adhere to the PSIA protocol. Each device responds with a DNS response. The JACE collects and displays the responses in a table in the upper pane of the **Mercury Device Manager**.



Step 4. Select the panels in the **Discovered** pane and drag them to the **Database** pane.

The system fills in the default properties for the first device.

Step 5. Change the properties, confirm the values, and click **OK**.

At a minimum you should change **Name** to something representative of the panel location or usage.

The discovered panels are added to the database and appear in the lower **Database** pane of the **Mercury Device Manager** view.

When you add the device to the database, the device remains offline briefly while establishing SSL security.

For more information about the discovery process, see the *NiagaraAX Drivers Guide* (module://docDrivers/doc/index.html).

Manually add a Mercury Panel

Any device may be added manually. With a JACE-2 or the JACE-4/5 series, you must add each Mercury Panel manually. Discovery is not supported.

Prerequisites:

- The station must be running.

Double-click the **MercuryNetwork** node in the **Station > Config > Drivers** node of the Nav tree.

Step 1. In the **Mercury Device Manager**, click **New**.

| Name | Type | Credentials | Service Name | Address | Port | Protocol Version |
|---------------|----------------|-------------|--------------|---------|------|------------------|
| MercuryDevice | Mercury Device | [/ ...] | | | 80 | 1.0, Revision 1 |

☐ **Name**
☐ **Type**
☐ **Credentials** Username
 Password
☐ **Service Name**
☐ **Address**
☒ **Port**
☐ **Protocol Version**

OK Cancel

For a description of each property, see [Add device dialog, page 23](#).

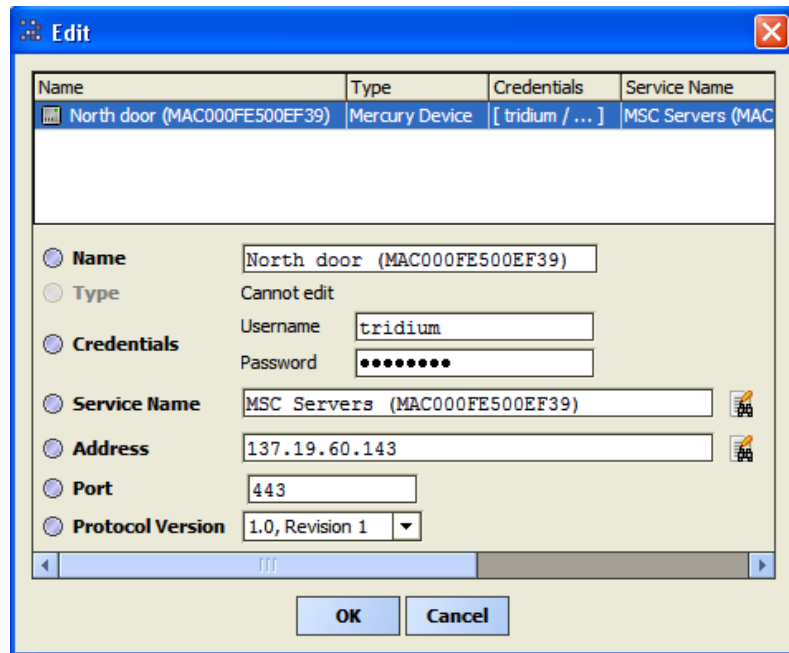
Step 2. Fill in the dialog and click **OK**.

After a brief pause, the system adds the panel to the database.

Edit device properties

After adding the device to the station, either manually or using device discovery, editing properties allows you to reconfigure the device.

Step 1. In the **Database** pane of the **Device Manager**, click the device and click **Edit**, or double-click the device.



For a description of each property, see [Add device dialog, page 23](#).

Step 2. Change the properties as required and click **OK**.

Discover and add Mercury points

Once a Mercury panel is added to the station's network, Mercury proxy point candidates can be discovered and added to the station database. The points you are able to discover depend on how you configured the panel.

Discovery adds Mercury panel input, output and virtual points:

- Inputs

The Mercury driver distinguishes between two types of inputs:

- Standard inputs are read-only BooleanPoints to which readers, sensors and other input devices are connected. Discovery labels these **Inputs**.
- Portal inputs, eight for each door, are attached to door switches and indicate door open, door forced, etc. Discovery labels these **Portals**.

Depending on your configuration, portals may be most significant because they provide the door status information for doors connected to the EP1501 devices.

- Outputs

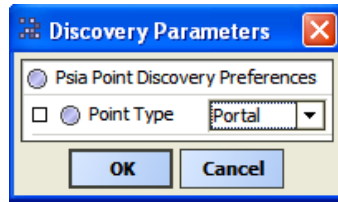
Standard outputs are BooleanWritables that may control output relays and devices, such as a siren, connected to the Mercury panel.

- Virtual inputs/outputs

In addition to standard inputs and outputs, virtual inputs and outputs are not connected to any physical device. They are used to communicate alarms and acknowledgments between the NiagaraAX station and the Mercury panel.

Step 1. Expand the **MercuryNetwork** node and the device node in the Nav tree, and double-click the **Points** extension.

Step 2. Below the **Database** pane, click **Discover**.



Unless you have a special use for input points, this procedure assumes you are most interested in discovering portal and output points.

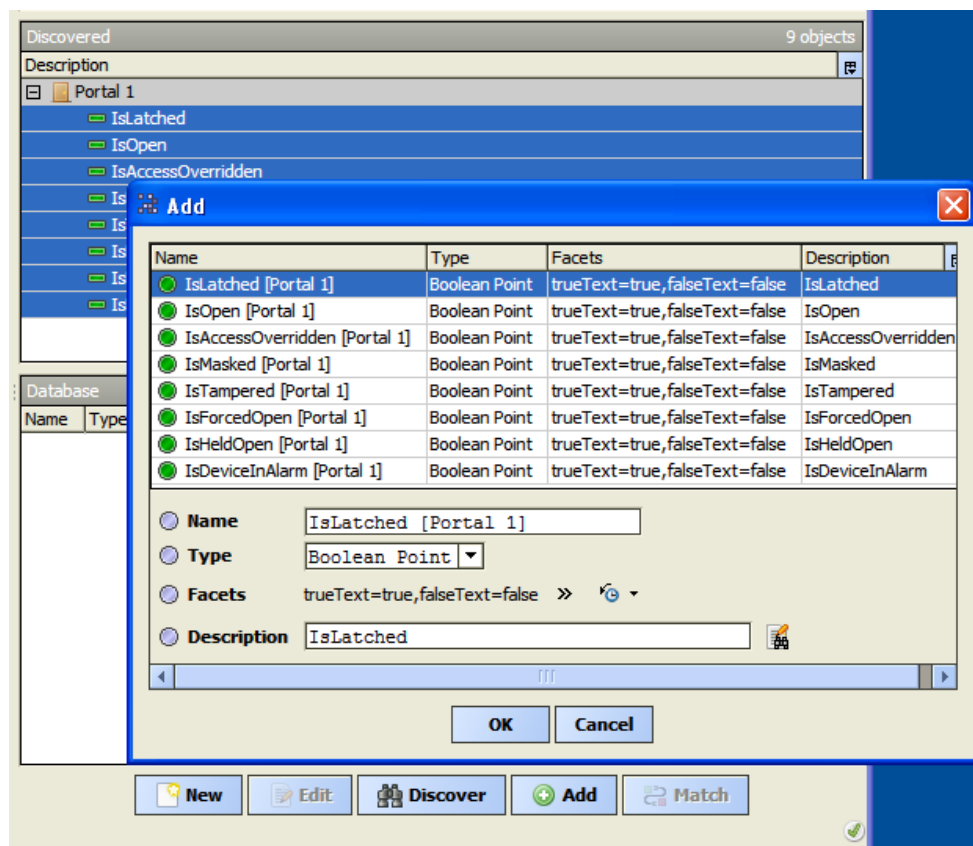
Step 3. To discover the points, use the drop-down list to select type of point and click **OK**.

You will need to discover the three types of points separately. Portals are different from standard input and output points and take a little longer to discover. Be patient.

Step 4. Expand the **Portal** folder, select and drag the discovered points to the **Database** pane.

You cannot drag the folder labeled “Portals” to the **Database** pane. You can only drag the individual points. However, beforehand you could use the “New Folder” feature in the Points Manager, say for each portal, then run the Discovery from that folder.

NOTE: For portals, the discovery process automatically adds “Portal 1,” “Portal 2,” etc. in square brackets to the name to ensure that the name is unique.



Step 5. Change the **Name**, **Facets** and **Description** to make them meaningful and click **OK**.

- When changing the point **Name**, make sure that it is unique within the parent container.
- BooleanPoint is the default **Type** for inputs and portals. Boolean Writeable is the default for output points. Do not change these values.

- Do change **Facets** to something more useful, such as On and Off.
- **Description** can provide additional information, for example, it could be used to explain what “IsLatched” means.

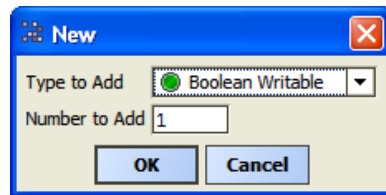
The points appear in the **Database** pane.

Manually add points

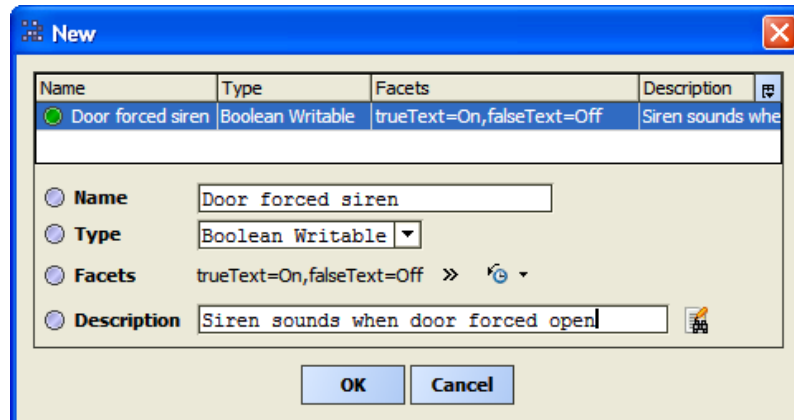
Configurations that do not support discovery (JACE-2 or the JACE-4/5 series) require the manual creation of each proxy point.

Step 1. Double-click the **Points** folder under the **Drivers > MercuryNetwork > Portal** folder in the Nav tree.

Step 2. Click **New**, and select the type of point.




Step 3. To create one or more inputs or portals, select BooleanPoint. To create one or more outputs, select Boolean Writable. Indicate the number of points to add, and click **OK**.



Step 4. Create a unique, meaningful **Name**, change **Facets** to something more appropriate than “true” and “false,” and include a **Description** that explains the purpose of the point.

If creating a virtual point, include the word “virtual” in the name or description.

The  icon accesses the batch search and replace function.

The point is ready to be connected on the wiresheet.

About updating point status

For real-time building automation management, alarm conditions and portal events must be reported on promptly. The Mercury driver updates proxy point values either by polling the points (this method requires significant overhead) or by asynchronous notification.

There are two ways to update the status of Mercury points in the NiagaraAX station:

- The default is point polling, which requires overhead and slows system efficiency.

The controller polls each discovered point regularly. Polling frequency defaults to once per second. Although it can be changed, more frequent polling is recommended to keep statuses current in real time. This type of update uses the HTTPS connection and port 443. This connection is also used for device ping (configured on the MercuryNetwork property sheet).

- The alternative is asynchronous notification, which requires less overhead, improving system efficiency.

The JACE “listens” on a separate connection for asynchronous event notification. When an event occurs in a Mercury panel input, output or portal, a notification is sent to the JACE asynchronously. This notification updates the corresponding point status in the points folder. This type of update uses a TCP socket connection and port 4001.

Set up asynchronous event notification

A special TCP socket connection, using port 4001, allows the JACE to listen for asynchronous event notification from the Mercury panel. This feature of the Mercury driver eliminates the overhead required when the station must poll the panel on a regular basis for point status.

Prerequisites: The Mercury panel must be configured to send event notifications using port 4001.

Step 1. Expand the **MercuryNetwork** node and double-click the device (panel).

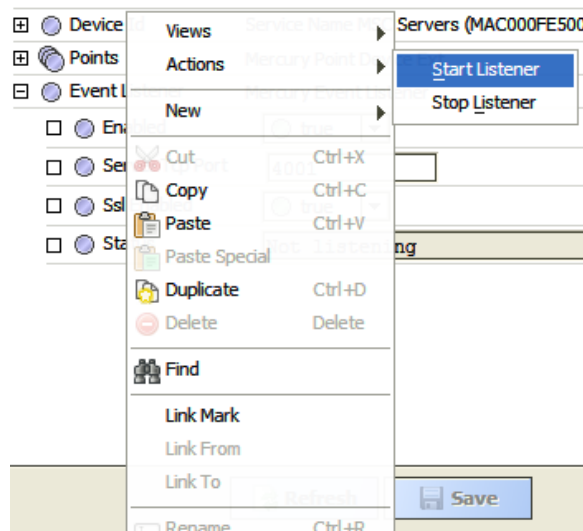
Step 2. Expand or double-click the **Event Listener**.

| Mercury Event Listener | |
|--|---------------------------------------|
| <input type="checkbox"/> Enabled | <input checked="" type="radio"/> true |
| <input type="checkbox"/> Server Tcp Port | 4001 |
| <input type="checkbox"/> Ssl Enabled | <input checked="" type="radio"/> true |
| <input type="checkbox"/> Status | Not listening |

The default for **Enabled** and **Ssl Enabled** is false.

Step 3. Set the values of **Enabled** and **SSL Enabled** to true.

Step 4. To activate event listening, right-click **Event Listener** on the property sheet and click **Actions > Start Listener**



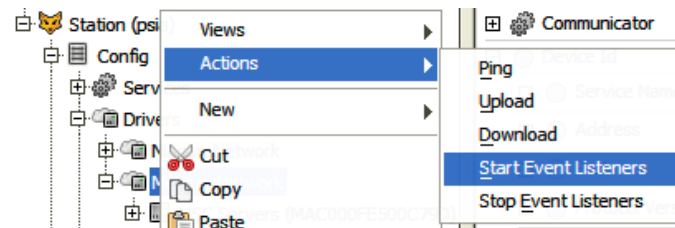
Status changes to **Sending...**, **Establishing...**, **Listening**.

NOTE: Do not invoke **Start Listener** again without invoking **Stop Listener**.

Manage asynchronous notification for all panels

Once asynchronous notification is enabled for each panel, event listening can be activated and deactivated for all panels in one convenient step.

Step 1. Right-click the **MercuryNetwork** node in the Nav tree.



Step 2. Do one of the following:

- To start all Event Listeners, click **Start Event Listeners**.
- To stop Event Listeners, click **Stop Event Listeners**.

While the ability to start and stop event listening for all panels is provided for convenience, it is highly unlikely that once started you will ever need or want to stop event listening.

NOTE: Do not invoke **Start Event Listeners** again without invoking **Stop Event Listeners**.

Set up point polling

Polling proxy points on a regular basis, such as once per second, is one way of refreshing point values. A value is sent to the station each time the device point is polled. Each device has its own polling schedule.

Prerequisites:

Step 1. To go to the property sheet for the device, double-click the device under the **MercuryNetwork** folder in the Nav tree, or right-click the device and click **Views > Property Sheet**.

Step 2. Expand **Communicator** and expand **Poll Scheduler**.

MSC Servers (MAC000FE500C79D) (Mercury Device)

- ☐ Status: {down}
- ☐ Enabled: true
- ☐ Fault Cause:
- ☒ Health: Fail [26-Nov-12 3:55 PM EST] Request Timeout
- ☒ Alarm Source Info: Alarm Source Info
- ☒ Communicator: Mercury Communicator
 - ☒ Transmitter: Mercury Transmitter
 - ☒ Receiver: Ddf Http Receiver
 - ☒ Transaction Manager: Ddf Single Transaction Mgr
 - ☒ Poll Scheduler: Ddf Poll Scheduler
 - ☐ Poll Enabled: true
 - ☐ Fast Rate: 00000h 00m 01.000s [1ms - +inf]
 - ☐ Normal Rate: 00000h 00m 05.000s [1ms - +inf]
 - ☐ Slow Rate: 00000h 00m 01.000s [1ms - +inf]
 - ☐ Statistics Start: 26-Nov-2012 03:53 PM EST
 - ☐ Average Poll: 0.0ms

In the example above, **Enabled** set to `true` turns point polling on. The **Slow Rate**, set to once every second in this example, controls the polling rate.

NOTE: If you are using polling for event notification, keep the poll rate low. Thirty (30) seconds is much too long for polling.

CHAPTER 3 CONFIGURING ALARMS AND ACKNOWLEDGMENTS

TOPICS COVERED IN THIS CHAPTER

- **About Niagara configuration**
- **Set up alarm and acknowledgment points**

About Niagara configuration

The final step to configure a system that combines the two controllers is to link inputs from both sides with their appropriate outputs, and set up how to report and acknowledge alarms.

Once input and output points have been discovered, links can be made between them regardless of the origin or destination of the points. For example, a door forced portal (input point from the Mercury side) could be connected to the output point that sounds a siren on the Mercury side. This is an example of a Mercury panel input driving a Mercury panel output through the station.

More significant perhaps are setting up:

- The control of a device, such as a lighting zone, on the Niagara side based on the portal status as reported from the Mercury side.
- The control of output devices connected to a Mercury panel, such as a siren, based on the status of a Niagara device.

Each of these configurations follow the Niagara standard procedure for connecting inputs to outputs.

A third type of configuration is unique to the Mercury driver. It involves configuring a pair of virtual points to report alarms from a control point in the Niagara station to the console on the Mercury side, and receive the acknowledgment back to the station. These points are not associated with physical I/O points in the Mercury panel or device proxy points in the station, however, there is nothing stopping you from using a physical point as a potential alarm point.

Set up alarm and acknowledgment points

Alarm handling involves configuring the Niagara station to send alarm status to the Mercury ACS console where the alarm can be acknowledged. This makes it possible to consolidate building monitoring and access security in one location.

Prerequisites:

Alarm configuration requires two virtual points added during Mercury panel point discovery:

- A BooleanWritable that must be linked with a source control point, such as a proxy point for a boiler on the Niagara side. When the source point goes into alarm, it sets this virtual point to “true,” which causes the alarm to appear on the Mercury console.
- A second virtual point that may be a BooleanWriteable (output) or a BooleanPoint (input). This point is used by the Mercury console to turn off the output to the Mercury panel and acknowledge the alarm in the NiagaraAX station.

This procedure links this pair of virtual points to the standard Niagara device points that issue and acknowledge an alarm. For the sake of illustration, the procedure assumes you want to configure a boiler alarm to appear on the monitor and be acknowledged by the security guard who is responsible for the Mercury access control system. During discovery, the virtual points were named as follows:

- Virtual boiler alarm
- Virtual boiler alarm acknowledgment

Step 1. Expand the **Config** node in the Nav tree and locate the proxy point associated with the Niagara device.

In the example, this would be the proxy point associated with boiler status.

Step 2. To view the point's property sheet, double-click the point or right-click the point and select **Views > Property Sheet**.

Step 3. Add a Mercury driver point extension by expanding the **Extensions** folder in the Mercury palette, and dragging and dropping the appropriate alarm extension onto the property sheet.

The Mercury alarm extensions are functionally the same as the standard alarm extensions with the addition of two properties, **Alarm Point Ord** and **Ack Point Ord**.

Step 4. Expand the alarm extension property sheet and scroll down to the bottom of the properties list.

Step 5. Scroll to the right, and click the browser icon (📁) next to **Alarm Point Ord**.

Step 6. Locate the appropriate virtual BooleanWriteable output point you discovered and added for the purpose of communicating the alarm from the NiagaraAX station to the Mercury panel.

Step 7. Associate the **Ack Point Ord** with the appropriate virtual input point you discovered and added for the purpose of deactivating and acknowledging the alarm.

| | |
|--|---|
| <input type="checkbox"/> Alarm Point Ord | <code>\$20Servers\$20\$28MAC000FE500C79D\$29/points/VirtualBoilerAlarm</code> |
| <input type="checkbox"/> Ack Point Ord | <code>28MAC000FE500C79D\$29/points/VirtualBoilerAlarmAcknowledgment</code> |

The example above shows the **Alarm Point Ord** and **Ack Point Ord** after scrolling to the right (to the end of the ord) to see the names of the points.

Behind the scenes, the Mercury driver configures the wiresheet to connect the points. When the source Niagara control point goes into alarm in the station, the output point reports the alarm to the Mercury side.

APPENDIX A MERCURY DRIVER TROUBLESHOOTING

TOPICS COVERED IN THIS APPENDIX

- **Conditions**
- **Error messages**

Conditions

Errors may indicate missing configuration links, virtual points, or credentials. Reviewing your plan and confirming all configuration steps should help resolve errors.

- **I added the Mercury Panel manually, but cannot connect to it.**

The station must have the correct credentials (username and password) to connect to the panel. In the **Database** pane of the **Device Manager**, double-click the device and define the expected credentials.

- **An acknowledgement does not propagate.**

There is no acknowledgment link. See [Set up alarm and acknowledgment points, page 15](#)

Error messages

Error messages provide starting points for resolving error conditions.

- **Ddf Auto Discovery Failed**

Make sure that the points of the type you selected have been discovered and added or created in the Mercury Panel and try again.

APPENDIX B MERCURY DRIVER REFERENCE

TOPICS COVERED IN THIS APPENDIX

- **About the Mercury driver**
- **Driver dialogs**
- **Mercury driver components**
- **MercuryDeviceFolder**
- **MercuryPointFolder**
- **Mercury Alarm Extensions**
- **Mercury driver component views (plugins)**

About the Mercury driver

The Mercury driver manages communication between a NiagaraAX Building Automation System (BAS) station and a Mercury Security Access Control System (ACS) panel.

The Mercury driver integrates one or more IP-based EP1501 Mercury Panels into a NiagaraAX station. Data items in the Mercury access control system and the station are exchanged via Mercury proxy points under Mercury devices (panels), using SSL encrypted connections. Niagara station-sourced alarms can be configured for monitoring and acknowledgement by the Mercury system.

The driver uses a familiar NiagaraAX driver hierarchy of a single parent network, one or more devices, where each device has proxy points. One atypical difference is that each Mercury device (panel) has its own poll scheduler, which may be used by the Mercury proxy points for that panel. Device polling is the default method for updating proxy point status, but asynchronous event notification using the driver's Event Listener is also possible.

The Mercury panel

The Mercury panel controls building access by monitoring door entries and exits.

The Mercury Security Corporation panel for which the Mercury driver was created is the EP1501. This controller provides door control and networking capability (Ethernet) as a component in a complete access control system.

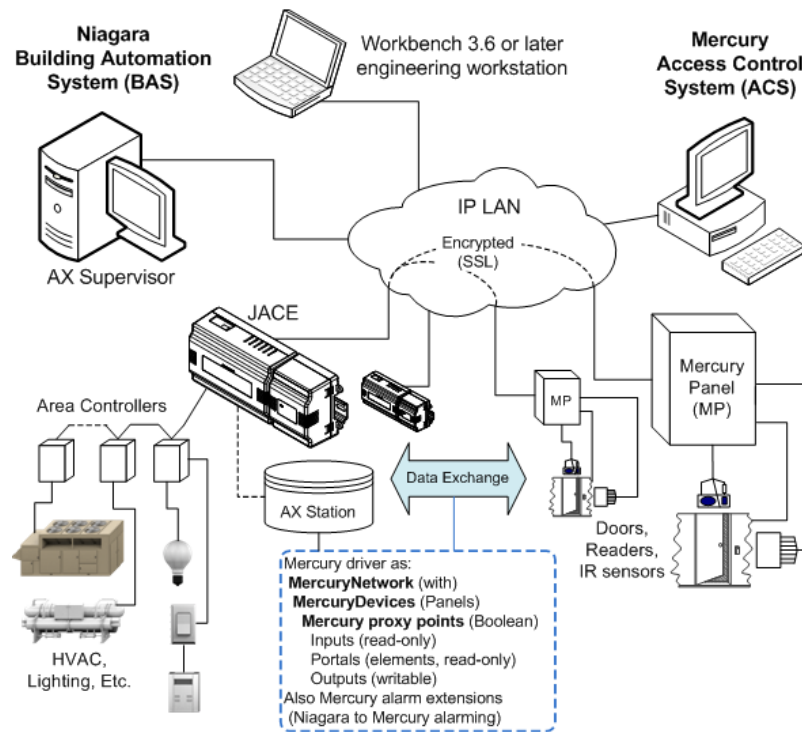
A third-party access program called Open Options manages the Mercury panels in the access control system. This program communicates with NiagaraAX through a Mercury panel. The two software programs do not communicate directly.

On the Mercury side, a portal is a door that is controlled by a Mercury EP1501 panel. With the Niagara Mercury driver, the access information monitored by each Mercury panel can be configured to manage building devices, such as turn lights on and off, turn the HVAC on and off, etc.

How the Mercury driver works

The Mercury driver uses a subset of the PSIA protocol over SSL encrypted IP connections to communicate to Mercury panels. The NiagaraAX station monitors data points in each Mercury panel, as defined by discovered or added Mercury proxy points. A mechanism for exposing Niagara alarms to Mercury Panels is available.

About Mercury and Niagara data exchange



The NiagaraAX Station functions as a client and the Mercury panel as the server. NiagaraAX sends a message and the panel replies. You can modify Boolean statuses. Relays light up when BooleanWriteables are sent to a panel.

The Mercury driver is especially suited to monitoring security and air handling. Virtual outputs in the panel receive alarms from the station. The acknowledgment is sent back from the panel to the station.

About Mercury Panel discovery

Each Mercury panel is represented as a Mercury device in the station's Mercury network. The network provides device discovery when adding panels for most recent platforms such as the **JACE-6** and **JACE-7** series, or any Windows-based platform. If your network includes a **JACE-2** or other "non-Hotspot VM" platform, Mercury Devices must be added manually. User credentials for a panel are required whenever adding a Mercury Device to the network.

About Mercury proxy points

All added Mercury proxy points are Boolean types, as either BooleanPoints (for inputs and portal elements) or BooleanWritables (for outputs).

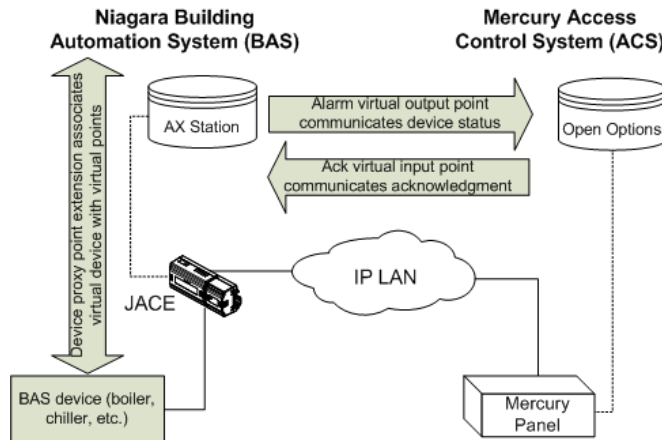
Once the Mercury's portal points and output points are added to the NiagaraAX station, any number of configurations are possible. For example, you could:

- Use the portal points to control an output device in the ACS, such as a siren.
- Use the virtual points to communicate the status of the BAS proxy point to the ACS for monitoring by the security guard.

In a NiagaraAX station, a portal is a container (folder) for the status of each door. Status values include IsLatched, IsOpen, IsTampered, etc.

How virtual points work

Virtual points are points that are added to the Mercury panel that do not correspond to a physical switch or relay. Two virtual points associated with each device proxy point in the station are used to exchange Boolean (alarm status and acknowledgment) with the Mercury panel.



A physical device, such as a boiler, uses a single proxy point to report its status to a Niagara station. To pass this alarm information on to a Mercury panel, and to receive an acknowledgment back from the panel, two virtual Boolean points must be set up in each Mercury panel and discovered by the station.

- The Mercury Panel gains awareness of what is happening in the Niagara side using a virtual *output* proxy point. This point communicates alarm status from the device (boiler, HVAC system, etc.) in the Niagara system to the Mercury ACS.
- Likewise, the panel acknowledges an alarm using a virtual *input* proxy point, which is sent as a Boolean status from the Mercury ACS to the NiagaraAX station.

The configuration process involves connecting the single proxy point that reports status from the source device in the NiagaraAX station to the virtual points that communicate with the Mercury panel.

NOTE: When defining the Name or Description properties for the virtual points in the station, you should use the term “virtual” to indicate clearly which points are being used solely for communication between the BAS and ACS.

About Niagara alarms to Mercury Panels

Providing an alarming function is not part of the PSIA protocol. The alarming portion of the driver has been added to allow third-party vendors to see and acknowledge alarms.

The **mercury** palette provides specialized alarm extensions, by data type, which you can add to any control or proxy point in the station (or any component that accepts regular alarm extensions). The purpose is to export an offnormal state (alarm) to a Mercury Panel, as defined by the Alarm Point Ord or Ack Point Ord property of the extension.

The Alarm Point Ord must reference a BooleanWritable (output) Mercury proxy point in a Mercury device, and the Ack Point Ord must be another Mercury proxy point in the same Mercury device—typically also a BooleanWritable (output) type, but possibly a BooleanPoint (input) type. This pair of points makes a loop.

When an offnormal state occurs in the component with the Mercury alarm extension, the Mercury Panel receives the alarm via the proxy point mechanism. Using the normal access control interface (Open Options) to the Mercury Panel, a user can then acknowledge the alarm, which is passed back to the alarming subsystem of the Niagara station.

Configuration of the Mercury alarm extensions is identical to standard alarm extensions, with the addition of two properties to reference Mercury proxy points. One referenced point is for *sending* the alarm to a Mercury panel, while the other referenced point is for receiving an *acknowledgment* from the Mercury panel.

Mercury driver terminology

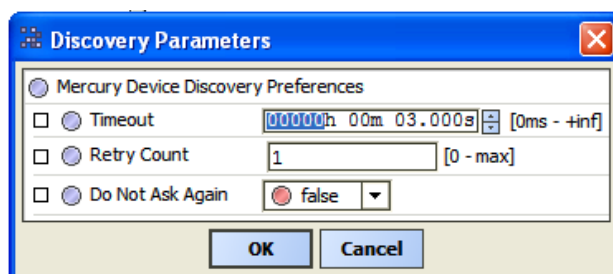
Mercury terms include standard NiagaraAX terminology and terms unique to the Mercury panel and driver.

| Term | Description |
|-----------------|---|
| ACS | Access Control System, the portion of the system managed by the Mercury panel. |
| BAS | Building Automation System. the portion of the system managed by the NiagaraAX station. |
| Event | A happening or occurrence brought about by a device state. Event notification occurs when the device state that sets a property to “true” causes a message to be sent to another system. |
| Mercury device | See Mercury panel. |
| Mercury network | Multiple Mercury EP1501 panels connected via Ethernet and managed from a security computer. |
| Mercury panel | The EP1501 controller board manufactured by Mercury Security Corporation. This Ethernet-ready card reader panel controls a single opening. It is also referred to as a Mercury device. |
| PSIA | Physical Security Interoperability Alliance: a global consortium of physical security manufacturers and systems integrators that provide IP-enabled security devices to industry. The alliance develops and promotes the open industry protocol used by Mercury security products and supported by the NiagaraAX Mercury driver. This XML-based protocol partitions security applications into a set of domains. A portion of the protocol provides access control (how intrusion would work). Another portion provides support for portals (doors). |

Driver dialogs

The discovery, device and point dialogs appear when discovering and adding devices and points.

Device discovery parameters dialog



| Type | Value | Description |
|--|--------------------------------------|--|
| Timeout | Hours (h), minutes(m) and seconds(s) | Depending on the number of panels, discovery could take time. Three (3) seconds is adequate for a single device. Ten (10) to 15 seconds should be enough for several devices. |
| Retry Count | Default = 1 | This property allows you to define how many times the system will attempt to discover panels before timing out. With Mercury devices, if they are not discovered the first time, it is unlikely they will be discovered on a retry. |
| Do Not Ask Again This property allows you to bypass the display of this dialog. | true false | <p>true causes the system to skip this dialog in the future.</p> <p>false causes the system to display this dialog every time you click Discover. This is the default.</p> <p>These discovery preferences can also be changed on the MercuryNetwork property sheet.</p> |



Add device dialog

Add

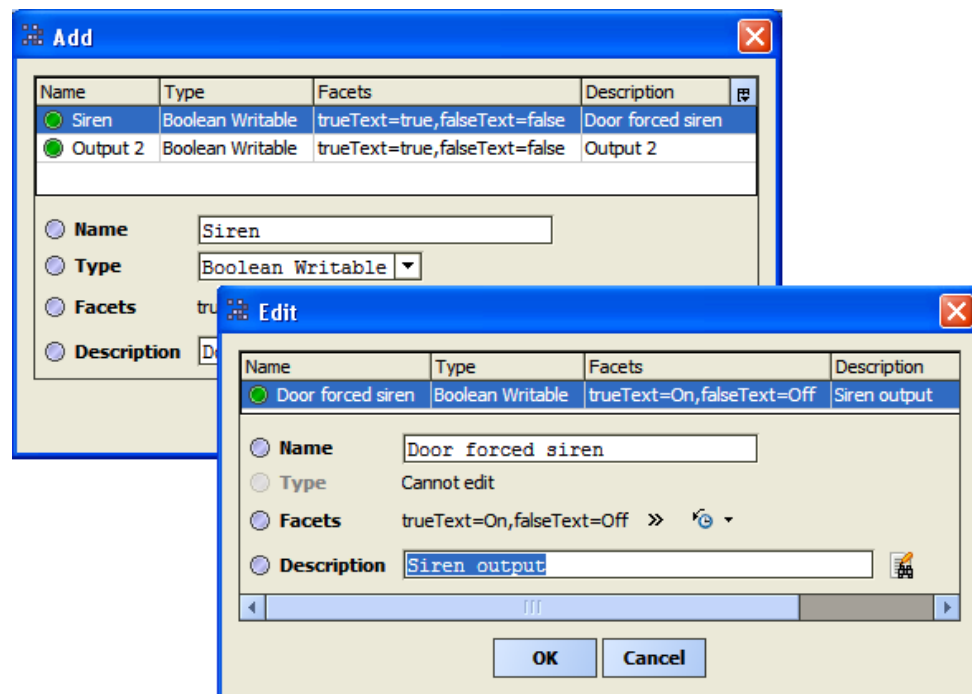
| Name | Type | Credentials | Service Name |
|-------------------------------|----------------|-------------|-------------------------------|
| MSC Servers (MAC000FE500C79D) | Mercury Device | [/ ...] | MSC Servers (MAC000FE500C79D) |

☐ **Name** MSC Servers (MAC000FE500C79D)
☐ **Type** Mercury Device
☐ **Credentials** Username Password
☐ **Service Name** MSC Servers (MAC000FE500C79D)
☐ **Address** 137.19.60.50
☐ **Port** 443
☐ **Protocol Version** 1.0, Revision 1

OK Cancel

| Type | Value | Description |
|--------------------------------------|----------------------|--|
| Name | SI defined | This is the name of the device as it will appear in the station and on reports. Use this name to identify the location of the device or use some other descriptor. |
| Credentials: Username Password | SI defined | These must be the credentials set up in the Mercury panel. You can only indicate what the credentials are. You cannot change them here. If all panel credentials are the same you can batch enter them using the Niagara batch editor. |
| Service Name | SI defined | This name can be anything you choose. It is a good idea to keep the MAC address as part of either this name or the Name above. The  icon provides access to batch search and replace. |
| Address | SI defined | This is the IP address of the panel. The  icon provides access to batch search and replace. |
| Port | Default = 443 | This is the HTTPS socket for the panel that provides SSL security. |
| Protocol Version | For information only | This is the version of the PSIA protocol being used by the panel and does not need to be changed. |

Add/Edit point dialog



| Type | Value | Description |
|-------------|---|--|
| Name | Text | A memorable name the clearly defines the purpose of the point. If this is a virtual point, consider including the word “virtual” in the Name or Description. |
| Type | The type of point (Boolean-Writable or Boolean Point) | Once created, this property cannot be changed. |
| Facets | Set up by SI | This is the standard Niagara facets property. |
| Description | Text | Used to provide additional information about the point. |

Mercury driver components

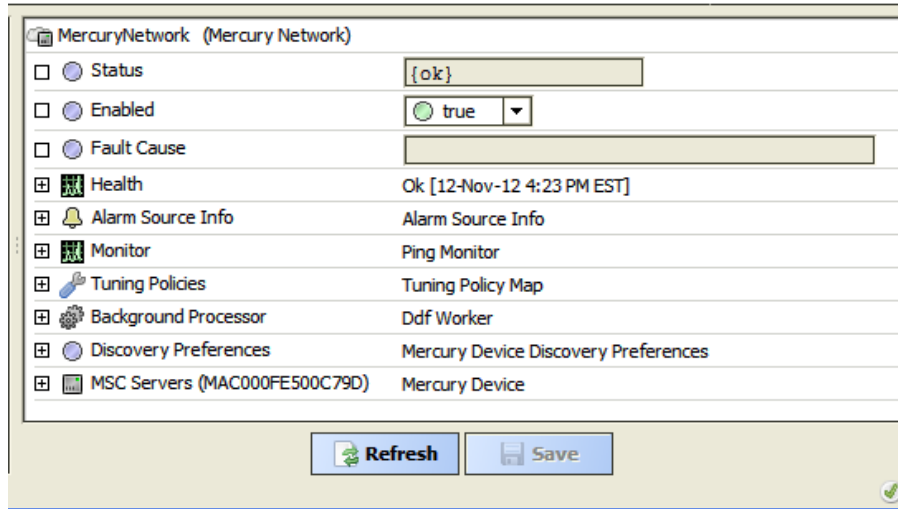
The Mercury driver components are available through the Mercury palette.

The driver components reside under the **Drivers** folder in the Nav tree.

MercuryNetwork properties

The **MercuryNetwork** component is the base container for all Mercury devices and their child data objects (Mercury proxy points).

As with other NiagaraAX driver networks, the **MercuryNetwork** should reside under the station's **Drivers** container. The default view of the **MercuryNetwork** is the **Mercury Device Manager**.

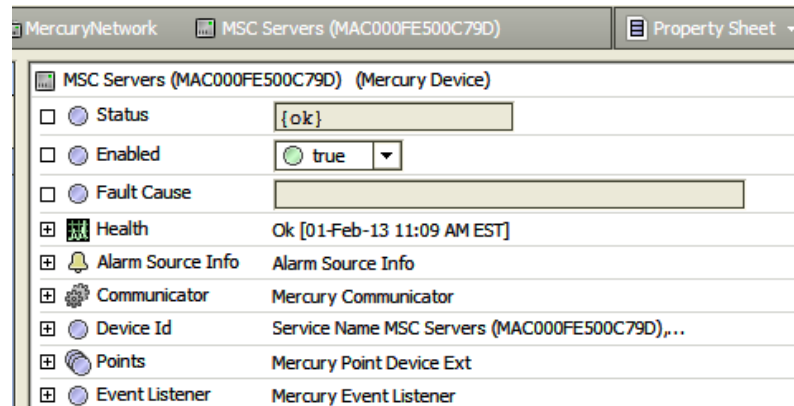


There are no unique properties for the **MercuryNetwork** component. For those that are not self-explanatory, see the *NiagaraAX Drivers Guide* (module://docDrivers/doc/index.html).

MercuryDevice properties

MercuryDevice is a NiagaraAX representation of a Mercury panel.

Each **MercuryDevice** resides under the station's **MercuryNetwork**.



Four sets of properties at the bottom of the property sheet relate directly to the Mercury driver. The other properties are standard NiagaraAX device properties.

- [Mercury Communicator properties, page 26](#) provides statistics regarding instances of communication between the station and panel.
- [MercuryDevice properties, page 26](#) provides access to the device properties.
- [Mercury points properties, page 29](#) provides access to the properties for all discovered and created points associated with each Mercury panel.
- [Event Listener, page 31](#) defines the properties that control asynchronous event notification.

Mercury Communicator properties

Mercury Communicator control communication between the NiagaraAX station and the Mercury panel.

This group of properties manages communication between the station and panel(s).

| | | |
|---|----------------------------|--|
| <input checked="" type="checkbox"/> Communicator | Mercury Communicator | |
| <input checked="" type="checkbox"/> Transmitter | Mercury Transmitter | |
| <input checked="" type="checkbox"/> Receiver | Ddf Http Receiver | |
| <input checked="" type="checkbox"/> Transaction Manager | Ddf Single Transaction Mgr | |
| <input checked="" type="checkbox"/> Poll Scheduler | Mercury Poll Scheduler | |
| <input checked="" type="checkbox"/> Unsolicited Mgr | Ddf Null Unsolicited Mgr | |
| <input type="checkbox"/> Credentials | Username | <input type="text" value="tridium"/> |
| | Password | <input type="password" value="....."/> |

- **Transmitter** keeps track of messages sent from the station to the Mercury panel.
- **Receiver** keeps track of the messages received by the station from the Mercury panel.
- **Transaction Manager** provides no end-user information.
- **Poll Scheduler** is used to configure point polling.
- **Unsolicited Manager** provides no end-user information.
- **Credentials** identify the username and password required by the Mercury panel. They do not refer to the JACE platform or station credentials.

Transmitter

| | | |
|---|-----------------------------------|-----------|
| <input checked="" type="checkbox"/> Transmitter | Mercury Transmitter | |
| <input type="checkbox"/> Transmission Attempts | <input type="text" value="2988"/> | [0 - max] |
| <input type="checkbox"/> Transmission Count | <input type="text" value="2988"/> | [0 - max] |
| <input type="checkbox"/> Retransmission Count | <input type="text" value="0"/> | [0 - max] |
| <input type="checkbox"/> Max Retry Count | <input type="text" value="0"/> | [0 - max] |

| Type | Value | Description |
|-----------------------|-------------------|---|
| Transmission Attempts | | Reports the number of attempted transmissions from the panel to the station. |
| Transmission Count | | Reports the number of successful transmissions from the panel to the station. |
| Retransmission Count | | Reports the number of retries attempted when a transmission attempt failed. |
| Max Retry Count | number of retries | Lets you set the number of times an attempt to transmit will be made before timing out. |

Receiver

| | | |
|--|---|--------------|
| <input checked="" type="checkbox"/> Receiver | Ddf Http Receiver | |
| <input type="checkbox"/> Response Timeout | <input type="text" value="00000h 00m 10.000s"/> | [0ms - +inf] |
| <input type="checkbox"/> Num Frames Received | <input type="text" value="2995"/> | [0 - max] |

| Type | Value | Description |
|------------------------|--|---------------------------------------|
| Response Timeout | hours minutes and seconds (hundredths of seconds) | |
| Number Frames Received | | Reports communication with the panel. |

Poll Scheduler

☐ Poll Scheduler

Mercury Poll Scheduler

☐ Poll Enabled
 ☐ false

☐ Fast Rate
 00000h 00m 01.000s [1ms - +inf]

☐ Normal Rate
 00000h 00m 01.000s [1ms - +inf]

☐ Slow Rate
 00000h 00m 05.000s [1ms - +inf]

☐ Statistics Start
 30-Jan-2013 11:52 AM EST

☐ Average Poll
 0.0ms

☐ Busy Time
 -

☐ Total Polls
 0 over 0ms

☐ Dibs Polls
 -% (0/0)

☐ Fast Polls
 -% (0/0)

☐ Normal Polls
 -% (0/0)

☐ Slow Polls
 -% (0/0)

☐ Dibs Count
 current=0 average=0

☐ Fast Count
 current=0 average=0

☐ Normal Count
 current=0 average=0

☐ Slow Count
 current=0 average=0

☐ Fast Cycle Time
 average = 5261ms

☐ Normal Cycle Time
 average = 5261ms

☐ Slow Cycle Time
 average = 5261ms

For more information, see *Poll Components* (<module://docDrivers/doc/aPollComponents.html>)

| Type | Value | Description |
|----------------------------|----------------------------|---|
| Poll Enabled | true false | Turns proxy point polling on and off. Polling is one way to monitor point status. |
| Fast, Normal and Slow Rate | hours, minutes and seconds | Lets you define the speed for each rate. |
| Polling statistics | | The remainder of the properties provide statistics on polling performance. For more information about these statistics, see the <i>NiagaraAX Drivers Guide</i> (module://docDrivers/doc/index.html). |

Device ID properties

This set of properties define each panel.

Device Id properties

The screenshot shows a configuration window titled 'Device Id'. It contains several fields with checkboxes: 'Service Name' (checked) with value 'MSC Servers (MAC000FE500C79D)', 'Address' (checked) with value '137.19.60.57', 'Port' (checked) with value '443', and 'Protocol Version' (checked) with a dropdown menu showing '1.0, Revision 1'.

The following are properties unique to the Mercury driver:

| Type | Value | Description |
|------------------|--------------------|---|
| Service Name | Alpha-numeric name | A name for the panel. |
| Address | | The IP address of the device. |
| Port | Defaults to 443 | The communications port with the device. |
| Protocol Version | | The version of the PSIA protocol supported. |

Mercury points properties

This extension is the container for all proxy points that represent real-time data originating from the Mercury panel(s). This includes virtual points.

Types of points

Two types of points are supported:

- Portal points represent “real” proxy points that are associated with devices.
- Virtual points represent “logical” points used to communicate with the Mercury panel(s).

Portal points

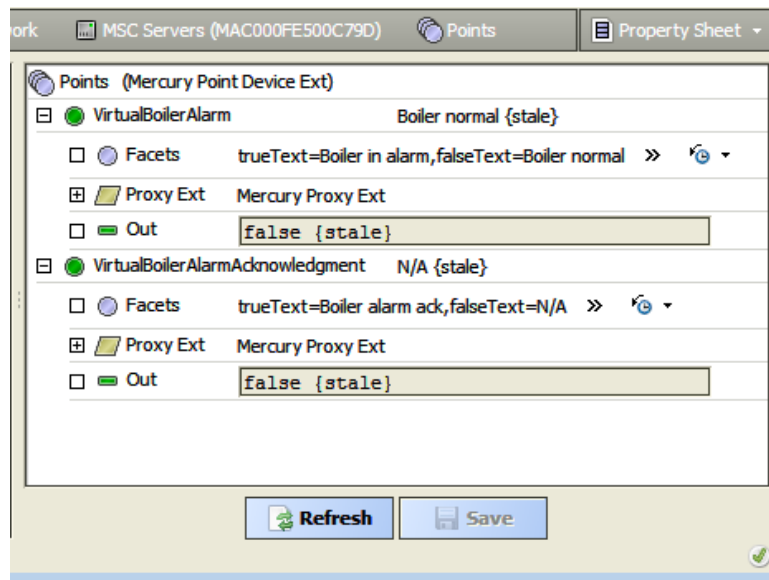
Points properties describe the proxy points that relate to portal events.

The screenshot shows a configuration window titled 'Points'. It contains a list of points with checkboxes and values: 'IsLatched [Portal 1]' (checked) with value 'true {ok}', 'Facets' (unchecked) with value 'trueText=true, falseText=false', 'Proxy Ext' (checked) with value 'Mercury Proxy Ext', 'Out' (unchecked) with value 'true {ok}', 'IsOpen [Portal 1]' (checked) with value 'false {ok}', 'IsAccessOverridden [Portal 1]' (checked) with value 'false {ok}', 'IsMasked [Portal 1]' (checked) with value 'false {ok}', 'IsTampered [Portal 1]' (checked) with value 'false {ok}', 'IsForcedOpen [Portal 1]' (checked) with value 'false {ok}', 'IsHeldOpen [Portal 1]' (checked) with value 'false {ok}', and 'IsDeviceInAlarm [Portal 1]' (checked) with value 'false {ok}'.

| Type | Value | Description |
|--------------------|--------------------------------------|--|
| IsLatched | Default: true or false | This is a state of the locking device for a portal. It means that the portal is in a locked position. |
| IsOpen | Default: true or false | This means that a door is normally open. It does not indicate an alarm state. |
| IsAccessOverridden | Default: true or false | This is a state of the door in which normal access control has been suspended. A door in this state may be latched or unlatched. The purpose of this state is to keep the door either latched or unlatched until the IsAccessOverridden state is released. |
| IsMasked | Default: true or false | This is a state in which normal reporting of state changes is suspended. A portal or an input point may be masked for alarm events. If a portal or an input point is masked, it is not monitored, and it will not generate alarm events. A portal may be in one of two states: monitored or masked. By default, state has a value of "Masked." |
| IsTampered | Default: true or false | |
| IsForcedOpen | Default: true or false | |
| IsHeldOpen | Default: true or false | |
| IsDeviceInAlarm | Default: true or false | Indicates that the input is in an alarm condition. An input may not be able to function correctly. An audible signal may indicating an alarm condition that requires immediate attention, such as an alarm initiated from an intrusion detector, door switch, or the like. |

Virtual points

Each pair of points is associated with an alarm proxy point in the BAS. An input proxy point reports the state of the device represented by the proxy point to the Mercury panel and an output virtual point communicates the acknowledgment from the Mercury panel back to the NiagaraAX station.



| Type | Value | Description |
|-----------|--------------------------------------|--|
| Facets | Default: true or false | Changing Facets may help to explain how these points are used. |
| Proxy Ext | Multiple properties | These properties are standard Niagara properties. For more information, see <i>ProxyExt Properties</i> (module://docDrivers/doc/aProxyExt-Properties.html) in the <i>NiagaraAX Drivers Guide</i> (module://docDrivers/doc/index.html). |
| Out | | Indicates the current status of the virtual point. |

Event Listener

Event Listener defines the properties related to asynchronous event monitoring, which provides a more efficient alternative to point polling.

Event listener properties

| | |
|--|---------------------------------------|
| <input checked="" type="checkbox"/> Event Listener | Mercury Event Listener |
| <input type="checkbox"/> Enabled | <input checked="" type="radio"/> true |
| <input type="checkbox"/> Server Tcp Port | 4001 |
| <input type="checkbox"/> Ssl Enabled | <input checked="" type="radio"/> true |
| <input type="checkbox"/> Status | Not listening |

The following are properties unique to the Mercury driver:

| Type | Value | Description |
|-----------------|--|--|
| Enabled | true false | Turns asynchronous event notification on and disables poll scheduling (to view this change, double-click the device, and expand Communicator > Poll Scheduler). |
| Server Tcp Port | Default = 4001 | Defines the Tcp port that the JACE uses to listen for asynchronous event notices from the Mercury panel. |
| Ssl Enabled | true false | Set this property to <code>true</code> to establish a secure socket, which is required by the Mercury driver. |
| Status | Not listening Listening | Reports the activation status of the Event Listener. |

MercuryDeviceFolder

MercuryDeviceFolder is the Mercury implementation of a folder under a **MercuryNetwork**.

Typically, you add a **MercuryDeviceFolder** using the **New Folder** button in the **Mercury Device Manager** view of the **MercuryNetwork**.

Each **MercuryDeviceFolder** has its own **Mercury Point Manager** view. The **MercuryDeviceFolder** is available in the **mercury** palette.

MercuryPointFolder

MercuryPointFolder is the Mercury implementation of a folder under a **MercuryDevice** that contains proxy and virtual points.

The **MercuryPointFolder** is available in the **mercury** palette. Use this folder to group points for easy management.

Mercury Alarm Extensions

All of the Mercury-specific alarm extensions are versions of their standard NiagaraAX counterparts with the addition of two properties used to define the input and output ords for alarm reporting and acknowledgment.

Mercury alarm extensions

The Mercury alarm extensions are:

- MercuryOutOfRangeAlarmExt
- MercuryStringChangeOfValueAlarmExt
- MercuryBooleanChangeOfStateAlarmExt
- MercuryBooleanCommandFailureAlarmExt
- MercuryEnumChangeOfStateAlarmExt
- MercuryEnumCommandFailureAlarmExt

- MercuryStatusAlarmExt

For more information about these alarm extensions, see *About Alarm Extensions* (module://docUser/doc/aAlarmExtensions.html) in the *NiagaraAX User Guide* (module://docUser/doc/index.html).

Additional properties

Each alarm extension has two additional properties for Mercury alarms as illustrated and explained below.

| | |
|--|---|
| <input type="checkbox"/> Alarm Class | Default Alarm Class ▼ |
| <input type="checkbox"/> Meta Data | » ↺ ▼ |
| <input type="checkbox"/> Alarm Point Ord | station: slot:/Drivers/MercuryNetwork/MSC\$20Servers\$20\$28MAC ▼ |
| <input type="checkbox"/> Ack Point Ord | station: slot:/Drivers/MercuryNetwork/MSC\$20Servers\$20\$28MAC ▼ |

| Type | Value | Description |
|-----------------|------------------------|---|
| Alarm Point Ord | Standard NiagaraAX Ord | <p>This point has to be a BooleanWriteable so that when an alarm occurs on the Niagara device proxy point with which this point is associated, this virtual output point is set to true. When set to true it activates a device on the Mercury side, such as a siren, and appears as an alarm on the Mercury security console. The Mercury device reflects the state of the Niagara device turning on or off accordingly.</p> <p>You define the virtual point to use by clicking the down arrow to the right of the field and selecting the virtual output point discovered from the Mercury panel.</p> |
| Ack Point Ord | Standard NiagaraAX Ord | <p>This point can be any Boolean point, input or output under the set of points discovered from the Mercury panel. The security console on the Mercury side has access to this point. When the point is set to true, it triggers an acknowledgement in the Niagara alarm service and clears all alarms associated with the device.</p> <p>You define the virtual point to use by clicking the down arrow to the right of the field and selecting the virtual output point discovered from the Mercury panel.</p> |

Mercury driver component views (plugins)

Mercury driver plugins are the views that accompany each component. A WbPlugin is a widget designed to provide plugin functionality in the Workbench tool environment.

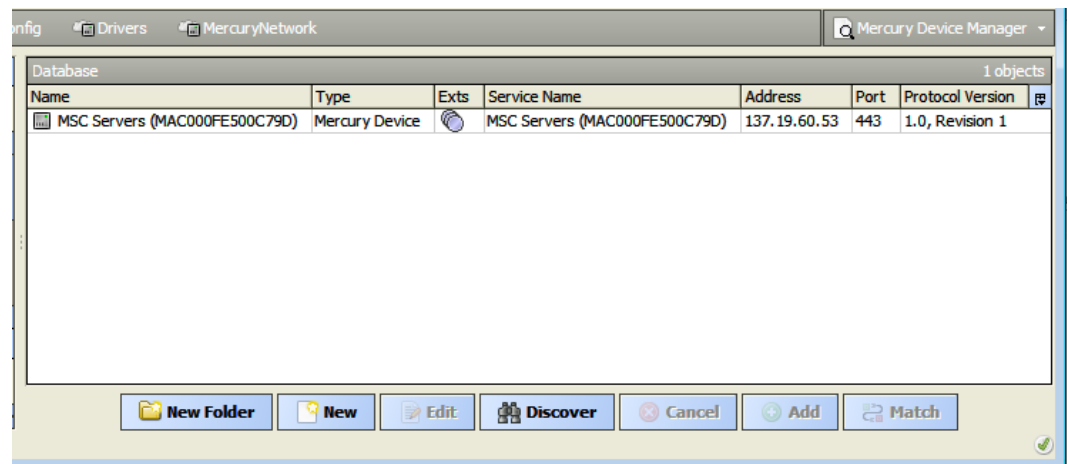
The following plugins are provided by the Mercury driver:

- [Mercury Device Manager plugin, page 35](#)
- [Mercury Point Manager plugin, page 35](#)

Mercury Device Manager plugin

The **Mercury Device Manager** view displays the Mercury panels that have been added to the database.

For more information about Device Manager, see *About the Device Manager* (module://doc-Drivers/doc/aDeviceManagerOverview.html) in the *NiagaraAX Drivers Guide* (module://doc-Drivers/doc/index.html).

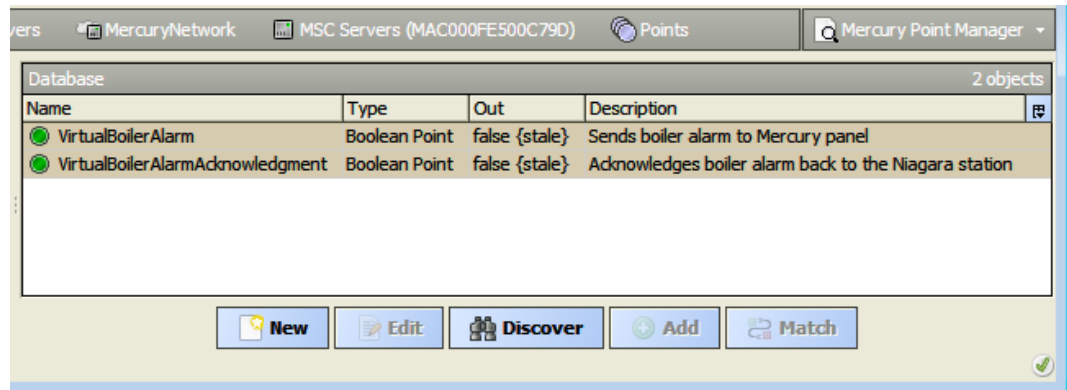


The buttons provide standard NiagaraAX device functions. For information about them, see the *NiagaraAX Drivers Guide* (module://docDrivers/doc/index.html).

| Type | Value | Description |
|------------------|-----------------|---|
| Name | Alpha numeric | The name used when the device was added to the database. |
| Type | Mercury Device | Indicates that this device is a Mercury panel. |
| Exts | | Provides shortcut access to default manager views for the component's device extensions—in this case, "Points." |
| Service Name | Alpha numeric | The name assigned to the Mercury panel on the Mercury side. |
| Address | nnn.nnn.nnn.nnn | IP address for the Mercury panel. |
| Port | Default = 443 | The NiagaraAX port use to communicate with the panel. |
| Protocol Version | | The version of the PSIA protocol supported. |

Mercury Point Manager plugin

For more information about Point Manager, see *About the Point Manager* (module://doc-Drivers/doc/aPointManager.html) in the *NiagaraAX Drivers Guide* (module://docDrivers/doc/index.html).



| Type | Value | Description |
|-------------|---------------|--|
| Name | Alpha numeric | The name assigned when the point was added to the database. |
| Type | Boolean Point | Indicates the type of point. |
| Out | false true | The last polled value (out slot) of a proxy point. This value reflects device status and facets. |
| Description | | The description created when the point was added to the database. |

INDEX

A

| | |
|---------------------------------------|----|
| Add device dialog | 23 |
| Add/edit points dialog | 25 |
| Alarms | |
| set up | 15 |
| Asynchronous event notification | 12 |
| Asynchronous notification | 13 |

B

| | |
|-----------------|----|
| BAS | |
| configure | 15 |

C

| | |
|----------------------------|----|
| Commissioning a JACE | 3 |
| Component views | 34 |
| Components | 25 |
| Confidentiality | 2 |
| Configuration | |
| plan | 1 |
| Configurations | 19 |
| Copyright notice | 2 |

D

| | |
|-----------------------------------|-------|
| Device | |
| add manually | 7 |
| Device ID | 28 |
| Device Manager plugin | 35 |
| Device properties | 8, 26 |
| Dialogs | 22 |
| Discovery | |
| of panel | 6 |
| Discovery Parameters dialog | 22 |
| document change log | iii |
| documentation | |
| related | iii |
| Doors | 19 |

E

| | |
|---------------------------|--------|
| EP1501 | 19 |
| version tested with | 3 |
| Error messages | 17 |
| Event Listener | 12, 31 |
| Event notification | |
| asynchronous | 12 |
| Extensions | 32 |

I

| | |
|--------------|----|
| Inputs | 19 |
|--------------|----|

J

| | |
|-------------------------|---|
| JACE requirements | 2 |
|-------------------------|---|

L

| | |
|-----------------------------|---|
| Licenses required | 2 |
| Licensing requirement | 2 |

M

| | |
|--|-------|
| Mercury | |
| components | 3 |
| modules | 3 |
| Mercury Communicator | 26 |
| Mercury Device Manager plugin | 35 |
| Mercury driver | |
| about | 19 |
| components | 25 |
| Mercury panel | 19 |
| alarm extensions | 19 |
| discovery | 6, 20 |
| Mercury Panel | |
| pre-configuration | 2 |
| Mercury proxy points | 20 |
| MercuryBooleanChangeOfStateAlarmExt | 32 |
| MercuryBooleanCommandFailureAlarmExt | 32 |
| MercuryDeviceFolder | 32 |
| MercuryEnumChangeOfStateAlarmExt | 32 |
| MercuryEnumCommandFailureAlarmExt | 32 |
| MercuryOutOfRangeAlarmExt | 32 |
| MercuryPointDeviceExt | 35 |
| MercuryPointFolder | 32 |
| MercuryStatusAlarmExt | 32 |
| MercuryStringChangeOfValueAlarmExt | 32 |
| Modules | |
| install | 3 |

N

| | |
|------------------|----|
| Network | |
| add | 5 |
| properties | 25 |
| Niagara | |
| configure | 15 |

O

Outputs 19

P

Panel 19

Patent notice.....2

Platform requirements2

Plugins..... 34

MercuryPointDeviceExt 35

Points

add portal points manually 11

discover portal inputs9

discover portal outputs9

interrogating 11

portal status 19

set up polling..... 13

virtual..... 21

Points properties 29

Portals 19

Properties

device..... 26

network 25

Q

Quick start1

R

Requirements

licensing2

modules.....2

pre-configuration2

security.....2

version of Workbench.....2

S

Security requirement2

SSL.....2

systems integrator2

T

TCP/IP settings

DNS.....3

Gateway IP address3

Terms..... 22

Trademarks2

Troubleshooting..... 17

V

Virtual points..... 21

W

Workbench

install3

version required.....2