Technical Document

Niagara$^{AX-3.x}$ oBIX Guide

Updated: February 11, 2015

Niagara^{AX} oBIX Guide

Copyright © 2015 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 own-
ers; 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 with-
out 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 Engi-
neers. 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 trade-
marks 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.
Copyright © 2015 Tridium, Inc.
All rights reserved. The product(s) described herein may be covered by one or more U.S or foreign patents of Tridium.

CONTENTS

ObixDriver Component Guides. 5–1

ObixDriver Component Guides Summary 5–1

Preface

This documents usage of the oBIX driver (obixDriver) for the NiagaraAX platform.

Document Change Log

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

- Updated: February 11, 2015
  Minor corrections to long-standing errors in "Component Guides" and Plugin Guides" summary descriptions about Obix driver client support for schedules. Only the specialized R2ObixClient has a Schedules device extension (R2ScheduleDeviceExt), which has an ObixScheduleManager view, which you use to add ObixScheduleExport components. A summary description for an "ObixScheduleDeviceExt" was removed, and edits were made in the descriptions of the items listed above.

- Updated: February 15, 2008
  Changed document to reference the *NiagaraAX Drivers Guide*, a new document created from sections formerly in the *NiagaraAX User Guide.*

- Updated: June 25, 2007
  Minor updates. Changes in the section "License requirements" on page 1-1, reflecting client operation change in "foreign limits" of devices, points, and so on. Additional details were added in the "Obix Server Operation" concepts section, including a new subsection "Serving history queries from oBIX clients" on page 3-5. The "Beta Draft" header and change bars were removed from the PDF version of this document.

- Updated: May 23, 2007
  Generally minor updates. Throughout, mentioned driver support in AX-3.2. More information in "License requirements" on page 1-1, and a few more conceptual details in the "NiagaraAX Obix Concepts" section, in subsections "About the Obix Network" on page 3-2, and "Obix Server Operation" on page 3-3. In "ObixDriver Component Guides" section, added subsection on new R2PointDeviceExt component (applies to R2ObixClient only). Note that in the PDF version of this document update, magenta "change bars" appear in page margins, to denote changed or new text.

- Updated: April 18, 2007
  Completely reworked as a "single-source" document, replacing the previous PDF-only *NiagaraAX oBIX User Guide*. Includes more details throughout, although more conceptual details will be added a later date.

- Revised: October 9, 2006
  New cover design with flyleaf (including copyright and trademark notices), as well as other minor formatting changes.

- Revised: August 31, 2006
  Only content change was "should" changed to "must" in the Href description found in the section about the "ObixProxyExt." Other minor formatting changes.

- Initial draft document: August 23, 2006
  Initial publication as PDF-only document.

Compatibility and Installation

Currently, this section has the following subsections:

- Compatibility
- License requirements
- Installation

Compatibility

NiagaraAX oBIX software meets the following compatibility criteria:

- Specification compatibility
- NiagaraAX platform compatibility

Specification compatibility

At the time of this document, the NiagaraAX oBIX driver meets "Committee Specification 1.0." using the specification document identifier of `obix-1.0-cs-01`. Specification documents are currently found at OASIS at this URL: http://www.oasis-open.org/committees/documents.php?wg_abbrev=obix.

NiagaraAX platform compatibility

NiagaraAX oBIX software will function on all NiagaraAX release 3.1 and above (AX-3.2) host platforms.

License requirements

To use the NiagaraAX obixDriver, you must have a target NiagaraAX host (JACE, AxSupervisor) that is licensed with the "`obixDriver`" feature. This provides the station with oBIX *client* operation.

Note: *To also operate as an oBIX server, note the* `obixDriver` *license feature must contain the attribute* "`export=true`". *For an example see below.*

```
<feature name="obixDriver" expiration="never" device.limit=none export=true
foreignDevice.limit=500 foreignHistory.limit=500 foreignPoint.limit=500 foreign-
Schedule.limit=500 history.limit=none point.limit=none schedule.limit=none
parts="ENG-WORKSTATION"/>
```

In addition, oBIX server operation requires the host to have the `web` *module installed, and so be licensed for the "*`web`*" feature.*

Note: *Note that for oBIX client operation, limits on "foreign" devices, histories, proxy points, or schedules may exist in your license's* `obixDriver` *feature. Such "foreign limits" apply only to external oBIX servers mapped as ObixClients that are not Niagara stations (either NiagaraAX or Niagara R2).*

In other words, Niagara oBIX servers mapped into the station's ObixNetwork (usually Niagara R2 stations) are not counted in these foreign limits, only in the "regular" limits, which are typically unlimited (have values of "`none`*") as shown in the license example above.*

Installation

From your PC, use the Niagara Workbench 3.*n.nn* installed with the "installation tool" option (checkbox "This instance of Workbench will be used as an installation tool"). This option installs the needed distribution files (*.dist* files) for commissioning various models of remote JACE platforms. The dist files are located under your Niagara install directory under a "sw" subdirectory.

For details, see "About your software database" in the *Platform Guide*.

Apart from installing the 3.*n.nn* version of the Niagara distribution in the JACE, make sure to also install the *obix* and obixDriver modules, plus any modules shown as dependencies. If the JACE will be operating as an oBIX server, make sure the *web* module is installed (and that feature is licensed). Upgrade any modules shown as "out of date".

For details, see "Software Manager" in the *Platform Guide*.

Following this, the remote JACE is now ready for Obix configuration in its running station, as described in the rest of this document. See the next section "oBIX Quick Start" for a series of task-based procedures.

oBIX Quick Start

This section provides a collection of procedures to use the NiagaraAX obixDriver to build an ObixNetwork with proxy points and other components. Like other NiagaraAX drivers, you can do most configuration from special "manager" views and property sheets using Workbench.

Note: *First see "Compatibility and Installation" on page 1-1 for licensing and software requirements.*

These are the main quick start subsections:

- Add the ObixNetwork
- Client usage:
  - Add ObixClient devices
  - Create Obix proxy points
  - Import oBIX histories
- Server usage:
  - Enable oBIX server operation
  - Expose writable control points for external link (input) control

Add the ObixNetwork

Only one ObixNetwork is supported (or needed) in a station, regardless of whether you are using oBIX client or server functions, or both.

To add an ObixNetwork, perform the followings:

Add an ObixNetwork

To add an ObixNetwork in the station

Use the following procedure to add an ObixNetwork component under the station's Drivers container.

Step 1   Double-click the station's **Drivers** container, to bring up the **Driver Manager**.

Step 2   Click the **New** button to bring up the New network dialog. For more details, see "Driver Manager New and Edit" in the *Drivers Guide*.

Step 3   Select "Obix Network," number to add: 1 and click **OK**.

This brings up a dialog to name the network.

Step 4   Click **OK** to add the ObixNetwork to the station.

You should have an ObixNetwork named "ObixNetwork" (or whatever you named it), under your Drivers folder, showing a status of "{ok}" and enabled as "true."

Add ObixClient devices

After adding an ObixNetwork, you can use the network's default "client manager" view to add the appropriate ObixClient and/or R2ObixClient devices.

Note: *The general "client/server" naming in this driver follows a "convention" used in some other drivers, for example the OPC driver and Modbus drivers, where a "client" device actually represents a server device (here, an oBIX server), and associated NiagaraAX components are named "client" because a client connection is used to retrieve data.*

To add ObixClients in the network

Use the following procedure to add devices in the network.

Step 1   In the Nav tree or in the Driver Manager view, double-click the ObixNetwork, to bring up the device manager (Obix Client Manager).

Note: *For general device manager information, see the* "About the Device Manager" *section in the* Drivers Guide.

Step 2   Click the **New** button to bring up the **New** device dialog.

Depending on the target oBIX server type, in **Type** select either ObixClient or R2ObixClient (where the latter applies only to a Niagara R2 host running the R2 obix driver).

Note: *If using the oBIX driver to integrate Niagara R2 stations in NiagaraAX, refer to the the engineering notes document* Niagara R2 to NiagaraAX via oBIX, *which has* much more information *than this document on that topic. Note the R2ObixClient is the only valid choice in that case.*

Step 3   Select for number to add: 1 (or more, if multiple) and click **OK**.

This brings up a dialog to name the device(s), enter the lobby URI, as well as any credentials needed for authentication (user name and password).

Step 4   Click **OK** to add the client device(s) to the network.

You should see the device(s) listed in the client manager view, showing a state of "Attaching" that changes to "Attached."

Note: *If a device appears down with a state of "*Detached*" check the syntax of the lobby URI and credentials. You can simply double-click a device in the client manager to review settings in an* **Edit** *dialog, identical to the* **New** *dialog when you added it. Also see the next section,* "Independent oBIX server verification".

After making any device changes, click **Save***, then right-click the device and select* **Actions > Ping***.*

Independent oBIX server verification

Note you can independently verify if a host is operating as an oBIX server, using a web browser, providing that you know its lobby's URI, as well as its login credentials (if needed). Simply open a web browser connection to that host, using its lobby URI.

For a Niagara R2 or NiagaraAX station operating as an oBIX server, the lobby URI syntax is as follows:

```
http://<host>[:port]/obix
```

where *<host>* is IP address or hostname, and [:port] is optional (if omitted, assumed as 80).

See "Obix Server Operation" on page 3-3 for additional details.

Note that a Niagara R2 or AX station will prompt you for a valid station user login, however, other oBIX servers may allow access without authentication. Following login, you should see an HTML representation of the station's oBIX lobby, including hyperlinks to traverse into the object tree structure.

Create Obix proxy points

As with device objects in other drivers, each ObixClient device has a **Points** extension that serves as the container for proxy points. The default view for any Points extension is the Point Manager (and in this case, the "**Obix Point Manager**"). You use it to add Obix proxy points under any client device.

For general information, see the "About the Point Manager" section in the *Drivers Guide*. Also see "obixDriver-ObixPointManager" on page 4-4.

Note: *Like the point managers in many other drivers, the* **Obix Point Manager** *offers a "Learn mode" with a* **Discover** *button and pane. Learned oBIX objects are found in the expandable "lobby" after a discover.*

To add Obix proxy points

Once a client Obix device is added, you can add proxy points to read and write data.

Step 1   In the **Client Manager**, in the **Exts** column, double-click the Points icon 🕲 in the row representing the device you wish to create proxy points.

This brings up the **Obix Point Manager**.

Step 2   (Optional) Click the **New Folder** button to create a new points folder to help organize points, and give it a short name, or whatever name works for your application. You can repeat this to make multiple points folders, or simply skip this step to make all proxy points in the root of **Points**.

Note that all points folders have their own **Obix Point Manager** view, just like **Points**. If making points folders, double-click one to move to its location (and see the point manager).

Step 3 At the location needed (**Points** root, or a points folder), click the **Discover** button in the point manger.

This launches an Obix Point Discovery job, after which an expandable "lobby" node appears in the discovered pane.

Step 4 In the discovered pane of the Obix Point Manager, expand the root "lobby" to see the tree organization. Items of practical interest for proxy points are typically under a "config" branch.

- Items that show a mode of "RW" can be proxied as either a writable point or a read-only point.
- Items that show a mode of "RO" can be proxied as a read-only point only.

Step 5 Double-click an item to add as a proxy point.

The **Add** dialog appears, in which you select a point "Name" and "Type", and typically review the other fields and change only if necessary. For more details, see "obixDriver-ObixProxyExt" on page 5-4.

Step 6 Click **OK**.

This adds the Obix proxy point to the database, where it is visible in the database pane of the view, showing the current value of the item.

Step 7 Continue to add proxy points as needed under the **Points** extension of each ObixClient device.

As needed, double-click one or more existing points for the **Edit** dialog, similar to the **New** dialog used to create the points. This is commonly done for re-editing items like names or facets.

Import oBIX histories

As with device objects in other a few other drivers, each ObixClient device has a **Histories** extension that serves as the container for history import descriptors. The default view for any Histories extension is the History Import Manager (and in this case, the "**Obix History Manager**"). You use it to add history imports, which create histories in the local station with data imported from the oBIX server.

For general information, see the "History Import Manager" section in the *Drivers Guide*. Also see "obixDriver-ObixHistoryManager" on page 4-3.

Note: *Like the history import managers in other drivers, the* **Obix History Manager** *offers a "Learn mode" with a* **Discover** *button and pane. Learned oBIX histories are found in the expandable "lobby" after a discover.*

To import oBIX histories

Once an ObixClient is added, you can import its histories into the history space of the station.

Step 1 In the **Client Manager**, in the **Exts** column, double-click the Histories icon 🔢 in the row representing the device from which you want to import histories.

This brings up the **Obix History Manager**.

Step 2 Click the **Discover** button in the history manger.

This launches an Obix History Discovery job, after which an expandable "lobby" node appears in the discovered pane.

Step 3 In the discovered pane of the Obix History Manager, expand the root "lobby" to see the tree organization. Items of practical interest for histories are typically under a "histories" branch, and appear with a History icon 🔺.

Step 4 Double-click a history to add a history import descriptor.

The **Add** dialog appears, in which you select its "Name" and "History Id", as well as its execution schedule and (local) collection parameters. For more details, see "obixDriver-ObixHistoryImport" on page 5-3.

Step 5 Click **OK**.

This adds the import descriptor to the database, where it is visible in the database pane of the view, showing "null" as last success. To archive locally (create the local history), click to select one or more import descriptors, then click the **Archive** button.

Step 6 Continue to add history import descriptors as needed under the **Histories** extension of each ObixClient device.

As needed, double-click one or more existing import descriptors for the **Edit** dialog, similar to the **New** dialog used to create the points. This is commonly done for re-editing items like execution times.

Note: *An additional device extension* **Alarms** *may also exist under an ObixClient. Related "quick start" procedures for these extensions may be added in a later version of this document. See more details in sections "obixDriver-ObixAlarmDeviceExt" on page 5-1 and "obixDriver-ObixAlarmManager" on page 4-1.*

Server operations

The following tasks apply to server usage of the obixDriver:

- Enable oBIX server operation
- Expose writable control points for external link (input) control

Enable oBIX server operation

Note: First see *"License requirements"* on page 1-1 for special server license and software requirements.

To enable the station for oBIX server operation

Use the following procedure to enable oBIX server operation.

Step 1   Right-click the station's **ObixNetwork** and select **Views > Property Sheet**.

The **Property Sheet** appears.

Step 2   Expand the **Server** slot.

Set its Enabled property to true (if not already). See "Obix Server Operation" on page 3-3 for related details.

Step 3   Verify its Status property reads "{ok}".

Step 4   Click the **Save** button.

The station's Lobby URI is http://*<hostnameOrIP>*/obix

The station's WSDL URI is http://*<hostnameOrIP>*/obix/wsdl

Expose writable control points for external link (input) control

You can selectively choose writable control points in the station (BooleanWritable, EnumWritable, NumericWritable, and StringWritable) to expose to external oBIX clients for write access at a particular control level. Note this includes any writable proxy points in the station.

To expose writable control points for external link control

Use the following procedure to expose writable control points for prioritized input writes.

Step 1   Right-click the station's **ObixNetwork** and select **Views > Property Sheet**.

The **Property Sheet** appears.

Step 2   Double-click the **Exports** folder 📁.

The **Obix Export Manager** view appears

Step 3   Click the **Discover** button in the export manger.

This opens the station's **Config** component tree in the discovered pane, as an expandable node.

Step 4   In the discovered pane of the Obix Export Manager, expand the component tree. Note that only writable control points are shown (other component types are filtered from view).

Step 5   Double-click a point to add an export descriptor.

The **Add** dialog appears, in which you select its "Name" and the priority level (Priority) of its input you wish to expose to oBIX. For details, see "Reserving writable inputs for oBIX client access" on page 3-3.

Step 6   Click **OK**.

This adds the export descriptor to the database, where it is visible in the database pane of the view, showing the current value of the writable point. Note this automatically creates a link (nub) on the target writable control point, such that link contention from within Niagara will not occur.

Step 7   Continue to add Obix Exports as needed under the **Exports** folder of the ObixNetwork.

CHAPTER 3

NiagaraAX Obix Concepts

This section, when completed, will provide conceptual details on the NiagaraAX Obix driver and its components, including views. These are the main (and planned) subsections:

- oBIX terms
- About Obix Architecture
- About the Obix Network
- Obix Client Manager
- About the ObixClient
- Obix Point Manager
- Obix Proxy Point
- Obix History Manager
- Obix Alarm Manager
- Obix Schedule Manager
- Obix Server Operation

Note: *Additional conceptual details about the NiagaraAX Obix driver can be found in the engineering notes document Niagara R2 to NiagaraAX via oBIX, which is specific to using oBIX for Niagara R2 to NiagaraAX integrations.*

oBIX terms

The following list of terms and abbreviations is specific to oBIX usage in NiagaraAX, and covers entries used in this document. For the definitive collection of terms found in oBIX publications, refer to OASIS (at the time of this document) at the following URL:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=obix.

Note: *For general NiagaraAX terms, see the Glossary in the* User Guide*.*

lobby The oBIX lobby is the root of a server's oBIX object tree. The lobby has certain semantics associated with it, such as how to create watches and batch operations. Therefore it is important that the URI given the oBIX client is that of the lobby, and not a sub-object.

oBIX The "open Building Information eXchange" is a web services protocol designed to enable communications between building mechanical and electrical systems, and enterprise applications.

URI A Universal Resource Identifier is the location of an internet resource (for example, web-page, ftp service, and so on). This term is a more general term for the commonly used Uniform Resource Location or URL.

watch And watches. oBIX watches are subscriptions. Watches allow a client to maintain a real-time cache for the current state of one or more objects. They are also used to access an event stream from a feed in the case of alarms.

About Obix Architecture

Essentially, Obix uses the standard NiagaraAX network architecture. Obix client components are the station interface to oBIX objects in one or more oBIX servers. See "About Network architecture" in the *Drivers Guide* for more details. For example, real-time data is modeled using Obix proxy points, which reside under an ObixClient "device", which in turn resides under an ObixNetwork container in the station's DriverContainer (Drivers).

Hierarchically, the component architecture is: network, device, points extension, points (Figure 3-1).

Figure 3-1   *Obix driver architecture*
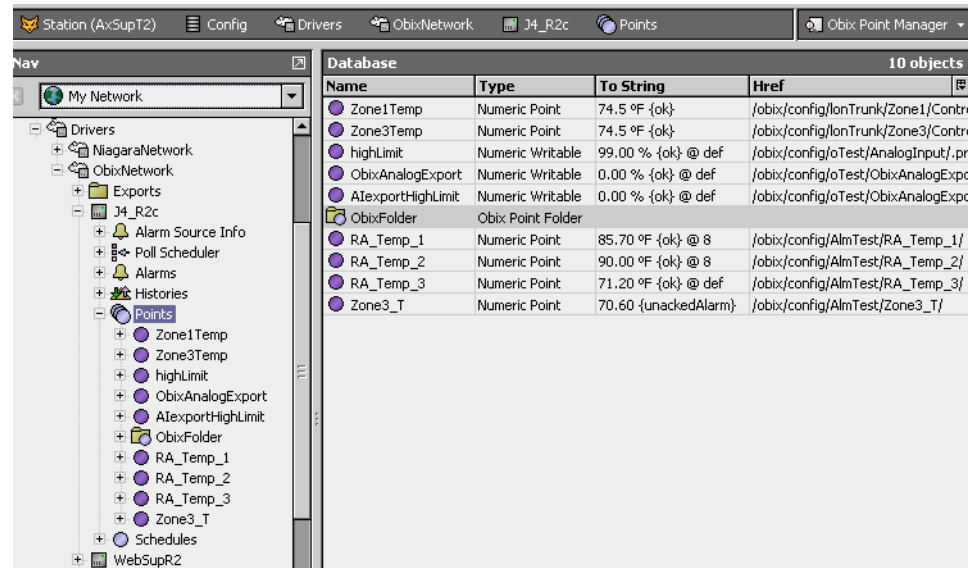
Like a few other drivers, ObixClient devices have a full range of device extensions—including **Points**, **Alarms**, **Histories**, and **Schedules**.

Note: *You use "Manager" views of Obix device extension components to add all Obix components to your station, including Obix proxy points. In these views, the Obix driver provides online "discovery" of available data items (Learn Mode), which greatly simplifies engineering.*

About the Obix Network

The ObixNetwork is the top-level container component for "everything oBIX" in a station.

Note: *Only one ObixNetwork component is valid in a station—regardless of how many Obix servers the station will make client connections to.*

The ObixNetwork should reside in the station's DriverContainer ("Drivers"). The simplest way to add an ObixNetwork is from the "Driver Manager" view, using the **New** command. Or, you can simply copy the ObixNetwork from the obixDriver palette into Drivers.

The ObixNetwork component has the typical collection of slots and properties as most other network components. For details, See "Common network components" in the *Drivers Guide.* One exception is the location of poll components (Poll Scheduler), which is *not* at the network-level, but under each ObixClient or R2ObixClient (device-level) component.

In addition, the following ObixNetwork property has special importance:

- **Thread Pool**
  Controls the number of threads used to execute all actions of all Obix objects in the network. This includes most communications with remote devices, which can be multi-threaded. In this case, if there are performance issues, you can increase the number of threads. The default value is 4.

The following sections provide additional ObixNetwork details:

- Obix Network status notes
- Obix Network monitor notes
- Obix Network tuning policy notes
- Obix Network views

Obix Network status notes

As with most "fieldbus" drivers, the status of an ObixNetwork is either the normal "ok" or less typical "fault" (fault might result from licensing error). The Health slot contains historical timestamp properties that record the last network status transitions from ok to any other status. The "Fault Cause" property further explains any fault status.

Note: *As in other driver networks, the ObixNetwork has an available "Alarm Source Info" container slot you can use to differentiate ObixNetwork alarms from other component alarms in the station. See "About network Alarm Source Info" in the* Drivers Guide *for more details.*

Obix Network monitor notes

The ObixNetwork's monitor routine verifies child ObixClient component(s)—the "pingable" device in the Obix driver. For general information, see "About Monitor" in the *Drivers Guide.*

Obix Network tuning policy notes

The ObixNetwork has the typical network-level Tuning Policy Map slot with a single default Tuning Policy, as described in "About Tuning Policies" in the *Drivers Guide.* By default, only a single ObixTuningPolicy exists, however, you can add new tuning policies (duplicate and modify) as needed.

Obix Network views

The ObixNetwork's default view is the **Obix Client Manager**, equivalent to the Device Manager in most other drivers. You use this view to add ObixClient (and/or R2ObixClient) components to the station. For details, see ObixClientManager.

Other standard views are also available on the ObixNetwork. However, apart from the Obix Client Manager, you typically access only its property sheet.

Obix Server Operation

If licensed for operation as an oBIX server (see "License requirements" on page 1-1), the station can expose components and histories accessible to oBIX clients, with access corresponding to the login credentials used for connection. This includes allowing operation (op) writes on components' actions.

Note: *Obix server operation by a NiagaraAX station is not used in any Niagara R2 to NiagaraAX integration, as the R2 oBIX driver is server-side only, without client capability.*

Note you can quickly verify if a station is operating as an oBIX server. Simply open a web browser connection to that station, using the syntax

```
http://<host>[:port]/obix
```

where *<host>* is IP address or hostname, and [:port] is optional (if omitted, assumed as 80).

For example: `http://192.168.1.94/obix` for a typically-configured station at that IP address,

or `http://192.168.1.75:85/obix` for a station running on Http Port 85 (as configured in its **WebService**) at its host IP address.

As shown in Figure 3-2, after you login with station credentials you see an HTML representation of the station's oBIX lobby, including hyperlinks to traverse into the object tree structure.

**Figure 3-2**   *Example browser connection to confirm NiagaraAX station oBIX server operation*

The following sections provide additional details:

* Reserving writable inputs for oBIX client access
* Serving history queries from oBIX clients

Reserving writable inputs for oBIX client access

The ObixNetwork provides a mechanism to reserve specific input(s) on any writable type points in the station, for op write access—effectively "linking" to oBIX for *continuous control.* Configuration is via the Obix Export Manager view on the **Exports** folder of the network (Figure 3-3).

Figure 3-3   *Obix Export Manager is default view of ObixNetwork's Exports folder*
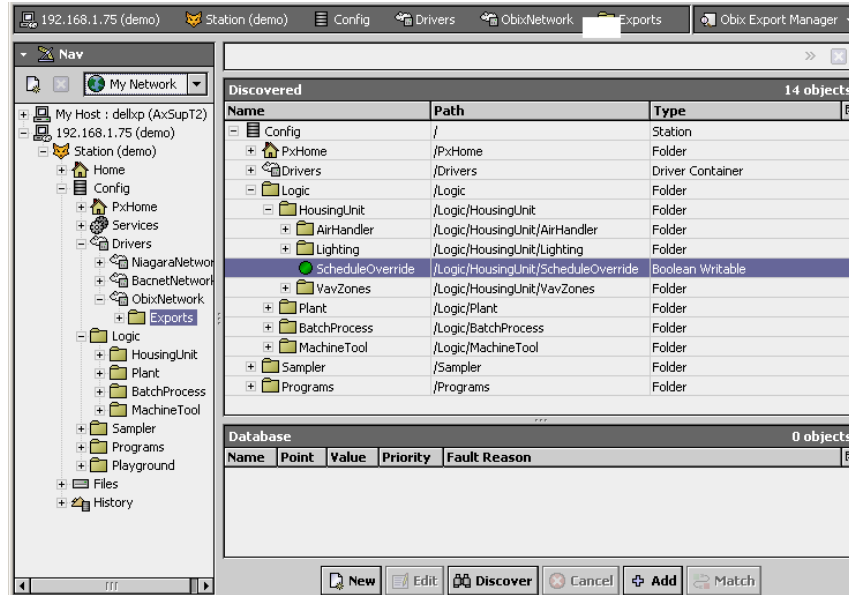
As shown in the figure above, a "Learn mode" is available in which you discover local writable points in the station (BooleanWritable, EnumWritable, NumericWritable, and StringWritable) by expanding the Config node in the discovered pane. Currently, only those point types (having priority input slots) are valid candidates for adding.

In the **Add** dialog for the ObixExport descriptor (that points to the selected writable point), you specify which priority (input) will be reserved for oBIX clients. See Figure 3-4.

Figure 3-4   *Add dialog for ObixExport descriptor specifies priority (input) used*
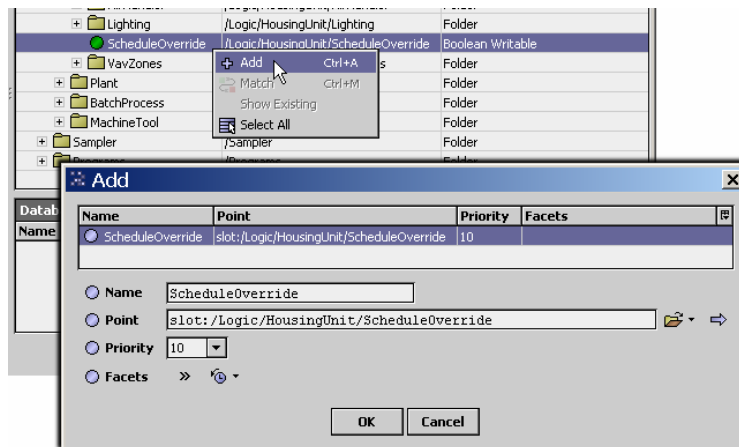
Other properties are automatically configured— although you can change the name of the export descriptor, it is typically left at default (unless multiple exported source points have the identical name).

After adding the ObixExport descriptor, the specified priority input of the writable point is linked (via a "nub") back to the descriptor, and available for link control from oBIX. See Figure 3-4.

Figure 3-5   *Example ObixExport descriptor and resulting reserved link on target writable point*
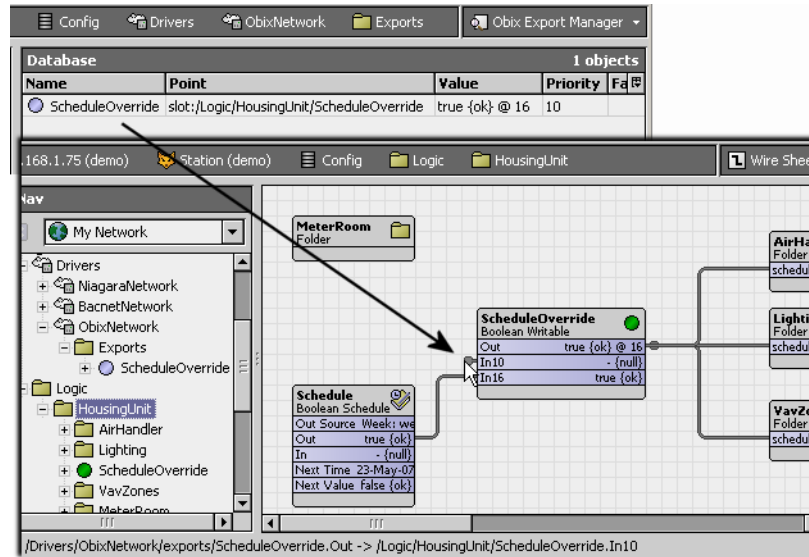

Use of ObixExports reduces the possibility of write contentions between oBIX and internal station operation.

Serving history queries from oBIX clients

Starting in obixDriver build 3.2.17 and later, histories exposed to oBIX clients provide a pre-defined set of query options, available as URIs using the HTTP "GET" mechanism (without an oBIX "op"). The available pre-defined queries match those provided in time range selections in the History Chart and History Table views, for example, "Today," "Last 24 Hours," "Yesterday," and so forth. See Figure 3-6.

Figure 3-6   *Example history query options for oBIX client connected to NiagaraAX station*
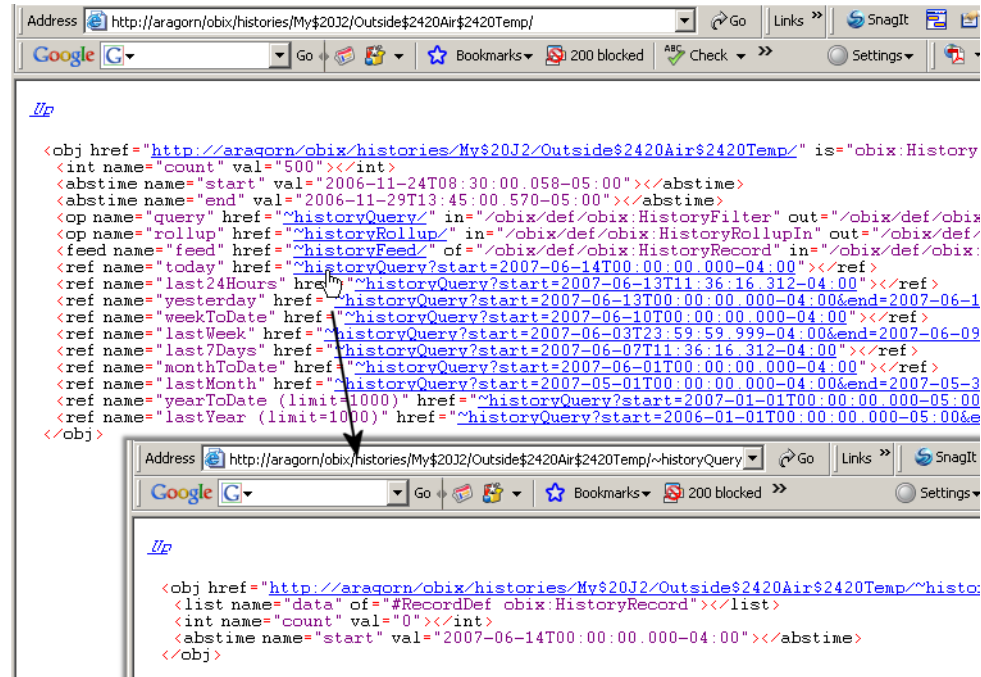
Note that for a few pre-defined history queries (Year to Date, Last Year), a default limit for number of records returned is used. If necessary, a client can modify any of the standard history queries.

ObixDriver Plugin Guides

Plugins provide *views* of components, and can be accessed many ways—for example, double-click a component in the tree for its *default* view. In addition, you can right-click a component, and select from its **Views** menu. For summary documentation on any view, select **Help > On View** (F1) from the Workbench menu, or press F1 while the view is open.

Summary information is provided here about the different obixDriver views.

ObixDriver Plugin Guides Summary

Summary information is provided on views specific to components in the obixDriver module, with views listed in alphabetical order as follows:

- ObixAlarmManager
- ObixClientManager
- ObixExportManager
- ObixHistoryManager
- ObixPointManager
- ObixScheduleManager

obixDriver-ObixAlarmManager

Use the ObixAlarmManager to discover, add, and manage alarm feeds of a selected ObixClient or R2ObixClient. The ObixAlarmManager is the default view of the Alarms device extension (ObixAlarmDeviceExt, R2AlarmDeviceExt) under these devices. To view, *double-click* the Alarms extension, or right-click and select **Views > Obix Alarm Manager**.

Although not a standard view in the driver architecture, it is similar to typical point and history manager views. As in those views, there is a Discovered table (if in Learn mode) and a Database table. In the discovery pane, objects with a value in the "Feed" column are valid alarm sources, and can be added as alarm imports (ObixAlarmImport or R2AlarmiImport, depending on parent device type).

Note: *During discovery, results are cached. Therefore if the server database is modified after a discovery has occurred (or while a discovery is in progress), the discovery pane may be inaccurate. Click the* **Discover** *button again to clear the cache.*

For details in a Niagara R2 to AX application, see the "R2ObixClient Alarms" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

Discovered table The Discovered table in the ObixAlarmManager view has the following available columns:

- **Obix Name**
  Name of the object on the oBIX server.
- **Href**
  The URI of the object on the oBIX server.
- **Feed**
  The URI of the object on the oBIX server, with ".alarm" suffix to denote a valid alarm feed.

Database table By default, the following columns appear in the Discovered table of the ObixAlarm-Manager view:

- **Name**
  Niagara name of the alarm import descriptor, often left the same as its oBIX Name.
- **Href**
  The URI of the object on the oBIX server.

- **Alarm Class**
  The local station's AlarmClass to use to process native alarms received via this alarm feed.

In addition (using the table options control), the following additional data column is available:

- **Subscription**
  Reflects whether the alarm import is Subscribed, Unsubscribed, or Pending subscription.

obixDriver-ObixClientManager

Use the **Obix Client Manager** to add, edit, and access Obix device components (ObixClients and R2ObixClients). The Obix Client Manager is the default view of an ObixNetwork. For general information, see "About the Device Manager" in the *Drivers Guide*.

Note: *Unlike in some other drivers, there is no "learn" mode (with "Discovered" pane). Instead, you use the* **New** *button to add devices.*

Added devices appear in the Database table. For details specific to an Niagara R2 to AX application, see the "ObixNetwork and R2ObixClient devices" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

Database table By default, the following columns appear in the Discovered table of the ObixClient-Manager view:

- **Name**
  Name of the device-level component for (client) interface to the oBIX server.
- **Lobby**
  URI to root of the server's object tree, using format:
  `http://<hostName or IP address>/obix`
- **Enabled**
  Indicates whether the device component is enabled (true) or disabled (false).
- **Exts**
  Provides shortcut access to the manager view for any of the component's device extensions (Alarms, Histories, Points, Schedules).
- **State**
  Reflects the client state, which is either attached, attaching, detached, or detaching.
- **Fault Cause**
  String describing the cause of the device status fault, if any.

In addition, using the table options control, the following additional data columns are available:

- **Path**
  Station path of the device-level component, relative to the root.
- **Auth User**
  User name for client access.
- **Auth Pass**
  Passphrase (password) of the Auth User.
- **Type**
  Device-level component type, currently either ObixClient or R2ObixClient.
- **State**
  Reflects the client state, which is either attached, attaching, detached, or detaching.
- **Fault Cause**
  String describing the cause of the device status fault, if any.

obixDriver-ObixExportManager

Use the ObixExportManager to add and manage export descriptors (ObixExports) for control points in the local station, such that remote oBIX clients can participate in continuous control applications (meaning, link into specific prioritized inputs of specific points).

The ObixExportManager is the default view of the ObixExportFolder under the ObixNetwork. To view, *double-click* the Exports folder, or right-click and select **Views > Obix Export Manager**.

Note: *Currently in a Niagara R2 to AX oBIX application, this view has no practical use as Niagara R2 stations running the* `obix` *driver do not have oBIX client capabilities.*

Although not a standard view in the driver architecture, it is similar to other manager views in that there is a Discovered table (if in Learn mode) and a Database table.

Discovered table The Discovered table in the ObixExportManager view has the following available columns:

- **Name**
  Name of the component in the local station.
- **Path**
  Station path of the component, relative to the root.
- **Type**
  Component type. Currently, only writable control points (including writable proxy points) can be added to the database: BooleanWritable, EnumWritable, NumericWritable, and StringWritable.

Database table By default, the following columns appear in the Discovered table of the ObixExport-Manager view:

- **Name**
  Niagara name of the export descriptor, often left the same as the source control point.
- **Point**
  Ord for the slot in the station for this control point.
- **Value**
  Current out value for the control point.
- **Priority**
  The priority input of the control point exported (linked) to oBIX for remote writes.
- **Fault Reason**
  If export descriptor is in fault, explains why. Typically this reflects misconfiguration, such as selecting a Priority level that is already linked on the target control point.

In addition (using the table options control), the following additional data column is available:

- **Facets**
  Shows the facets in use by the target control point.

obixDriver-ObixHistoryManager

Use the ObixHistoryManager to discover, add, and managing history *imports* under the Histories extension (ObixHistoryDeviceExt) of a selected ObixClient or R2ObixClient. The ObixHistory-Manager is the default view on the Histories extension. To view, *double-click* the Histories extension, or right-click and select **Views > Obix History Manager**.

There is a Discovered table (if in Learn mode) and a Database table. In the discovery pane, objects that have values in the "Start," "End," "Count," and "Query" columns are valid histories, and can be added as ObixHistoryImports.

Note: *During discovery, results are cached. Therefore if the server database is modified after a discovery has occurred (or while a discovery is in progress), the discovery pane may be inaccurate. Click the* **Discover** *button again to clear the cache.*

For details specific to an Niagara R2 to AX application, see the "R2ObixClient Histories (logs and archives)" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

Discovered table The Discovered table in the ObixHistoryManager view has the following available columns:

- **Obix Name**
  Name of the history on the oBIX server.
- **Href**
  The URI of the history on the oBIX server.
- **Start**
  Timestamp of the first record in the oBIX history.
- **End**
  Timestamp of the last record in the oBIX history.
- **Count**
  Total number of records in the oBIX history.
- **Query**
  Similar to Href but with ".log" or "query" suffix to describe history query.

Database table By default, the following columns appear in the Discovered table of the ObixHistory-Manager view:

- **History Id**
  Niagara History Id for the history created by the import descriptor, which defaults to:
  <name of ObixClient or R2ObixClient> / <oBIX history name>
- **Status**
  Status of the history import descriptor.

- **State**
  Current state of history import descriptor, as either `Idle` or `In Progress`.
- **Last Success**
  Timestamp of when the last successful history import occurred.
- **Href**
  The URI of the query to the history on the oBIX server.

In addition (using the table options control), the following additional data columns are available:

- **Name**
  Niagara name of the history import descriptor, defaulting to the name of the oBIX history.
- **Execution Time**
  Reflects the configured time to execute (re-import history).
- **Enabled**
  Reflects whether history import descriptor is enabled (true) or disabled (false).
- **Last Attempt**
  Timestamp of last attempted history import.
- **Last Failure**
  Timestamp of when last attempted history import failed (could not complete).
- **Fault Cause**
  Reason why last history import failed.
- **Full Policy**
  Full Policy of import descriptor (Roll or Stop).
- **Capacity**
  Configured record capacity of import descriptor (Unlimited, or some specific count).

obixDriver-ObixPointManager

Use the ObixPointManager to add, edit, and access Obix proxy points under the Points extension of a selected ObixClient, R2ObixClient, or in an ObixPointFolder. The ObixPointManager is the default view on all these components. To view, *double-click* the Points extension or ObixPointFolder, or right-click and select **Views > Obix Point Manager**.

As in some other point managers, there is a Discovered table (if in Learn mode) and a Database table. Every object on the remote server can me modeled as a point in the station. Non-value objects are modeled as string points, and their value is the oBIX display string.

Note: *During discovery, results are cached. Therefore if the server database is modified after a discovery has occurred (or while a discovery is in progress), the discovery pane may be inaccurate. Click the* **Discover** *button again to clear the cache.*

For details specific to an Niagara R2 to AX application, see the "R2ObixClient Points" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

Discovered table The Discovered table in the ObixPointManager view has the following available columns:

- **Obix Name**
  Name of the object on the oBIX server.
- **Value**
  Value of the object at the time of discovery (expansion of its parent's leaf in the lobby).
- **Mode**
  Either RO (read-only) or RW (read-writable). Note that an Obix proxy point for a RW item can be created either as a read-only type (NumericPoint, BooleanPoint, etc.) or as a writable type (NumericWritable, BooleanWritable, etc.).
- **Href**
  The URI of the object on the oBIX server.

Database table By default, the following columns appear in the Discovered table of the ObixPointManager view:

- **Name**
  Niagara name of the proxy point, if a "root level" point often left the same as the (discovered) object item name.
- **Type**
  Niagara type of component, as either an Obix Point Folder (for a folder) or a type of control point if an Obix proxy point (for example, Boolean Point, Boolean Writable, Numeric Point, and so on).
- **To String**
  Last read value of a data item.

- **Href**
  The URI of the object on the oBIX server.
- **Fault Cause**
  String describing the cause of the proxy point status fault, if any.

In addition, using the table options control, the following additional data columns are available:

- **Enabled**
  Reflects whether proxy point is enabled (true) or disabled (false).
- **Facets**
  Reflect the facets in use by the proxy point.
- **Conversion**
  Niagara conversion type used by the ObixProxyExt, which is typically Default.
- **Tuning Policy Name**
  Name of the Niagara ObixTuningPolicy that the proxy point is assigned to.
- **Device Facets**
  Reflects the read-only device facets used in the point's proxy extension.
- **Path**
  Station path of the proxy point component, relative to the root.
- **Read Value**
  Reflects current read value in point's ObixProxyExt.
- **Write Value**
  Reflects current write value (if any) in point's ObixProxyExt.
- **Subscription**
  Reflects whether proxy point is Subscribed, Unsubscribed, or Pending subscription.

obixDriver-ObixScheduleManager

Use the ObixScheduleManager to discover, add, and manage schedule *exports* under the Schedules extension (R2ScheduleDeviceExt) of a selected R2ObixClient. The ObixScheduleManager is the default view on this Schedules extension. To view, *double-click* the Schedule extension, or right-click and select **Views > Obix Schedule Manager**.

As in a other schedule manager views, there is a Discovered table (if in Learn mode) and a Database table. See "Schedule Export Manager" in the *Drivers Guide* for general information.

In the discovery pane, objects in the lobby tree that appear with a schedule icon 📅 can be added as schedule export descriptors (ObixScheduleExports). For details specific to a Niagara R2 to AX application, see the "R2ObixClient R2 Schedule exports" section in the engineering notes document *Niagara R2 to NiagaraAX via oBIX*.

Discovered table The Discovered table in the ObixScheduleManager view has the following available columns:

- **Obix Name**
  Name of the target schedule on the oBIX server.
- **Href**
  The URI of the schedule on the oBIX server.

Database table By default, the following columns appear in the Discovered table of the ObixScheduleManager view:

- **Name**
  Niagara name for the schedule descriptor, which defaults to same as target oBIX schedule name.
- **Subordinate**
  The URI of the target schedule on the oBIX server in the R2 station..
- **Supervisor**
  Ord of the schedule component in the local station that acts as "master."
- **Enabled**
  Reflects whether schedule export descriptor is enabled (true) or disabled (false).
- **Execution Time**
  Reflects the configured time to execute (re-export schedule events).
- **Fault Cause**
  Reason why last schedule export failed.
- **Href**
  The URI of the query to the history on the oBIX server.

In addition (using the table options control), the following additional data columns are available:

- **Subordinate Expires**
  Timestamp of when "mastering" of target oBIX schedule expires, assuming no more executions.
- **Supervisor Version**
  Timestamp of the last configuration change to local "master" schedule component.
- **State**
  Reflects whether inactive (Idle) or executing (In Progress).
- **Capacity**
  Configured record capacity of import descriptor (Unlimited, or some specific count).
- **Last Attempt**
  Timestamp of last attempted history import.
- **Last Success**
  Timestamp of when the last successful history import occurred.
- **Last Failure**
  Timestamp of when last attempted history import failed (could not complete).
- **Full Policy**
  Full Policy of import descriptor (Roll or Stop).

ObixDriver Component Guides

These component guides provides summary help on obixDriver components.

ObixDriver Component Guides Summary

Summary information is provided on components specific to the `obixDriver` module, listed in alphabetical order as follows:

- ObixAlarmDeviceExt
- ObixAlarmImport
- ObixClient
- ObixClientFolder
- ObixExport
- ObixExportFolder
- ObixHistoryDeviceExt
- ObixHistoryImport
- ObixNetwork
- ObixPointDeviceExt
- ObixPollScheduler
- ObixProxyExt
- ObixScheduleExport
- ObixServer
- ObixThreadPool
- ObixTuningPolicy
- ObixTuningPolicyMap
- R2AlarmDeviceExt
- R2AlarmiImport
- R2ObixClient
- R2PointDeviceExt
- R2ScheduleDeviceExt

obixDriver-ObixAlarmDeviceExt

ObixAlarmDeviceExt (**Alarms**) is a frozen device extension of the ObixClient component. It allows integration of native alarms from the oBIX client into the NiagaraAX alarming subsystem. Its default view is the ObixAlarmManager.

See "About the Alarms extension" in the *Drivers Guide* for general information. Alarming is not configured on this object. An oBIX server can have many alarm sources. Therefore configuration such as alarm class is done on the ObixAlarmImport object.

The ObixAlarmDeviceExt creates and manages a single watch used for all alarm feeds. The following properties are unique or have special importance to the ObixAlarmDeviceExt:

- **Watch Interval**
  This is how often the client should poll the alarm watch.

obixDriver-ObixAlarmImport

ObixAlarmImport is a child of the ObixAlarmDeviceExt (Alarms extension of ObixClient), and represents an alarm feed on the oBIX server.

The following properties are unique or have special importance to the ObixAlarmImport:

♦ **Alarm Class**
Provides a selection list of local Alarm Classes, from you which you select one to use for all alarms received from this alarm subject.

♦ **Href**
The URI of the alarm feed on the server. This value *must* be unique among all the ObixAlarmImports of any given ObixClient.

obixDriver-ObixClient

ObixClient represents the client access to an oBIX server device (note that a special-purpose variation for a Niagara R2 station also exists, as the R2ObixClient). Each is a "device-level" component in the NiagaraAX Obix driver architecture.

Note: *"Client/server" naming in the Obix driver follows a "convention" used in some other drivers, for example the OPC driver and Modbus drivers, where a "client" device actually represents a server device (here, an oBIX server), and associated NiagaraAX components are named "client" because a client connection is used to retrieve data.*

The ObixClient has the standard device component properties such as status and enabled (see "Common device components" in the *Drivers Guide* for general information). In addition, the following properties are unique or have special importance:

♦ **Lobby**
The URI to the root of the server's object tree. If the server host changes, only the authority (scheme://host[:port]) here needs to be changed and all sub-objects will work for the new host.

♦ **Auth User**
The user name the client should use to access the server. It can be blank if the server supports unauthenticated access.

♦ **Auth Pass**
The passphrase for the auth user.

♦ **Poll Scheduler**
The poll scheduler is only used when watches on the server are not working.

Two actions are available on the ObixClient, as follows:

♦ **Ping**
Sends a ping monitor request to verify device "health. "

♦ **Reattach**
Attempts to reattach to the oBIX server (detaches, then attaches).

obixDriver-ObixClientFolder

ObixClientFolder is the Obix driver implementation of a folder under an ObixNetwork. Usage is optional. Each ObixClientFolder has its own ObixClientManager view.

You can use the **New Folder** button in the ObixClientManager view to add an ObixClientFolder. It is also available in the **obixDriver** palette.

obixDriver-ObixExport

ObixExport is a child of an ObixExportFolder, and is an oBIX server concept that allows remote clients to have "link" write access to control points. While many NiagaraAX points have some actions that allow oBIX clients to write a value to them (each action appears as an "op"), an ObixExport defines a *link* to an input property of a specific NiagaraAX point, at a particular control level.

The following properties have special importance to an ObixExport:

♦ **Point**
The ord to a control point. Set this value and the export will be automatically configured.

♦ **Priority**
The priority level for linking to the control point. When changed, the export object will be automatically reconfigured.

♦ **Facets**
These should always be a copy of the facets of the target control point. This is automatically configured.

♦ **Status**
This is for use in the ObixExportManager view. It will only ever be fault or ok, and does not affect the value read by oBIX clients. It will be fault if point doesn't point to a control point, or something is already linked into the control point at the priority level specified in the Priority property.

♦ **Fault Reason**
If status is fault, this explains why.

- **WritePoint**
  This is for oBIX encoding, and should not be modified in any way.

See "Reserving writable inputs for oBIX client access" on page 3-3 for more details.

obixDriver-ObixExportFolder

ObixExportFolder (default name **Exports**) is a frozen container slot under an ObixNetwork, used to simplify the addition and management of ObixExport objects. It is not limited to containing ObixExport objects, and can be duplicated as needed (no restriction on where and how many ObixExportFolders can be in a station database). Each ObixExportFolder has its own ObixExportManager view.

Note: *In an ObixNetwork used only for Niagara R2 station integration (all interaction with oBIX devices is limited to R2 Niagara hosts), the ObixExportFolder and ObixExports are not used, as the Niagara R2 stations operate as oBIX servers only (have no client interface).*

obixDriver-ObixHistoryDeviceExt

ObixHistoryDeviceExt (default name **Histories**) is a frozen device extension of the ObixClient and R2ObixClient component. It allows the *import* of historical data from the oBIX server into the NiagaraAX history space. Use its default ObixHistoryManager view to add ObixHistoryImport descriptors.

See "About the Histories extension" and "History Import Manager" in the *Drivers Guide* for general information. For details specific to an R2ObixClient, see the "R2ObixClient Histories (logs and archives)" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

obixDriver-ObixHistoryImport

ObixHistoryImport is a "history import descriptor" child of the ObixHistoryDeviceExt (Histories extension of an ObixClient or R2ObixClient), and corresponds to a history (log or trend) on the oBIX server.

For details specific to an R2ObixClient, see the "R2ObixClient Histories (logs and archives)" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

obixDriver-ObixNetwork

ObixNetwork represents a tree of oBIX clients and ancillary objects, and is the top-level component for the Obix driver in a station. This network object is a NiagaraAX Framework convention, and has no physical correspondence to any oBIX systems.

The ObixNetwork component has the typical collection of slots and properties as most other network components. For details, See "Common network components" in the *Drivers Guide*. In addition, the following slots are unique or have special importance in an ObixNetwork:

- **Status**
  Will be fault if oBIX is not licensed on the host platform, where the license feature is "obixDriver".
- **Thread Pool**
  See ObixThreadPool on page 5.
- **Server**
  See ObixServer on page 4.

The ObixClientManager is the default view of the ObixNetwork. For details specific to an Niagara R2 to AX application, see the "ObixNetwork and R2ObixClient devices" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

obixDriver-ObixPointDeviceExt

ObixPointDeviceExt (default name Points) is the container for Obix proxy points under an ObixClient. It operates as in other drivers; see "About the Points extension" in the *Drivers Guide*. This component creates and manages a single watch used for all points that are currently read subscribed.

The following slots are unique or have special importance:

- **Watch Interval**
  How often the client should poll the point watch for changes. See "Watch operation summary".
- **Force Update**
  This action calls forceUpdate on all ObixProxyExts in this subtree.

The default and primary view for the Points extension is the ObixPointManager. Note that a different Points extension is used under an R2ObixClient device, the R2PointDeviceExt. It has a few additional properties related to the discovery of R2 objects.

Watch operation summary   Note that a watch is not really "COV." The basic mechanics of a watch are as follows:

1. The client requests a watch (object) to be created on the server.

2. The client registers (and unregisters) objects included in the server's watch, using hrefs. The standard subscription mechanism is used on the NiagaraAX client side to select/deselect objects.

3. The client periodically polls the watch on the server at the defined Watch Interval (above).

4. The server returns a list of any changes in the watched items (since the last poll).

obixDriver-ObixPointFolder

ObixPointFolder is an optional container for Obix proxy points. It provides organizational utility, and may (or may not) mirror organizational structure within the underlying oBIX server. Points can be organized in any fashion under the Points device extension (ObixPointDeviceExt).

You can use the **New Folder** button in the ObixPointManager view to add an ObixPointFolder. It is also available in the **obixDriver** palette. Each ObixPointFolder has its own ObixPointManager view.

obixDriver-ObixPollScheduler

An ObixPollScheduler is a child component of every (device-level) ObixClient. It has the standard collection of poll scheduler properties, as described in "About poll components" in the *Drivers Guide*. The poll scheduler provides a flexible polling algorithm based on four "buckets."

obixDriver-ObixProxyExt

ObixProxyExt is the proxy extension for any type of Obix proxy point. It has standard proxy extension properties such as Status and Enabled, among others (see "ProxyExt properties" in the *Drivers Guide* for more information).

In addition, the following properties are unique or have special importance:

* **Href**
  The URI to the point on the server. This value *must* be unique among all the points of any given ObixClient or R2ObixClient. This value is automatically learned upon a point discover.

* **Force Update**
  This action forces a read, updates the dynamic actions on the control point and updates the device facets in the ObixProxyExt. Lastly, if the point is supposed to be subscribed but is not, this will attempt to re-subscribe the point.

obixDriver-ObixScheduleExport

ObixScheduleExport is a "schedule export descriptor" child of an R2ScheduleDeviceExt (Schedules extension of an R2ObixClient), and corresponds to a target schedule in a Niagara R2 station to receive events from a local NiagaraAX schedule component.

For specific details, see the "R2ObixClient R2 Schedule exports" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

obixDriver-ObixServer

ObixServer (default name **Server**) is a frozen container slot under the ObixNetwork. Properties are associated with the station's oBIX server operation, and are described below.

* **Status**
  (read only) Reflects status of server as either "ok", "disabled", or "fault" (fault occurs when the platform's license does not have 'export="true"' attribute in the obixDriver feature line), yet the server is enabled. See "License requirements" on page 1-1 for related details.

* **Fault Cause**
  (read only) If status is fault, explains why, such as "Server not licensed".

* **Enabled**
  * If true (default), remote oBIX clients can access the lobby of the station, following login using station user credentials (providing host platform's license has 'export="true"' attribute in the obixDriver feature line).
  * If false, remote oBIX clients cannot access the station's lobby. The server returns HTTP error code 410 for all requests.
  Note: If enabled, but not licensed for obixDriver export (status is fault), the server returns HTTP error code 500 to all oBIX client requests.

* **Servlet Name**
  (read only) Currently fixed at: obix

* **Debug**
  Default is false. If set to true this will print debug information to the station's standard output for incoming requests from oBIX clients, as well as the server's outgoing responses.

For more details see "Obix Server Operation" on page 3-3.

obixDriver-ObixThreadPool

ObixThreadPool (**Thread Pool**) is a frozen slot under the ObixNetwork. A single property is described as follows:

- **Max Threads**
  Controls the number of threads used to execute all actions of all Obix objects in the network. This includes most communications with remote devices, which can be multi-threaded. In this case, if there are performance issues, you can increase the number of threads. The default value is 4.

obixDriver-OBixTuningPolicy

A tuning policy for the ObixNetwork, with standard NiagaraAX tuning policy properties. For an explanation of driver tuning policies, see "About Tuning Policies" in the *Drivers Guide*.

obixDriver-ObixTuningPolicyMap

A container for one or more ObixTuningPolicy(ies). You might create multiple tuning policies and assign Obix proxy points as needed, based upon different criteria. For an explanation of driver tuning policies, see "About Tuning Policies" in the *Drivers Guide*.

obixDriver-R2AlarmDeviceExt

R2AlarmDeviceExt (**Alarms**) is a special variant of the ObixAlarmDeviceExt component, as a frozen device extension under an R2ObixClient. It allows integration of "native Niagara R2 alarms" from the client into the NiagaraAX alarming subsystem. Its default view is the ObixAlarmManager.

See "About the Alarms extension" in the *Drivers Guide* for general information. For more specific details, see the "R2ObixClient Alarms" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

obixDriver-R2AlarmImport

R2AlarmImport is a child of the R2AlarmDeviceExt (Alarms extension of R2ObixClient), and represents an oBIX alarm feed from the R2 Niagara station. Typically, you create one R2AlarmImport for each discovered R2 NotificationClass node under the "NotificationService" area of the server's lobby. For more details, see the "R2ObixClient Alarms" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

obixDriver-R2ObixClient

R2ObixClient represents client access to a Niagara R2 station (host) running the ObixService, operating as an oBIX server. It is a special variant of the ObixClient component. Each is a "device-level" component in the NiagaraAX Obix driver architecture. The R2ObixClient differs by offering specialized native R2 alarming support (Alarms extension setup) and mastering of R2 schedule objects (Schedules extension setup).

The R2ObixClient has the standard device component properties such as status and enabled (see "Common device components" in the *Drivers Guide* for general information). For more details, see the "ObixNetwork and R2ObixClient devices" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX.

Two actions are available on the R2ObixClient, as follows:

- **Ping**
  Sends a ping monitor request to verify device "health. "
- **Reattach**
  Attempts to reattach to the oBIX server (detaches, then attaches).

obixDriver-R2PointDeviceExt

R2PointDeviceExt (default name **Points**) is the container for Obix proxy points under an R2ObixClient. It operates as in other drivers; see "About the Points extension" in the *Drivers Guide*. This component creates and manages a single watch used for all points that are currently read subscribed.

The following slots are unique or have special importance:

- **Watch Interval**
  How often the client should poll the point watch for changes. See "Watch operation summary".
- **Force Update**
  This action calls forceUpdate on all ObixProxyExts in this subtree.
- **Include Ui Nodes**
  Either false (default) or true. When left at false, an R2 Points Discovery job does not include Gx objects under the config branch of the lobby. If you wish to include Gx objects, set this to true and perform another Discover.

- **Include Internal Props**
  Either false (default) or true. When left at false, an R2 Points Discovery job globally omits properties of R2 objects that are typically for internal configuration only (they do not appear in config branch of the lobby). If you wish to include these properties, set this to true and perform another Discover.

See the "R2ObixClient Points" section in the engineering notes document Niagara R2 to NiagaraAX via oBIX. The default and primary view for the Points extension is the ObixPointManager.

Watch operation summary Note that a watch is not really "COV." The basic mechanics of a watch are as follows:

1. The client requests a watch (object) to be created on the server.
2. The client registers (and unregisters) objects included in the server's watch, using hrefs. The standard subscription mechanism is used on the NiagaraAX client side to select/deselect objects.
3. The client periodically polls the watch on the server at the defined Watch Interval (above).
4. The server returns a list of any changes in the watched items (since the last poll).

obixDriver-R2ObixScheduleDeviceExt

R2ObixScheduleDeviceExt (`Schedules`) is a frozen device extension of the R2ObixClient component. It allows exporting of NiagaraAX BooleanSchedules events to R2 Schedule objects in the target R2 station. Use its default Obix Schedule Manager view to export schedules.

See "Schedule Export Manager" in the *Drivers Guide* for general information. For specific details, see the "R2ObixClient R2 Schedule exports" section in the engineering notes document *Niagara R2 to NiagaraAX via oBIX*.