Technical Document

oBIX Guide – N4

August 6, 2022

niagara4

oBIX Guide – N4

Tridium, Inc.
3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

Trademark notice

Copyright and patent notice

Contents

About this guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

The Open Building Information Xchange (oBIX), based on RESTful Web Services, is an industry-wide standard for creating XML and web services that facilitate the exchange of information among intelligent buildings, enable enterprise-wide applications and integrate disparate systems. This guide documents the requirements, features and procedures for using the Niagara oBIX driver in building automation applications.

Document Change Log

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

August 06, 2022

Updated the document for 4.12 release.

December 15, 2019

Updated the document for 4.9 release.

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 **`Obix Schedule Manager`** 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.

February 15, 2008

Changed document to reference the *Niagara Drivers Guide*, a new document created from sections formerly in the *Getting Started with Niagara*.

Updated: June 25, 2007

Minor updates.

Updated: May 23, 2007

Generally minor updates. Throughout, mentioned driver support in AX-3.2.

Updated: April 18, 2007

Completely reworked as a single-source document, replacing the previous PDF-only *Niagara N4 oBIX User Guide*. Includes more details throughout, although more conceptual details will be added at 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.

Related documentation

Several documents provide additional information about this software.

- The *Niagara Drivers Guide* documents all properties shared in common with other drivers.
- The *Niagara Platform Guide* references the oBIX driver.

Chapter 1 oBIX driver

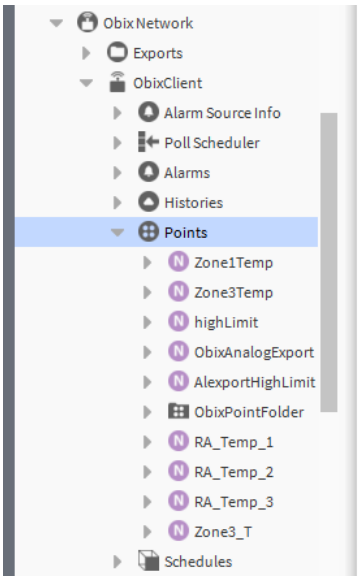
Topics covered in this chapter

♦ Compatibility
♦ License requirement
♦ Module requirements

oBIX uses the standard Framework network architecture. ObixClient components in the station interface to oBIX objects in one or more oBIX servers. oBIX proxy points under an ObixClient device, which, in turn, resides under an **ObixNetwork** container in the station's **Drivers** node, model real-time data.

Hierarchically, the component architecture is: network, device, points extension, points.

Figure 1 oBIX network in the Nav tree



The **ObixNetwork** is the top-level container component for "everything oBIX" in a station and resides under the station's **Drivers** container.

NOTE:

Only one **ObixNetwork** component is valid in a station regardless of how many oBIX servers a station may make a client connection to.

- The **ObixNetwork** component has the typical collection of slots and properties as do most other network components. One exception is the location of the network's poll components (**Poll Scheduler**), which is not at the network level, but under each **ObixClient** component.
- Like a few other drivers, oBIX client devices have a full range of device extensions—including **Points**, **Alarms**, **Histories**, and **Schedules**.
- The default view for the **ObixNetwork** is the **Obix Client Manager**. This view is equivalent to the Device Manager view in most other drivers. Other standard views are available to the network, however, apart from the **Obix Client Manager**, you typically access only each component's Property Sheet rather than its manager view.

Compatibility

The oBIX driver meets the standards defined by the "Committee Specification 1.0." using the specification document identifier of `obix-1.0-cs-01`.

Specification documents are currently found at OASIS using this URI: `http://www.oasis-open.org/committees/documents.php?wg_abbrev=obix`.

oBIX software functions on all host platforms running Niagara 4.

License requirement

To use the oBIX driver, you must have a target host controller or Supervisor PC that is licensed with the `obixDriver` feature. This provides the station with oBIX client operations.

To support an oBIX server, the `obixDriver` license must contain the attribute `export=true` and be licensed for the `web` feature.

For example:

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

The foreign limit arguments refer to limits on the number of foreign devices, histories, proxy points or schedules that may be part of the oBIX network. These limits apply only to external oBIX servers mapped as ObixClients that are not Framework stations. oBIX servers mapped to the **ObixNetwork** in a Framework station count under the regular limits with values of `none`, which means they are unlimited.

Module requirements

The obixDriver and obixSeriesTransform palettes contain the components used by the driver.

A station functioning as an oBIX server or client requires these modules:

- `obixDriver-rt.jar`
- `obixDriver-wb.jar`
- `obixMigrator-wb.jar`
- `obix-rt.jar`
- `obix-se.jar`
- `obixSeriesTransofrm-rt.jar`
- `web-rt.jar` (if the station will serve as an oBIX server)

Chapter 2 Installing, configuring and troubleshooting

Topics covered in this chapter

- Adding the Obix Network
- Adding ObixClient devices
- Independently verifying an oBIX server
- Adding oBIX proxy points
- Importing oBIX histories
- Enabling an oBIX server
- Exposing writable inputs for oBIX client access
- History queries from oBIX clients
- Troubleshooting

Like other drivers, you configure the oBIX driver features using Workbench manager views and property sheets.

Prerequisites: Your Supervisor or remote controller station is licensed to use oBIX and Workbench is connected to the station. The version of Workbench you are using was installed with the installation tool option (checkbox) set for

Step 1 If you need to install Workbench, make sure you enable the check box in the installation tool that indicates, "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 platforms. The .dist files are located under your `Niagara\sw` install directory.

Step 2 Confirm that the oBIX driver modules are available in the `C:\Niagara\[NiagaraVersionx.x.xx]\modules` folder, where `[NiagaraVersionx.x.xx]` is the version of the Framework.

Step 3 If the controller will operate as an oBIX server, make sure the `web` module is installed (and that feature is licensed).

Step 4 Upgrade any modules shown as out-of-date plus any modules shown as dependencies.

Following this, you are ready to configure the oBIX network as described in the rest of this document. As with other drivers, you configure driver features in Workbench using special manager views and Property Sheets. Manager views of oBIX device extensions add components, including proxy points, to the station. These views provide online discovery of available data items (Learn mode), which greatly simplifies the configuration process.

NOTE: The following procedures primarily use the manager views to add components. You may also open the obixDriver palette and drag components and folders from the palette to the driver property sheets.

Adding the Obix Network

Only one ObixNetwork is supported (or needed) in a station, regardless of whether you are using ObixClient or server functions, or both. This procedure adds an ObixNetwork component under the station's **Drivers** container.

Step 1 Expand **Config** and double-click the station's **Drivers** container.

The **Driver Manager** view opens.

Step 2 Click the **New** button.

The **New** network window opens.

Step 3 Select `ObixNetwork`, number to add: `1` and click **OK**.

This opens a window to name the network to something other than the default. A station may have only one ObixNetwork.

Step 4 Name the network or leave the default name and click **OK**.

The **ObixNetwork** under the **Drivers** folder should show a status of `{ok}` and **`Enabled`** as `true`.

Step 5 Configure properties and click **Save**.

The network's **Monitor** properties verify child oBIX client component(s). These are the "pingable" devices in the oBIX driver.

The **`Thread Pool`** property has special importance. It controls the number of threads used by the station to execute all actions of all oBIX objects in the network. This includes most communications with remote devices, which you can set up for multi-threading. In this case, if there are performance issues, you can increase the number of threads. The default value is four (4).

The network-level **Tuning Policies** slot with a single **Default Policy** container configures network-wide performance. You can add new tuning policies (duplicate and modify) as needed.

As in other driver networks, the network's **ObixClient** has an available **Alarm Source Info** container slot you can use to differentiate Obix alarms from other component alarms in the station.

As with most fieldbus drivers, the status of an ObixNetwork is either the normal `{ok}` or in `{fault}` (fault might result from licensing error). The **`Health`** property 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.

Adding ObixClient devices

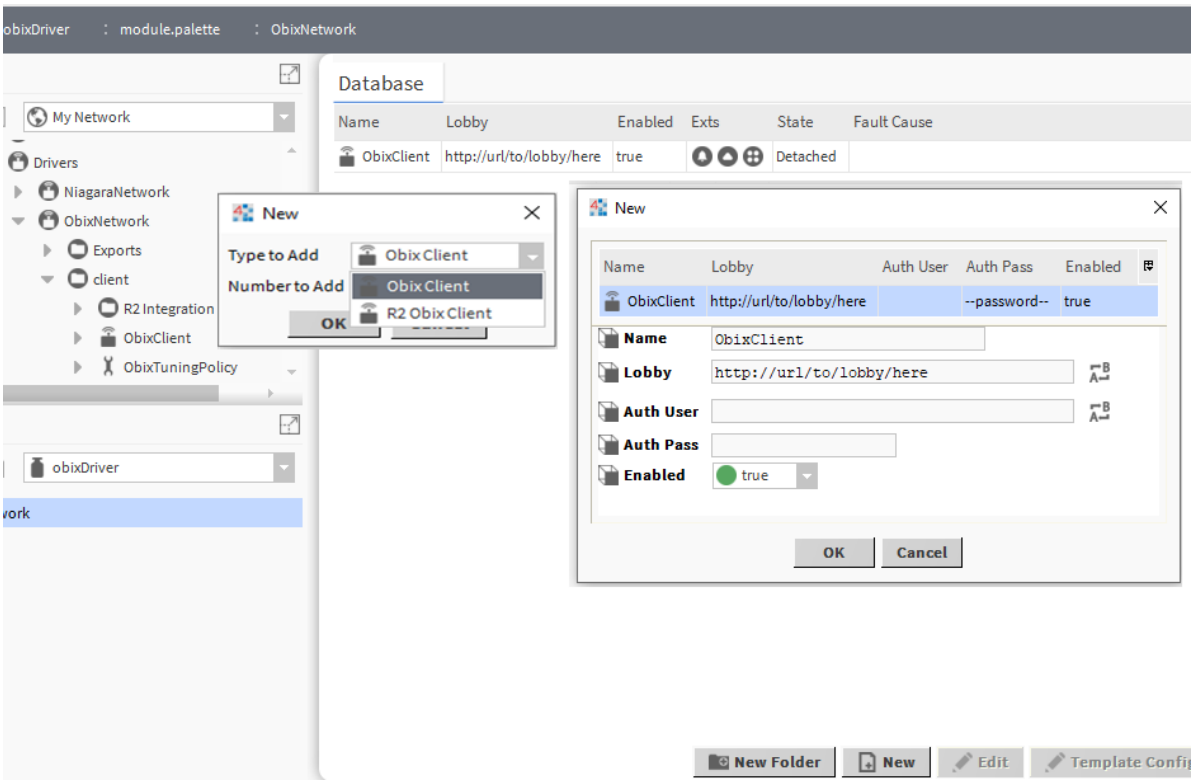
After adding an **ObixNetwork**, you can use the network's default **Client Manager** view to add the appropriate **`ObixClient`** and/or **`R2ObixClient`** devices. An **`ObixClient`** actually represents an oBIX server and associated Framework components represent client devices because the Framework uses a client connection to retrieve data.

Step 1 In the Nav tree or in the **Driver Manager** view, double-click the **ObixNetwork**.

The device manager (**Obix Client Manager**) opens.

Step 2 Click the **New** button.

The **New** device window opens.



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

This opens the window to name the device(s).

Step 4 Enter the **`Lobby`** URI, as well as the credentials needed for authentication (user name and password), and click **OK**.

The Framework adds the client device(s) to the database. You should see the device(s) listed in the **Client Manager** view with a State of `Detached` that changes to `Attaching` and then to `Attached`.

Step 5 If a device appears down with a state of `Detached`, double-click it in the **Client Manager** view and review the syntax of the **`Lobby`** URI and credentials.

Step 6 After making any device changes, click **Save**, then right-click the device and select **Actions→Ping**.

The device should report `Attached`.

Independently verifying an oBIX server

If your installation is licensed for operation as an oBIX server, the station can expose components and histories accessible to oBIX clients with access corresponding to the login credentials used for the connection. This includes allowing operation (op) writes on component actions. This procedure uses a web browser to verify that a host is operating as an oBIX server.

Prerequisites: You know the server lobby's URI, as well as its login credentials (if needed).

R2 integration does not support Niagara R2 to Niagara 4 operations because the R2 feature in the oBIX driver is server-side only. An R2 station cannot be an oBIX client.

Step 1 Open a browser and enter the following in the **`Address`** property: `http://[host][:port]/obix`

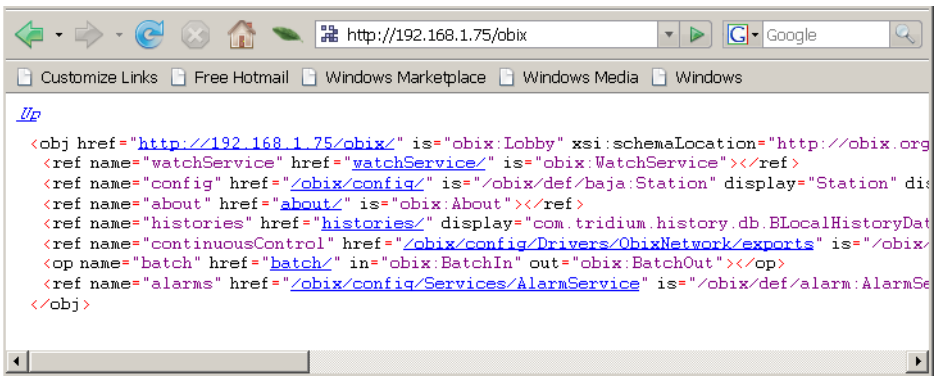
where `[host]` is the IP address or hostname of the server, and `[:port]` is optional (if omitted, the Framework assumes it to be port 80).

The station prompts you for a valid station user login, however, other oBIX servers may allow access without authentication.

NOTE: Niagara 4 requires basic digest authentication for browser clients to navigate to Obix server. Browser clients are restricted to use basic digest authentication, because of this browser clients are not able to access Obix server.

Step 2 Enter the station credentials and click **OK**.

You should see an HTML representation of the station's oBIX Lobby including hyperlinks to traverse the object tree structure.



Adding 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 **Obix Point Manager**. You use this view to add oBIX proxy points under any client device.

Prerequisites: You added a ClientObix device.

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 the discover job completes.

Step 1 Locate the row in the **Obix Client Manager** view that represents the device for which to create proxy points and double-click the Points icon ( ) under the Exts column.

The **Obix Point Manager** view opens.

Step 2 (Optional) To help organize points, create a new points folder by clicking the **New Folder** button and giving the folder a short name. Repeat this step to make multiple points folders or drag an ObixPointFolder from the palette.

If you skip this step, the Framework places all proxy points at the root of **Points** node.

All points folders have their own **Obix Point Manager** view, just like the **Points** node. If you make multiple points folders, double-click the folder to open its **Obix Point Manager** view.

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

An oBIX point discovery job launches, after which the expandable Lobby contains the learned objects of the **Discovered** pane.

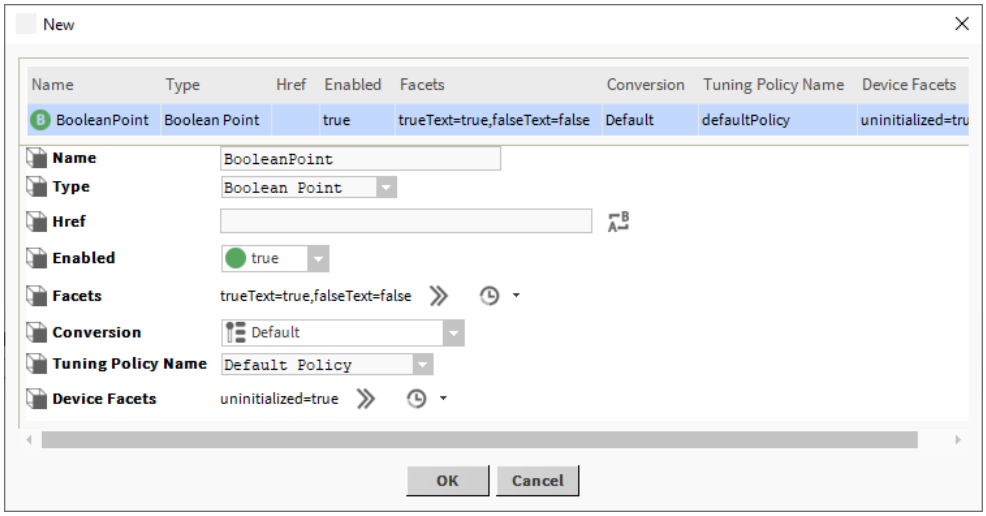
Step 4 To see the tree organization, expand the root Lobby.

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** window opens.



Step 6 Select a point **Name** and **Type**, review the other properties, change them if necessary, and 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.

Step 8 To edit an existing point, double-click one or more points.

The **Edit** window opens.

This window is similar to the **New** window used to create the points. Common properties to change are **Names** and **Facets**.

Importing oBIX histories

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

Prerequisites: You have added an ObixClient.

Step 1 Locate the row in the **Client Manager** view that represents the device for which import histories are available and double-click the Histories icon ( ) under the Exts column.

The **Obix History Manager** view opens.

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

An oBIX history discovery job launches, after which an expandable Lobby node appears in the **Discovered** pane. After the discovery job runs, all log objects and archives appear in the **Discoverd** pane. This includes log objects with identical names. However, by default their swid/href, which give them varying locations, give them unique names.

Step 3 To sort the discovered logs by name, click the "Obix Name" column in the **Discovered** pane.

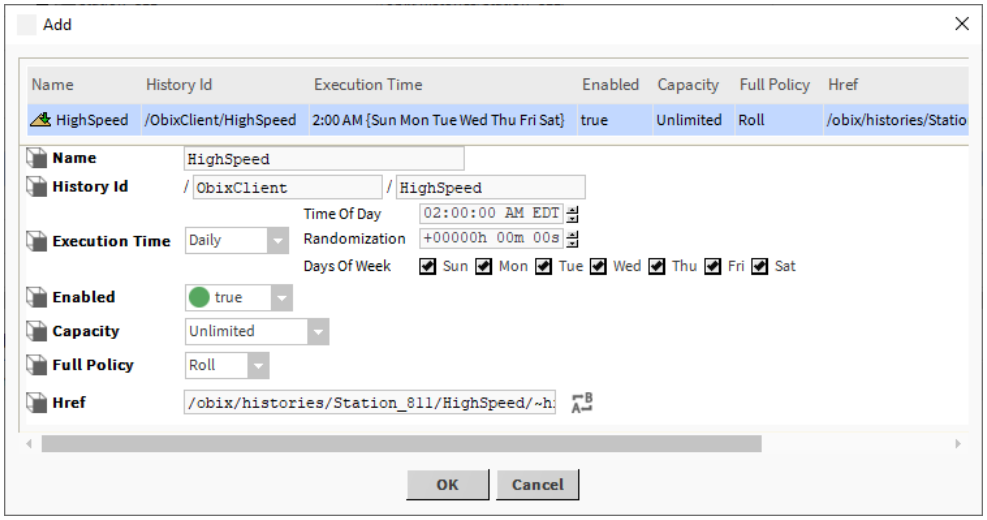
This groups identically-named log objects together.

Step 4 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 5 To add a history import descriptor, double-click a history.

The **Add** window opens.



Step 6 Enter its **`Name`** and **`History Id`**, as well as its **`Execution Schedule`** and (local) collection properties, and 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.

Step 7 To archive locally (create the local history), click to select one or more import descriptors, then click the **Archive** button.

Step 8 Continue to add history import descriptors as needed under the **Histories** extension of each client device.

Step 9 To edit one or more existing import descriptors, double-click the descriptor row in the table.

The **Edit** window opens.

This window is similar to the **New** window used to create the descriptors. Common properties to edit are **`Execution Times`**.

NOTE:

An additional device extension **Alarms** may also exist under an **ObixClient**.

Although you can create multiple import descriptors (with default values) for an identical Obix Name, only one can successfully import using the same **`History Id`**. Import descriptors with duplicate History Ids go into fault upon N import attempts. Therefore, by grouping you can select and edit History Ids appropriately when you add them to the database.

For example, if you have a station named "RN_Hall", with several logs each named "RmTemp", you could edit the second property of the History Id for each descriptor to make each unique: "Zn1_RmTemp", "Zn2_RmTemp", and so on. This way, complete History Ids for each would be "RN_Hall/Zn1_RmTemp", "RN_Hall/Zn2_RmTemp", and so forth.

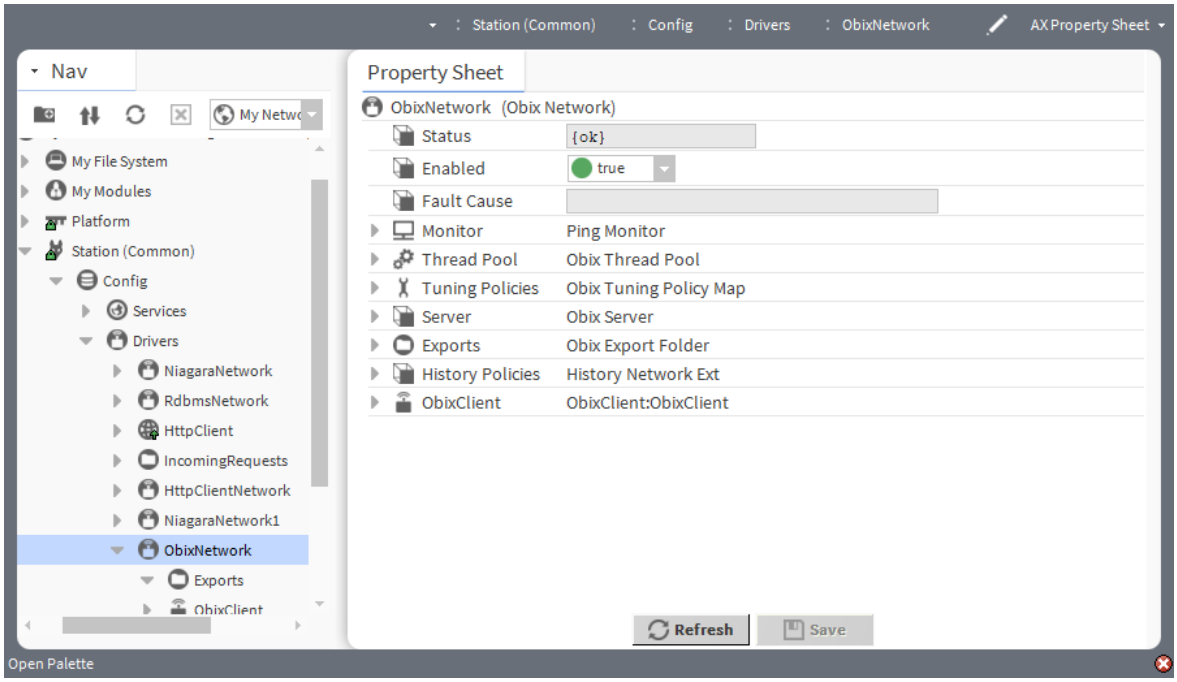
Enabling an oBIX server

When configured as an oBIX server, the station can expose components and histories accessible to oBIX clients, with access corresponding to the login credentials used for the connection. This includes allowing operation (op) writes on component actions.

Prerequisites: Your installation is licensed for an oBIX server.

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

The **Property Sheet** opens.



Step 2 Expand the **Server** slot and, set its **`Enabled`** property to `true` (if not already).

Step 3 Verify that the **`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`

where `[hostnameOrIP]` is the station name or IP address.

Exposing writable inputs for oBIX client access

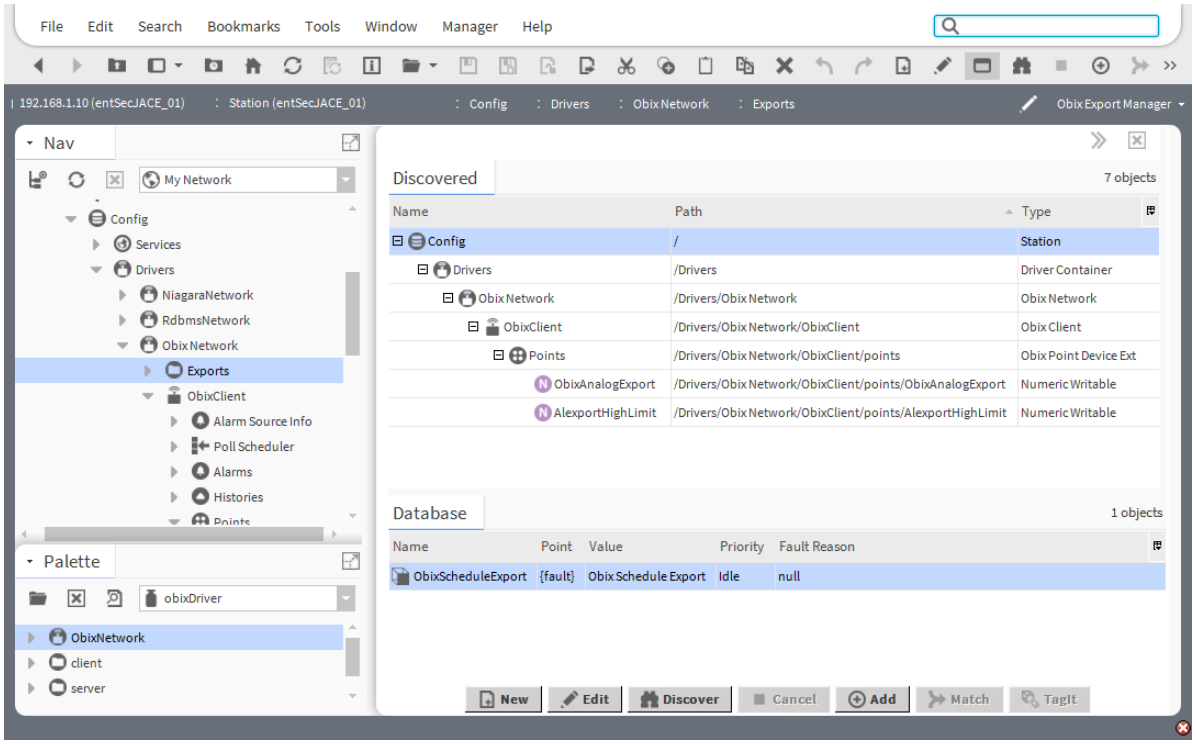
The ObixNetwork provides a mechanism to reserve specific inputs on any writable point (BooleanWritable, EnumWritable, NumericWritable, and StringWritable) in the station for op write access. Configuration is via the **Obix Export Manager** view on the **Exports** folder of the network.

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

The **Property Sheet** opens.

Step 2 Double-click the **Exports** folder ( ).

The **Obix Export Manager** view opens.



As shown above, a **Discover** button is available in which you discover local writable points in the station (BooleanWritable, EnumWritable, NumericWritable, and StringWritable).

Step 3 Click the **Discover** button.

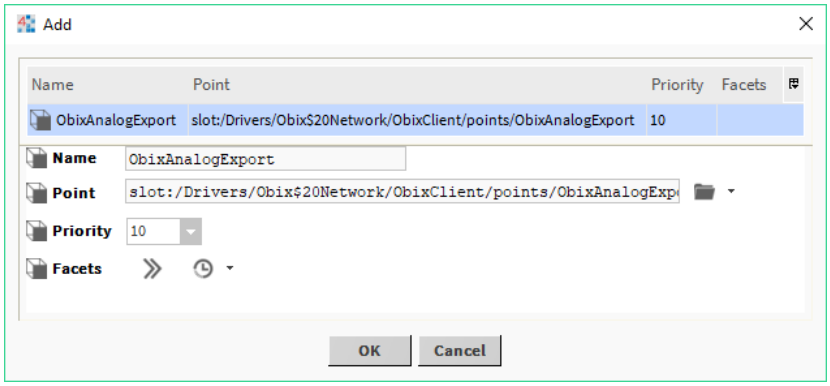
This opens the station's **Config** component tree in the **Discovered** pane, as an expandable node.

Step 4 In the **Discovered** pane of the **Obix Export Manager**, expand the component tree.

Currently, only those point types having priority input slots are valid candidates for adding. Consequently, this view shows only writable control points (other component types are filtered from view).

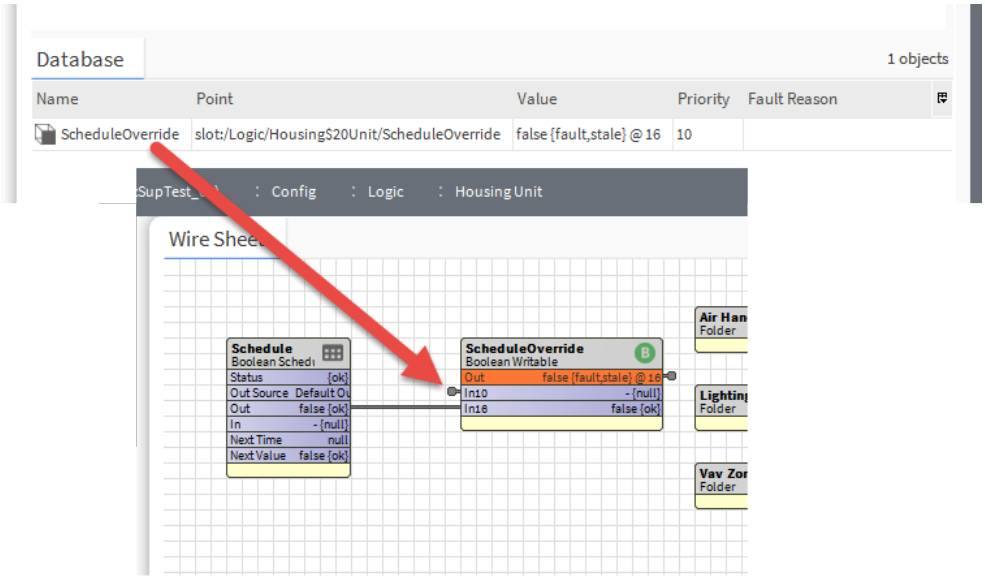
Step 5 To add an export descriptor, double-click a point.

The **Add** window opens.



Step 6 Configure **`Name`** and the **`Priority`** level of the points to expose to oBIX.

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. Points with the **`Priority`** (input) you define are reserved for oBIX clients.

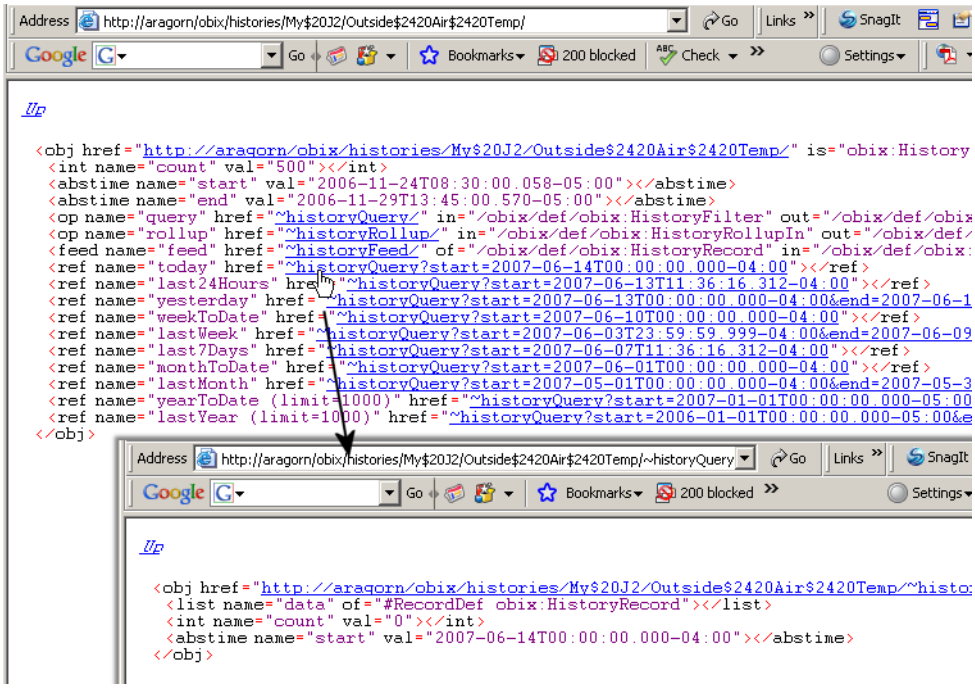


Adding this descriptor automatically creates a link (nub) on the target writable control point, such that link contention from within the Framework does not occur.

Step 7 Continue to add oBIX export descriptors as needed under the **Exports** folder of the **ObixNetwork**.

History queries from oBIX clients

Starting in obixDriver build 3.2.17 and later, histories exposed to oBIX clients provide a predefined set of query options, available as Lobby URIs using the HTTP GET mechanism (without an oBIX op). The available predefined 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.



For a few predefined history queries (Year to Date, Last Year), a default limit for number of records returned is used. If necessary, you can modify any of the standard history queries.

Troubleshooting

There are many reasons why oBIX network operations may fail.

A component indicates a status of "fault."

Occurs when a platform is not licensed for a specific feature. For example to use an oBIX network, the license must have the `export="true"` attribute in the `obixDriver` feature line.

A component reports a fault cause.

If status is `fault`, this property explains why, such as `Server not licensed`.

When adding an oBIX device, the State of the device remains "Detached."

- Review the syntax of the object's Lobby, including the IP address of any R2 controller.
- Open a command prompt window and ping the IP address
- Verify that any R2 station's user credentials for Auth User and Auth Pass are correct.

You should be able to open a browser connection to an R2 station using the URI entered for **`Lobby`**, and after entering user credentials, see its oBIX Lobby.

NOTE: Niagara 4 requires basic digest authentication for browser clients to navigate to Obix server. Browser clients are restricted to use basic digest authentication, because of this browser clients are not able to access Obix server.

For Browser Clients

NOTE: Niagara 4 requires basic digest authentication for browser clients to navigate to Obix server. Browser clients are restricted to use basic digest authentication, because of this browser clients are not able to access Obix server.

Chapter 3 Components

Topics covered in this chapter

- obixDriver- ObixAlarmDeviceExt
- obixDriver-ObixAlarmImport
- obixDriver-ObixClient
- obixDriver-ObixClientFolder
- obixDriver-ObixExport
- obixDriver-ObixExportFolder
- obixDriver-ObixHistoryDeviceExt
- obixDriver-ObixHistoryImport
- obixDriver-ObixNetwork
- obixDriver-ObixOp
- obixDriver-ObixPointDeviceExt
- obixDriver-ObixPointFolder
- obixDriver-ObixPollScheduler
- obixDriver-ObixProxyExt
- obixDriver-ObixScheduleExport
- obixDriver-ObixServer
- obixDriver-ObixThreadPool
- obixDriver-OBixTuningPolicy
- obixDriver-ObixTuningPolicyMap
- obixDriver-R2AlarmImport
- obixDriver-R2ObixClient

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette.

Descriptions included in the following topics appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

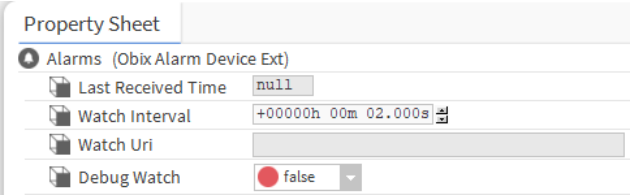
obixDriver- ObixAlarmDeviceExt

This frozen device extension of the **ObixClient** component integrates its native alarms into the Framework alarming subsystem. Its default view is the **Obix Alarm Manager**.

This component does not configure alarms. An oBIX server can have many alarm sources. The **ObixAlarmImport** object configures properties, such as **Alarm Class**.

The **ObixAlarmDeviceExt** creates and manages a single monitor (watch) used for all alarm feeds. The properties on the alarms **Property Sheet** configure this watch.

Figure 2 Alarm device extension properties



One way to access these properties, expand **Config→Drivers→ObixNetwork→ObixClient**, right-click **Alarms** and click **Views→AX Property Sheet**.

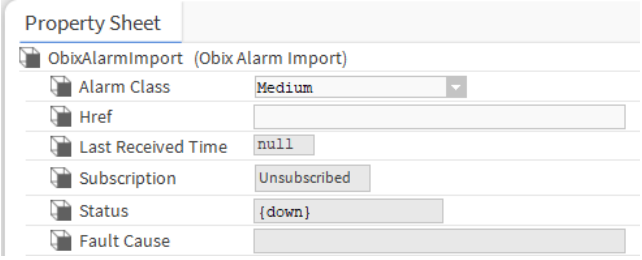
The **ObixAlarmImport** object configures alarm properties, such as alarm class.

Property	Value	Description
Last Received Time	read-only	Reports the last time the station received an alarm from the device.
Watch Interval	hours minutes seconds	Controls the polling of the watch subscription (how often the client should poll the alarm watch). To reduce the load on the-platform, increase this value to 10 seconds or more.
Watch Uri	read-only	Defines the Universal Resource Identifier for the watch.
Debug Watch	`true` or `false` (default)	Enables and disables the use of the watch for this **ObixClient**.

obixDriver-ObixAlarmImport

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

Figure 3 Obix Alarm Import properties



To access these properties, locate the component in the Nav tree and double-click it. It could be in any number of places.

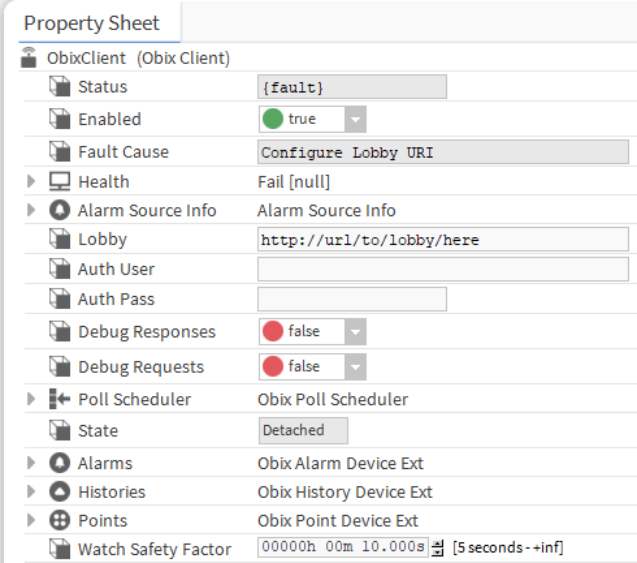
In addition to the standard properties (Status and Fault Cause), these are unique or have special importance to the **ObixAlarmImport**:

Property	Value	Description
Alarm Class	drop-down list	Provides a list of local Alarm Classes from which to select one to use for all alarms received from this alarm subject.
Href	text	Defines the URI of the alarm feed on the server. This value must be unique among all the**ObixAlarmImport** of any given **ObixClient**.
Last Received Time	read-only	Reports the last time the component received an alarm.
Subscription	read-only	Indicates if the component is subscribed.

obixDriver-ObixClient

This component represents the client access to an oBIX server device. A special-purpose variation for a Niagara R2 station also exists as the **R2ObixClient**. Each is a device-level component in theNiagara 4 oBIX driver architecture.

Figure 4 ObixClient properties



One way to access these properties is to right-click the **ObixNetwork**, click **Views→AX Property Sheet**, and expand or double-click **ObixClient**.

As with other driver networks, an oBIX network has an **Alarm Source Info** container you can use to differentiate oBIX network alarms from other component alarms in the station.

In addition to the standard components (Status, Enabled Fault Cause, Health, Alarm Source Info), these properties are unique or have special importance:

Property	Value	Description
Lobby	URL-style text	Defines the URI at the root of the server's object tree. If the server host changes, only the authority (scheme:`//host[:port]`) needs to change and all sub-objects will work for the new host.
Auth User	text	Defines the user name the client should use to access the server. It can be blank if the server supports unauthenticated access.
Auth Pass	text	Defines the passphrase for the **`Auth User`**.
Debug Responses	`true` and `false` (default)	Enables and disables responses from the server to client requests.
Debug Requests	`true` and `false` (default)	Enables and disables client requests to the server.
Poll Scheduler	additional properties	Defines the client polling schedule only when watches on the server are not working.
State	read-only	Detached
Alarms	additional properties	Integrates native alarms from the ObixClient into the Framework alarming subsystem.
Histories	additional properties	Opens the **Obix History Manager** view.

Property	Value	Description
Points	additional properties	Opens the **Obix Points Manager** view.
Watch Safety Factor	hours minutes seconds (defaults to 10 seconds)	Specifies the time added to the watch interval to calculate the requested lease time of the watch. This is the change-of-value (CoV) subscription lifetime that the client requests from the server. This is more important if using a shorter watch interval, because the driver calculates two values to use as the requested lease time, choosing the larger of the two: `T1 = watch interval * 2T2 = watch interval + watch safety factor` For large watch intervals, the requested lease time is inherently big (2x). But the safety factor provides an independent way to control requested lease time, such that you could make it 3 or 4 times the watch interval, if desired.

Actions

The **`ObixClient`**supports two actions:

- `Ping` sends a ping monitor request to verify device health.
- `Reattach` disconnects from the oBIX server and attempts to connect again.

obixDriver-ObixClientFolder

This component is the oBIX driver implementation of a folder under an **ObixNetwork**. Usage is optional. Each**`ObixClientFolder`** has its own **Obix Client Manager** view.

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

obixDriver-ObixExport

This component is a child of an **ObixExportFolder**. It is an oBIX server that provides link-write access to remote client control points. While many Framework points allow oBIX clients to write a value to them (each action appears as an op), an**`ObixExport`** defines a link to the input property of a specific Framework point, at a particular control level.

Figure 5 Obix Export properties

One way to access these properties is to expand the **ObixNetwork** and double-click the **ObixExport** node in the Nav tree.

Property	Value	Description
Point	ORD	Defines the location of a control point. Setting this value configures the export automatically.
Priority	drop-down list	Defines when to link the control point. When changed, the driver the export object again automatically.
Facets	Ref chooser	Defines control point facets. These should always be a copy of the facets of the target control point, which the driver configures automatically.
Status	read-only	Reports the health of this manager view. It will only ever be `ok` or `fault` and does not affect the value read by oBIX clients. `fault` indicates that a point does not reference a control point or that something is already linked into the control point at the priority level specified in the **`Priority`** property.
Fault Reason	read-only	Explains why an export descriptor is in fault. Typically, this reflects an invalid configuration, such as selecting a **`Priority`** level that is already linked on the target control point.
WritePoint	additional properties	This is for oBIX encoding, and should not be modified.

obixDriver-ObixExportFolder

This component (default name Exports) is a frozen container slot under an Obix Network. It simplifies the addition and management of **`oBIX export`** objects and is not limited to containing only oBIX objects. You can duplicate this folder as needed with no restrictions on where and how many you can have in a station.

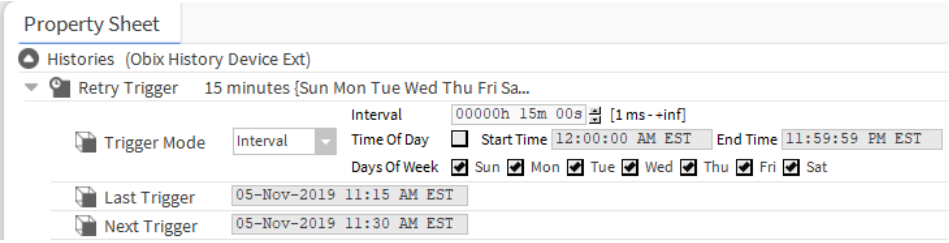
Each folder has its own **Obix Export Manager** view. Although you can view the manager's Property Sheet, it contains no properties to configure.

You add a folder to your station by dragging it from the palette. You access its contents by expanding it in the Nav tree or double-clicking on it.

obixDriver-ObixHistoryDeviceExt

This component discovers, adds, and manages history imports for the **`ObixClient`** and **`R2ObixClient`**. It manages the import of historical data from the oBIX server into the Niagara 4 history space. Use its default **Obix History Manager** view to add **`oBIX history import`**descriptors.

Figure 6 History device extension properties



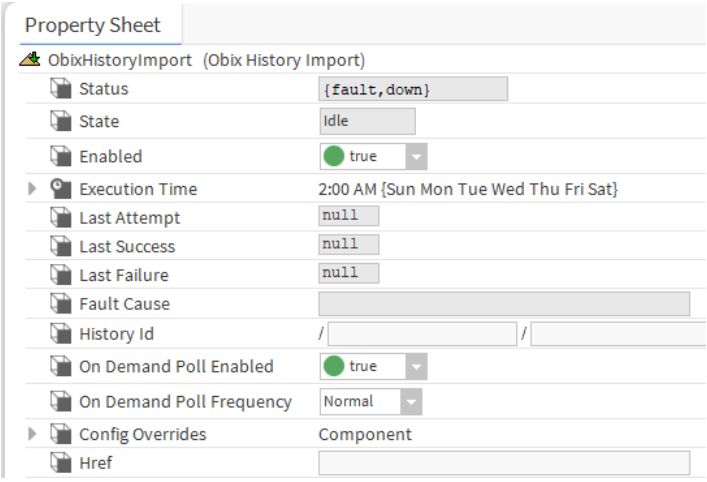
To access these properties expand **ObixNetwork→ObixClient**, right-click **Histories** and click **Views→AX Property Sheet**.

Property	Value	Description
Retry Trigger	container	Defines how many times the retry trigger automatically tries to execute a descriptor component following an unsuccessful attempt. The retry action executes a retry on any descriptors whose status is in fault. This continues until successful execution occurs.
Trigger Mode	drop-down list	Determines when a TimeTrigger fires. `Interval` fires a trigger each time the specified interval elapses. You would use it to fire the trigger several times per day (for example, every 5 minutes). `Daily` fires the trigger at a specific time on selected days of the week, and includes a randomized interval so that the trigger does not fire at exactly the same time every day. `Manual` requires a human to fire a trigger.
Last Trigger	read-only	Reports when (by displaying a timestamp) the last trigger fired.
Next Trigger	read-only	Reports when the trigger is scheduled to fire next.

obixDriver-ObixHistoryImport

This component is a history-import-descriptor child of the**`ObixHistoryDeviceExt`** (**Histories** extension under an **`ObixClient`**). It corresponds to a history (log or trend) on the oBIX server.

Figure 7 Obix History Import properties



One way to access these properties is to expand the **Config→Drivers→ObixNetwork→ObixClient→Histories** node in the Nav tree and double-clicking the **ObixHistoryImport** component in the Nav tree.

In addition to the standard properties (Status, State, Enabled and Fault Cause), these properties apply to history imports:

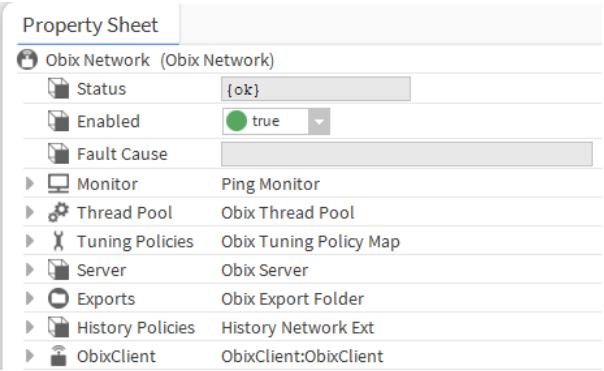
Property	Value	Description
Execution Time	additional standard properties	Controls when to invoke an input based on an interval of time, daily at a specific time, or manually by action execution.
Last Attempt	read-only, null (default)	Reports the timestamp of the last attempted history import.
Last Success	read-only, null (default)	Reports the timestamp of the last successful history import.
Last Failure	read-only, null (default)	Reports the timestamp of the last attempted history import failure (could not complete).
History ID	text separated by / defaults to `[name of ObixClient] / [oBIX history name]` Where `[name of ObixClient]` identifies the Obix-Client, and `[oBIX history name]` identifies the history.	Creates an ID for the history created by the import descriptor.
On Demand Poll Enabled	`true` (default) and `false`	Enables and disables the polling of oBIX histories. Applies to the associated imported histories; operates the same as for regular HistoryImport descriptors. You may disable this polling to limit bandwidth usage.
Config Overrides	additional properties	These properties (capacity, fullPolicy) work as they do in the History Config properties of a normal history extension.
Href	URL-type text	Defines the URI to the history on the oBIX server.

obixDriver-ObixNetwork

This component 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 Framework convention, and has no physical correspondence to any oBIX system.

The **Obix Client Manager** is the default view of the **ObixNetwork**.

Figure 8 Obix Network properties



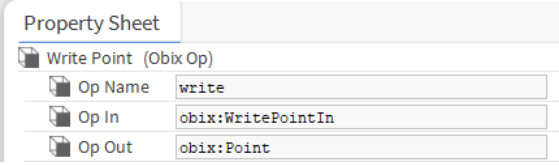
You access these properties by right-clicking the **Obix Network** in the Nav tree followed by clicking **Views→AX Property Sheet**.

Property	Value	Description
Status	read-only	Indicates the condition of the network, device or component at the last check. `{ok}` indicates that the component is licensed and polling successfully. `{down}` indicates that the last check was unsuccessful, perhaps because of an incorrect property, or possibly loss of network connection. `{disabled}` indicates that the **`Enable`** property is set to `false`. `{fault}` indicates another problem. Refer to **`Fault Cause`** for more information.
Enabled	`true` (default) or `false`	Activates (`true`) and deactivates (`false`) the object (network, device, point, component, table, schedule, descriptor, etc.). If `true` (the default), remote oBIX clients can access the lobby of the station following login using station user credentials (provided that the 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.
Fault Cause	read-only	Indicates the reason why a system object (network, device, component, extension, etc.) is not working properly (in fault). This property is empty unless a fault exists.
Monitor	additional properties	Configures a network's ping mechanism, which verifies network health. This includes verifying the health of all connected objects (typically, devices) by pinging each device at a repeated interval. The ObixNetwork's monitor properties verify child client component(s), which are the devices in the oBIX driver that are able to receive a ping.
Thread Pool	additional property	Serves as a container for the **`Max Threads`** property.
Server	additional properties	Identifies a frozen container slot under the **ObixNetwork**.
Exports	no properties	Serves as a folder for data exports.
History Policies	additional properties	This extension serves as a container for the poll scheduler and default configuration rules.
`Obix Client`	additional properties	Represents client access to an **Obix Network**(server) device.

obixDriver-ObixOp

These properties configure oBIX operations. Do not modify them.

Figure 9 Obix Op properties



obixDriver-ObixPointDeviceExt

This component (default name **Points**) is the container for Obix proxy points under an **ObixClient**. It creates and manages a single watch used for all points that are currently read-subscribed.

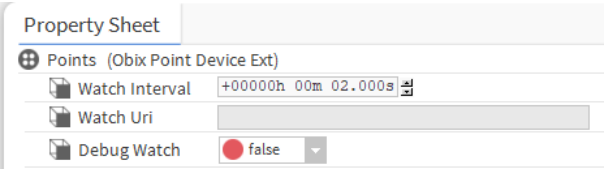
The default and primary view for the Points extension is the **Obix Point Manager**.

NOTE: The **`R2ObixClient`** and **`R2PointDeviceExt`** use different points extensions with additional properties related to the discovery of R2 objects.

A watch is not really change–of–value function. 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 Niagara 4client 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).

Figure 10 Obix Point Device Extension properties



To access these properties, right-click the **Obix Network** in the Nav tree, click **Views→AX Property Sheet**, expand or double-click the **ObixClient**, and expand **Points**.

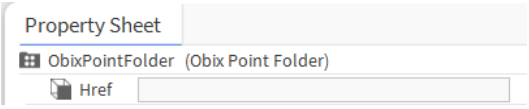
Three properties configure the watch.

Property	Value	Description
Watch Interval	hours minutes seconds (defaults to 2 seconds)	Controls the polling of the watch subscription (how often the client should poll the alarm watch). To reduce the load on the-platform, increase this value to 10 seconds or more.
Watch Uri	read-only	Defines the Universal Resource Identifier for the watch.
Debug Watch	`true` or `false` (default)	Enables and disables the use of the watch for this **`ObixClient`**.

obixDriver-ObixPointFolder

This optional container organizes oBIX proxy points. Its organization may (or may not) mirror the organizational structure within the underlying oBIX server. Points can be organized in any fashion under the **Points** device extension (**`ObixPointDeviceExt`**).

Figure 11 Obix Point Folder property



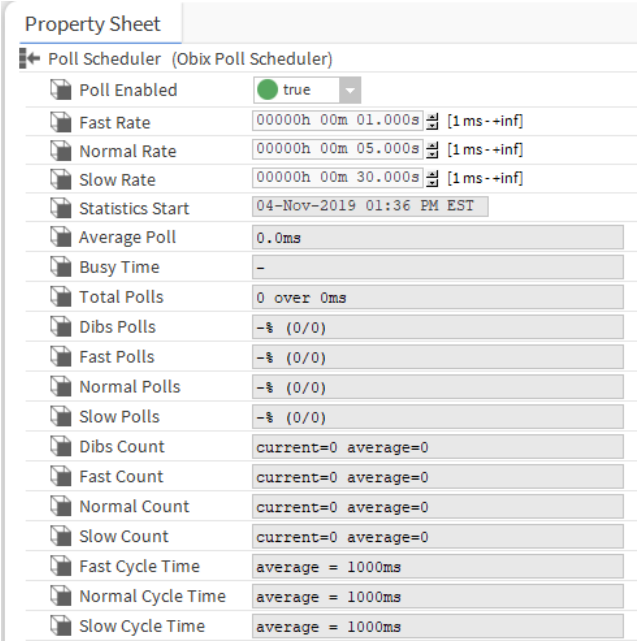
You can use the **New Folder** button in the **Obix Point** view to add an **ObixPointFolder**. It is also available in the **obixDriver** palette. Each **ObixPointFolder**has its own **Obix Point Manager** view.

Property	Value	Description
Href	URL-style text	Defines the URI to the oBIX point on the server. This value must be unique among all the points of any given **ObixClient** or **R2ObixClient**. The driver automatically learns each point during discovery.

obixDriver-ObixPollScheduler

This component is a child component of every device-level **ObixClient**. It provides a flexible polling algorithm based on four polling buckets.

Figure 12 Poll Scheduler properties



One way to access these properties, expand **Config→Drivers→ObixNetwork→ObixClient**, right-click **Alarms** and double-click **Poll Scheduler** in the Nav tree.

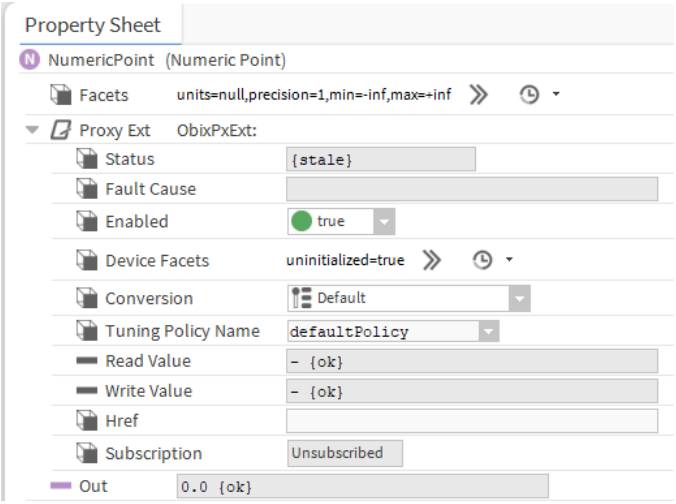
Property	Value	Description
Poll Enabled	`true` (default) or `false`	Enables or disables the Poll Scheduler.
Fast Rate	`hh-mm-ss, 1ms ... +inf` Defaults to 00–00–01	Sets the target polling interval for pollables assigned to this rate group.
Normal Rate	`hh-mm-ss, 1ms ... +inf` Defaults to 00–00–05	Sets the target polling interval for pollables assigned to this rate group.

Property	Value	Description
Slow Rate	`hh-mm-ss, 1ms ... +inf` Defaults to 00–00–30	Sets the target polling interval for pollables assigned to this rate group.
Statistics Start	read-only `date time`	Reports the last time statistics were reset.
Average Poll	read-only `time`	Reports the average time spent in each poll.
Busy Time	read-only `time`	Reports the percentage of the time that the station was busy doing polls.
Total Polls	read-only `Numeric` over `time`	Reports the total number of polls conducted and the time spent waiting for polls to execute.
Dibs Polls	read-only `Numeric` % (`Numeric`/ `Numeric`)	Reports the percentage and ratio of the number of DIBS polls versus total polls.
Fast Polls	read-only `Numeric` % (`Numeric`/ `Numeric`)	Reports the total number of polls made processing the fast queue.
Normal Polls	read-only `Numeric` % (`Numeric`/ `Numeric`)	Reports the total number of polls made processing the normal queue.
Slow Polls	read-only `Numeric` % (`Numeric`/ `Numeric`)	Reports the total number of polls made processing the slow queue.
Dibs Count	read-only current= `numeric` average= `numeric`	Reports the current and average number of components in the dibs stack. (DIBS stands for Distributed Internet Backup System).
Fast Count	read-only current= `numeric` average= `numeric`	Reports the current and average number of components in the fast queue.
Normal Count	read-only current= `numeric` average= `numeric`	Reports the current and average number of components in the normal queue.
Slow Count	read-only current= `numeric` average= `numeric`	Reports the current and average number of components in the slow queue.
Fast Cycle Time	read-only average = `time`	Reports the average cycle time for the fast queue.
Normal Cycle Time	read-only average = `time`	Reports the average cycle time for the normal queue.
Slow Cycle Time	read-only average = `time`	Reports the average cycle time for the slow queue.

obixDriver-ObixProxyExt

This component is the proxy extension for any type of Obix proxy point.

Figure 13 Obix Numeric Point proxy extension properties



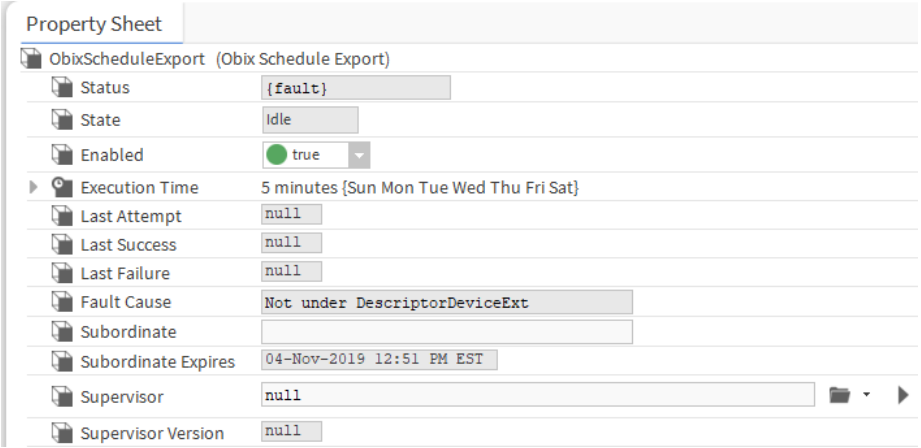
In addition to the standard properties (Status and Enabled) among others, these properties are unique or have special importance:

Property	Value	Description
Href	URL-style text	Defines the URI to the oBIX point on the server. This value must be unique among all the points of any given **ObixClient**or **R2ObixClient**. The driver automatically learns each point during discovery.
Force Update		Forces a read, updates the dynamic actions on the control point and updates the device facets in the **ObixProxyExt**. If the point is supposed to be subscribed but is not, this update attempts to re-subscribe the point.

obixDriver-ObixScheduleExport

This component is a schedule-export-descriptor child of an **R2ScheduleDeviceExt** (Schedules extension of an **R2ObixClient**. It corresponds to a target schedule in a Niagara R2 station to receive events from a local schedule component.

Figure 14 Obix Schedule Export properties



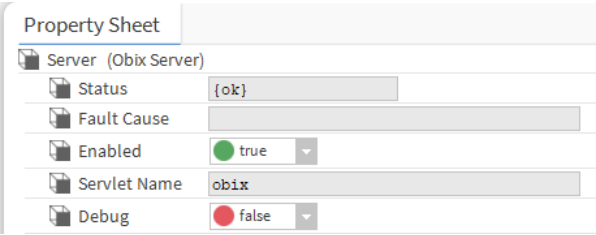
In addition to the standard properties (Status, Enabled and Fault Cause), these properties support export schedules:

Property	Value	Description
State	read-only	Displays the current state
Execution Time	additional standard properties	Controls when to invoke an export based on an interval of time, daily at a specific time, or manually by action execution.
Last Attempt	read-only, null (default)	Reports the timestamp of the last attempted schedule export.
Last Success	read-only, null (default)	Reports the timestamp of the last successful schedule export.
Last Failure	read-only, null (default)	Reports the timestamp of the last failed schedule export.
Subordinate	URL-style text	Identifies the URI of the target schedule on the oBIX server in an R2 station.
Subordinate Expires	read-only	Displays the timestamp of when "mastering" of target oBIX schedule expires, assuming no more executions.
Supervisor	text	Specifies which local AX BooleanSchedule will serve as the supervisor (master) schedule for a target R2 schedule.
Supervisor Version	read-only	Displays the version.

obixDriver-ObixServer

This component is a frozen container slot under the **Obix Network**.

Figure 15 Server properties



You access this property by right-clicking the **Obix Network** in the Nav tree followed by clicking **Views→AX Property Sheet** and expanding or double-clicking the **Server** container.

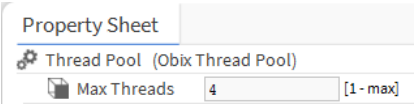
In addition to the standard properties (Status, Fault Cause and Enabled) these properties are unique to this component:

Property	Value	Description
Servlet Name	read-only	Currently fixed at: `obix`.
Debug	`true` or `false` (default)	Controls the printing of debug information to the station's standard output. `true` enables the output for incoming requests from Obix-Clients, as well as the server's outgoing responses. `false` disables the output.

obixDriver-ObixThreadPool

This is a frozen slot under the ObixNetwork.

Figure 16 Obix Thread Pool property



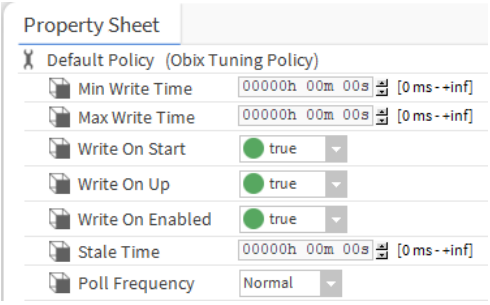
You access this property by right-clicking the **Obix Network** in the Nav tree followed by clicking **Views→AX Property Sheet** and expanding or double-clicking the **Thread Pool** container.

Property	Value	Description
Max Threads	number (defaults to 4)	Tunes large networks (those with many station components) to process more than a single thread at a time. It is the only visible part of a shared thread-pool scheme for large-job scalablilty and allows the local station's thread pool to grow uncapped. If your **`ObixNetwork`**has performance issues, increase the number of threads.

obixDriver-OBixTuningPolicy

This component provides a tuning policy for the oBIX network, with standard tuning policy properties.

Figure 17 Obix Network default tuning policy properties



Property	Value	Description
Min Write Time	hours minutes seconds	Specifies the minimum amount of time allowed between writes to writable proxy points, especially ones that have one or more linked inputs. This provides a way to throttle rapidly changing values so that only the last value is written. The default value (0) disables this rule causing all value changes to attempt to write.
Max Write Time	hours minutes seconds	Specifies the maximum amount of time to wait before rewriting the value, in case nothing else has triggered a write, to writable proxy points. Any write action resets this timer. The default (0) disables this rule resulting in no timed rewrites.
Write On Start	`true` (default) or `false`	Defines a writeable proxy point's behavior when its status transitions from disabled to enabled. `true` initiates a write when the transition occurs.

Property	Value	Description
		`false` prevents a write when the transition occurs.
Write On Up	`true` (default) or `false`	Defines a writeable proxy point's behavior at station startup. `true` initiates a write when the station first reaches a steady state. `false` prevents a write when the station first reaches a steady state.
Write On Enabled	`true` (default) or `false`	Defines a writeable proxy point's behavior when the point and its parent device transition from down to up. `true` initiates a write when the transition occurs. `false` prevents a write when the transition occurs.
Stale Time	hours minutes seconds	Defines the period of time without a successful read (indicated by a read status of {ok}) after which a point's value is considered to be too old to be meaningful (stale). A non-zero value causes the point to become stale (status stale) if the configured time elapses without a successful read, indicated by Read Status `{ok}`. The default value (zero) disables the stale timer causing points to become stale immediately when unsubscribed.
Poll Frequency	drop-down list (defaults to `Normal`)	Selects among three rates (Fast, Normal and Slow) to determine how often to query the component for its value. The network's Poll Service or Poll Scheduler defines these rates in hours, minutes and seconds. For example: `Fast` may set polling frequency to every second. `Normal` may set poll frequency to every five seconds. `Slow` may set poll frequency to every 30 seconds. This property applies to all proxy points.

obixDriver-ObixTuningPolicyMap

This component is 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.

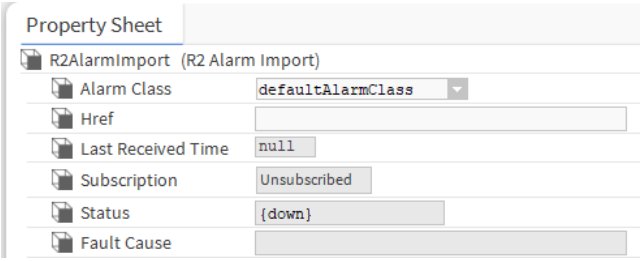
As a container, this component has no properties of its own.

obixDriver-R2AlarmImport

This component is a special variant of the **`ObixAlarmDeviceExt`** component. As a frozen device extension under an **`R2ObixClient`**, it represents an oBIX alarm feed from the Niagara R2station. Typically, you create one **`R2AlarmImport`** for each discovered R2 **NotificationClass** node under the NotificationService area of the server's Lobby.

This component's default view is the **Obix Alarm Manager**.

Figure 18 R2 Alarm Import properties



In addition to the standard properties (Status and Fault Cause), these properties support the R2 alarm import feature

Property	Value	Description
Alarm Class	drop-down list	Lists local Alarm Classes, from to select one to use for all alarms received from this alarm subject.
Href	URL-style text	Defines the URI of the alarm feed on the server. This value must be unique among all the **ObixAlarmImports**of any given **ObixClient**.
Last Received Time	read-only	Reports the last time the component received an alarm.
Subscription	read-only	Indicates if the component is subscribed.

obixDriver-R2ObixClient

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

The **R2ObixClient** has the standard device component properties, such as **Status** and **Enabled** and supports the same actions as those documented for the **ObixClient**.

Two actions are available on the **R2ObixClient**, as follows:

- `Ping` sends a ping monitor request to verify the health of the device.
- `Reattach` disconnects the client from the oBIX server and attempts to connect it again.

Chapter 4 Plugins (views)

Topics covered in this chapter

- obixDriver-ObixAlarmManager
- obixDriver-ObixClientManager
- obixDriver-ObixExportManager
- obixDriver-ObixHistoryManager
- obixDriver-ObixPointManager

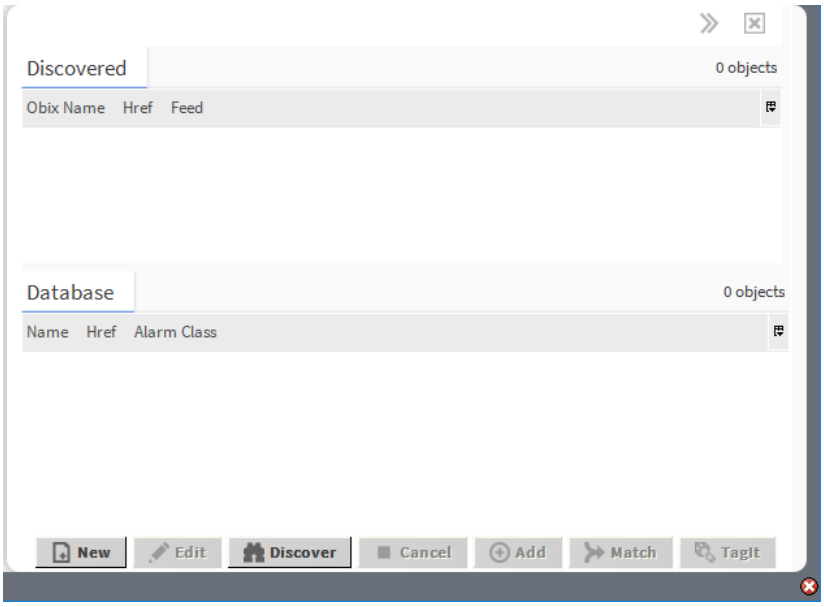
Plugins provide views of components and can be accessed in many ways. For example, double-click a component in the Nav tree to see its default view. In addition, you can right-click on a component and select from its **Views** menu.

For summary documentation on any view, select **Help→On View** (**F1**) from the menu or press **F1** while the view is open.

obixDriver-ObixAlarmManager

This view discovers, adds, and manages alarm feeds from a selected **`ObixClient`** or **`R2ObixClient`**. It is the default view of the **Alarms** device extension (**`ObixAlarmDeviceExt`**, **`R2AlarmDeviceExt`**) under these devices.

Figure 19 Obix Alarm Manager view



To view, double-click the **Alarms** extension under the **Obix Network→ObixClient** node in the Nav tree, or right-click and select **Views→Obix Alarm Manager**.

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

Discovered pane

By default, these columns appear in the **Discovered** and **Database** panes of the **Obix Alarm Manager** view.

Table 1 Discovered pane columns

Column	Description
Obix Name	Reports the name of the object on the oBIX server.
Href	Reports the URI of the object on the oBIX server.
Feed	Reports the lobby URI of the object on the oBIX server, with `.alarm` suffix to denote a valid alarm feed.

NOTE:

Discovery caches results. 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.

Database pane

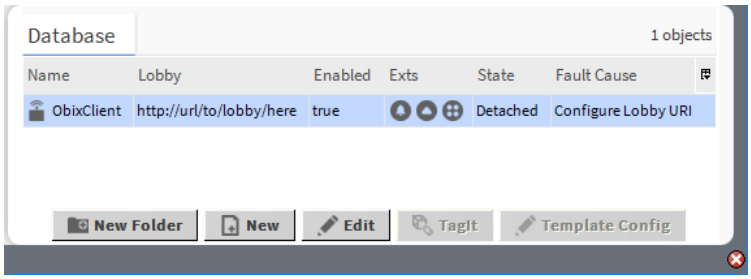
Table 2 Database pane columns

Column	Description
Name	Reports the name of the alarm import descriptor, often left the same as its oBIX Name.
Href	Reports the URI of the object on the oBIX server.
Alarm Class	Reports the local station's AlarmClass used to process native alarms received via this alarm feed.
Subscription	Reflects whether the alarm import is Subscribed, Unsubscribed, or Pending subscription.

obixDriver-ObixClientManager

This view adds, edits, and accesses oBIX device components (**`ObixClients`** and **`R2ObixClients`**). It is the default view of an **Obix Network**. It is equivalent to the **Device Manager** view in most other drivers.

Figure 20 Obix Client Manager view



To access this view, double-click the **Obix Network** node in the Nav tree.

NOTE:

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

Added devices appear in the **Database** pane.

By default, these columns appear in the **Discovered** pane of the **Obix Client Manager** view:

Column	Description
Name	Reports the name of the device-level component for (client) interface to the oBIX server.
Lobby	Reports the URI to root of the server's object tree, using format: `http://[hostName or IP address]/obix`, where `[hostName or IP address]` is the station's host name or IP address.
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	Describes the cause of the client device status fault, if any.
Path	When configured, reports the station path of the device-level component, relative to the root.
Auth User	When configured, reports the name of the authorized user for client access.
Auth Pass	When configured, identifies the passphrase (password) of the authorized user.
Type	When configured, reports the device-level component type, currently ObixClient.

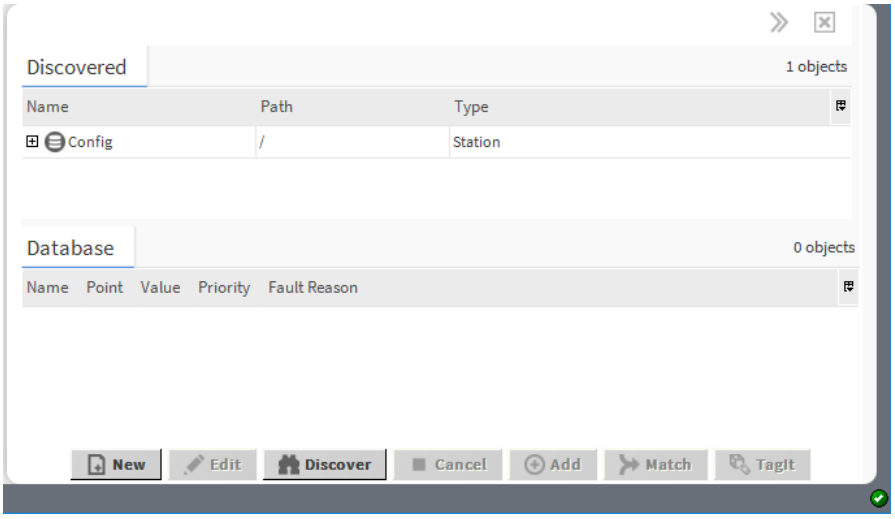
obixDriver-ObixExportManager

This view discovers, adds, and manages 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). It is the default view of the **ObixExportFolder** under the **Obix Network**.

NOTE:

In an Obix Network used only in a Niagara R2 station all interaction with oBIX devices is limited to Niagara R2 hosts. The ObixExportFolder and ObixExports are not used, as the Niagara R2 stations operate as oBIX servers only (they have no client interface).

Figure 21 Obix Export Manager view



To view, double-click the **Exports** folder in the Nav tree, or right-click and select **Views→Obix Export Manager**.

Discovered pane

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

The **Discovered** pane in the **Obix Export Manager** view has these available columns:

Table 3 Discovered pane columns

Column	Description
Name	Reports the name of the component in the local station.
Path	Reports the station path of the component, relative to the root.
Type	Identifies the type of component. Currently, only writable control points (including writable proxy points) can be added to the database: BooleanWritable, EnumWritable, NumericWritable, and StringWritable.

Database pane

By default, these columns appear in the **Discovered** pane of the **Obix Export Manager** view:

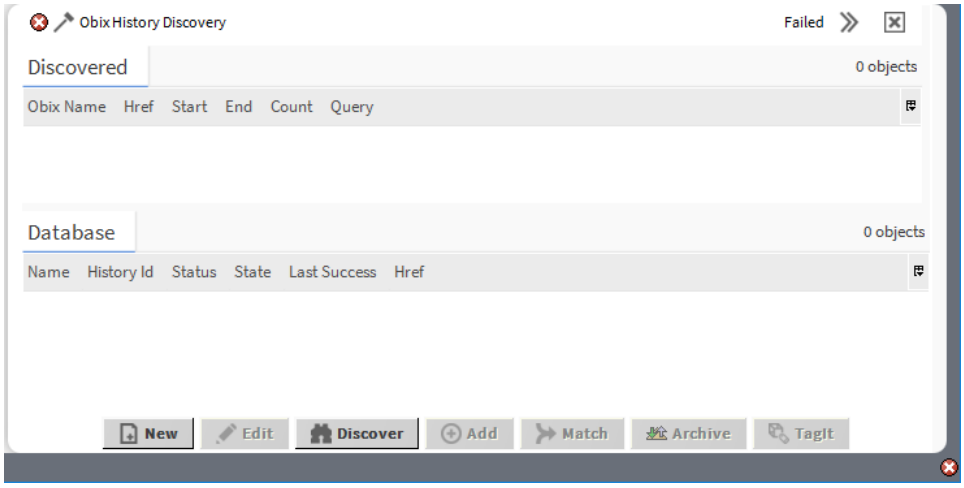
Table 4 Database pane columns

Column	Description
Name	Reports the name of the export descriptor, often left the same as the source control point.
Point	Reports the ORD for the slot in the station for this control point.
Value	Reports the current out value for the control point.
Priority	Identifies the **`Priority`** input of the control point exported (linked) to oBIX for remote writes.
Fault Reason	Explains why an export descriptor is in fault. Typically, this reflects an invalid configuration, such as selecting a **`Priority`** level that is already linked on the target control point.
Facets	If configured, shows the facets in use by the target control point.

obixDriver-ObixHistoryManager

This view discovers, adds, and manages history imports under the **Histories** extension (ObixHistoryDeviceExt) of a selected **`ObixClient`** or **`R2ObixClient`**. It is the default view on the **Histories** extension.

Figure 22 Obix History Manager view



To access this view double-click the **Histories** extension or right-click and select **Views→Obix History Manager**.

Discovered pane

The **Discovered** pane in the **Obix History Manager** view has these available columns:

Table 5 Discovered pane columns

Column	Description
Obix Name	Reports the name of the history on the oBIX server.
Href	Reports the URI of the history on the oBIX server.
Start	Displays the timestamp of the first record in the oBIX history.
End	Displays the timestamp of the last record in the oBIX history.
Count	Displays the total number of records in the oBIX history.
Query	Similar to Href but with `.log` or `.query` suffix to describe a history query.

Database pane

By default, the following columns appear in the **Discovered** pane of the **Obix History Manager** view:

Table 6 Database pane columns

Column	Description
History Id	Displays the **`History Id`** for the history created by the import descriptor, which defaults to: `[name of ObixClient] / [oBIX history name]` Where `[name of ObixClient]` identifies the ObixClient, and `[oBIX history name]` identifies the history.
Status	Reports the status of the history import descriptor.
State	Reports the current state of history import descriptor, as either `Idle` or `In Progress`.
Last Success	Reports the timestamp of the last successful history import.
Href	Reports the URI of the query to the history on the oBIX server.

Using the table options control, these additional data columns are available:

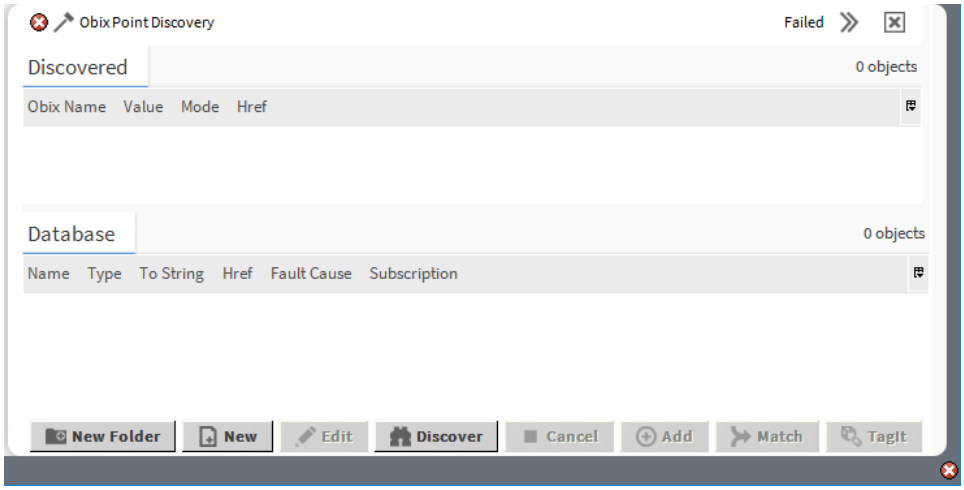
Table 7 Database pane (additional columns)

Column	Description
Name	Reports the name of the history import descriptor, defaulting to the name of the oBIX history.
Execution Time	Reports when the driver is configured to import a history.
Enabled	Reflects whether history import descriptor is enabled (`true`) or disabled (`false`).
Last Attempt	Reports the timestamp of the last attempted history import.
Last Failure	Reports the timestamp of the last attempted history import failure (could not complete).
Fault Cause	Provides the reason why last history import failed.
Full Policy	Reports `Roll` or `Stop` for the import descriptor.
Capacity	Reports the configured record capacity of import descriptor (Unlimited, or some specific count).

obixDriver-ObixPointManager

This view adds, edits, and accesses Obix proxy points under the **Points** extension of a selected ObixClient, **`R2ObixClient`** or in an **ObixPointFolder**. It is the default view on all these components.

Figure 23 Obix Point Manager view



To access this view, double-click the **Points** extension or **ObixPointFolder,** or right-click the **Obix Network** in the Nav tree, click **Views→AX Property Sheet**, expand or double-click the **ObixClient**, and click **Points**.

As in some other point managers, there is a **Discovered** pane (if in Learn mode) and a **Database** pane. 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.

Discovered pane

The **Discovered** pane in the **Obix Point Manager** view has these available columns:

Table 8 Discovered pane columns

Column	Description
Obix Name	Reports the name of the object on the oBIX server.
Value	Reports the value of the object at the time of discovery (expansion of its parent's leaf in the lobby).
Mode	Reports if the point is `RO` (read-only) or `RW` (read-writable). An oBIX proxy point for an `RW` item can be created either as a read-only type (NumericPoint, BooleanPoint, etc.) or as a writable type (Numeric-Writable, BooleanWritable, etc.).
Href	Reports the URI of the point on the oBIX server.

NOTE:

Discovery caches results. 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.

Database pane

By default, these columns appear in the **Discovered** pane of the **Obix Point Manager** view:

Table 9 Database pane columns

Column	Description
Name	Reports the name of the proxy point, if a root level point often left the same as the (discovered) object item name.
Type	Identifies the 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	Reports the last read value of a data item.
Href	Reports the URI of the point on the oBIX server.
Fault Cause	String describing the cause of the proxy point status fault, if any.

Using the table options control, these additional data columns are available:

Table 10 Database pane (additional columns)

Column	Description
Enabled	Reflects whether proxy point is enabled (`true`) or disabled (`false`).
Facets	Reflect the facets in use by the proxy point.
Conversion	Reports the conversion type used by the ObixProxyExt, which is typically Default.
Tuning Policy Name	Reports the name of the ObixTuningPolicy that the proxy point is assigned to.
Device Facets	Reflects the read-only device facets used in the point's proxy extension.
Path	Reports the 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.

Chapter 5 Windows

Topics covered in this chapter

♦ New Device Type Window
♦ New Device Properties Window
♦ New Point Properties Window

Windows create and edit database records or collect information when accessing a component. You access them by dragging a component from a palette to a Nav tree node or by clicking a button.

Windows do not support **On View (F1)** and **Guide on Target** help. To learn about the information each contains, search the help system for key words.

New Device Type Window

This window adds a sequentially-addressed range of multiple oBix devices.

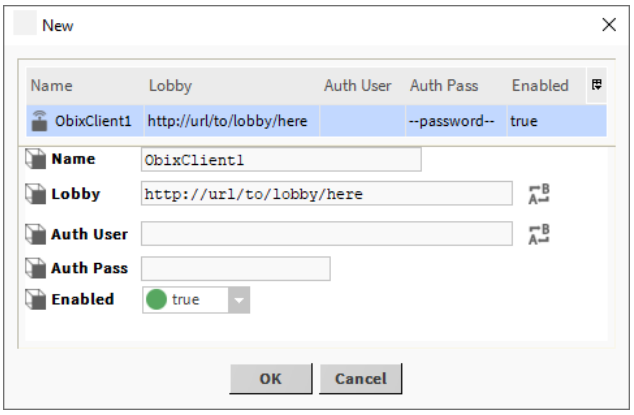
Figure 24 Example of a New ObixClient window

Type	Value	Description
Type to Add	drop-down list	Identifies the network to which the device is connected.
Number to Add	number (defaults to 1)	Configures how many devices to add.

New Device Properties Window

This window contains the properties for creating a new device.

Figure 25 New device window



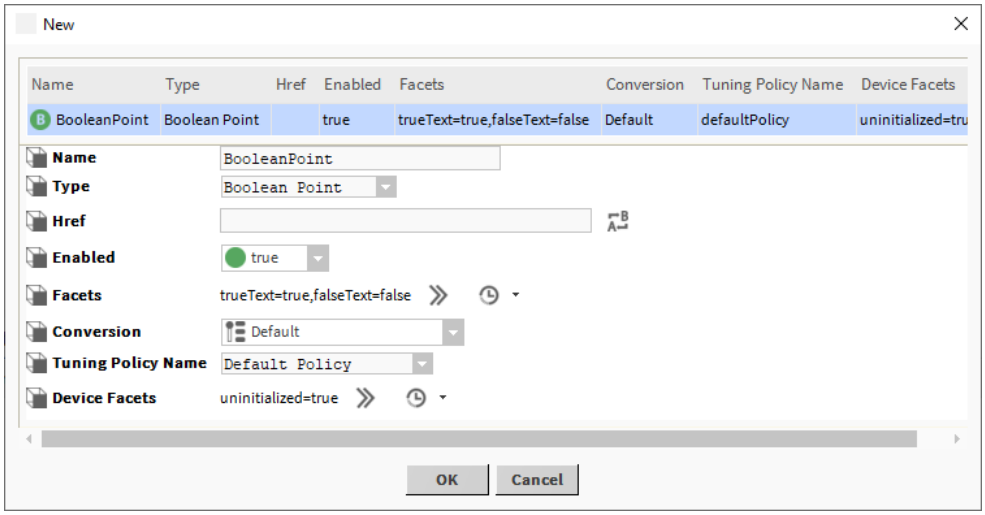
Type	Value	Description
Name	text	Specifies the name of the object.
Lobby	url	Specifies the URL
Auth user	text	Specifies the User name

Type	Value	Description
Auth Pass	text	Specifies the User password
Enabled	`true` or `false` (defaults to `true`)	Activates and deactivates use of the component.

New Point Properties Window

This window configures the point properties.

Figure 26 New point properties



Type	Value	Description
Name	text	Shows the name of the point as reported by the device.
Type	drop-down list	Selects the type of point.
Href	text	Reports the URI of the point on the oBIX server.
Facets	`true` or `false` (defaults to `true`) Please Confirm	Displays point's value in the station
Device Facets	Read only	Reflects the read-only device facets used in the point's proxy extension.
Enabled	`true` or `false`	Activates (`true`) and deactivates (`false`) the object (network, device, point, component, table, schedule, descriptor, etc.).

Type	Value	Description
Conversion	drop-down list	Selects the units to use when converting values from the device facets to point facets. `Default` automatically converts similar units (such as Fahrenheit to Celsius) within the proxy point. **NOTE:** In most cases, the standard `Default` conversion is best. `Linear` applies to voltage input, resistive input and voltage output writable points. Works with linear-acting devices. You use the Scale and Offset properties to convert the output value to a unit other than that defined by device facets. `Linear With Unit` is an extension to the existing linear conversion property. This specifies whether the unit conversion should occur on "Device Value" or "Proxy Value". The new linear with unit convertor, will have a property to indicate whether the unit conversion should take place before or after the scale/offset conversion. `Reverse Polarity` applies only to Boolean input and relay output writable points. Reverses the logic of the hardware binary input or output. `500 Ohm Shunt` applies to voltage input points only. It reads a 4-to-20mA sensor, where the Ui input requires a 500 ohm resistor wired across (shunting) the input terminals. `Tabular Thermistor` applies to only a Thermistor input point and involves a custom resistance-to-temperature value response curve for Type 3 Thermistor temperature sensors. `Thermistor Type 3` applies to an Thermistor Input point, where this selection provides a "built-in" input resistance-to-temperature value response curve for Type 3 Thermistor temperature sensors. `Generic Tabular` applies to non-linear support for devices other than for thermistor temperature sensors with units in temperature. Generic Tabular uses a lookup table method similar to the "Thermistor Tabular" conversion, but without predefined output units.
Tuning Policies	additional properties	Configures network rules for evaluating both write requests to writable proxy points as well as the acceptable freshness of read requests.

Glossary

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.
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 URI.
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.

Index