Technical Document

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

August 17, 2010

Niagara^{AX} Opc Guide

Confidentiality Notice

Trademark Notice

Copyright and Patent Notice

Contents

Preface

This documents usage of the OPC Data Access client driver for the NiagaraAX platform.

Document Change Log

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

- Updated: August 17, 2010
  Minor typo correction in "Windows Tasks" appendix, where command to launch DCOMCNFG was corrected. A note about Windows in that appendix was also added. Added new legal text in flyleaf.
- Updated: February 18, 2008
  More details in DCOMCNFG procedures for Windows XP given in the "Windows Tasks" appendix. Converted document to "new look" print format. Changed document to reference the *NiagaraAX Drivers Guide*, a new document created from sections formerly in the *NiagaraAX User Guide*.
- Revised: September 12, 2006
  Reworked as a "single-source" document, replacing the previous *docOpc* in NiagaraAX Help and the PDF-only *NiagaraAX Opc User Guide*. Includes more details throughout, as well as various screen-caps in the "Niagara Opc Concepts" section.
- Revised: July 14, 2006
  Added Read Delay and Write Delay properties to OpcPointDeviceExt.
- Revised: May 12, 2006
  Fixed step numbering problem in section "Configure the OPC Server Application".
- Publication: June 14, 2005
  Initial publication as 11 page PDF-only document.

CHAPTER 1

Opc Setup

See the following sections about setup for the NiagaraAX OPC driver:

Compatibility

NiagaraAX OPC driver compatibility encompasses the following:

OPC Specifications

The OPC Data Access Client (driver) is compatible with servers implementing OPC Data Access Specifications 1.x - 2.x. Other specifications such as Alarms & Events are not supported.

NiagaraAX platform

The OPC Data Access Client software functions only on Windows operating systems, starting with Window 2000 Service Pack 3 and beyond. This means the station must run on a Win-32 based platform, such as a JACE-NXS or -NX, AX SoftJACE, or a PC especially licensed for the OPC driver.

Note: *The OPC driver is not supported in any QNX-based platform, such as any JACE-2/6, -4, or -5 series controller.*

Also, starting in AX-3.4, NiagaraAX support for Windows 2000 was dropped.

OPC network

Although not necessary, it is highly recommended that both OPC client (NiagaraAX platform) and OPC server machines be on a domain, and it be the same domain for both.

Supported OPC data types

Table 1-1 lists the OPC data types that can be modeled as points by this driver, including the “label” for that data type shown in some Workbench dialogs. Some data types such as currency are unsupported, but can be manually created as string objects (arrays cannot be supported this or any other way).

Table 1-1 *Supported OPC data types*

OPC Item Type and value	Data Type (WB) label	OPC Item Type Description
VT_I2 = 2	Vt Int2	2-byte integer value
VT_I4 = 3	Vt Int4	4-byte integer value
VT_R4 = 4	Vt Real4	IEEE 4-byte real value
VT_R8 = 5	Vt Real8	IEEE 8-byte real value
VT_I1 = 16	Vt Signed Byte	1-byte character value
VT_UI1 = 17	Vt Unsigned Byte	Unsigned 1-byte character
VT_UI2 = 18	Vt Uint2	Unsigned 2-byte integer value
VT_UI4 = 19	Vt Uint4	Unsigned 4-byte integer value
VT_INT = 22	Vt Int	Integer value
VT_UINT = 23	Vt Uint	Unsigned integer value
VT_BSTR = 8	Vt String	String value
VT_BOOL = 11	Vt Boolean	Unsigned 4-byte integer value

NiagaraAX OPC driver 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 in various revision-named subfolders under the "sw" folder.

Apart from installing the 3.*n.nn* version of the Niagara distribution file in the JACE-NXS or -NX, make sure to install the **opc** module too (if not already present, or upgrade if an older revision). For more details, see "About the Commissioning Wizard" in the *JACE-NXS NiagaraAX Install and Startup Guide*.

Additional DCOM configuration may be necessary on both the OPC client (Niagara) side and the OPC application server side, as described in the next two sections. See "Client DCOM Configuration" and "Server DCOM Configuration".

Note: *Following this, the station is ready for OPC software integration, as described in other main sections of this document. Procedures for using the Opc driver, including online discovery of OPC folders and data items (and their addition to your Niagara station), is in the next main section. See "Opc Quick Start" on page 2-1.*

Client DCOM Configuration

If the client operating system is Windows XP Service Pack 2 (such as with a JACE-NXS), see the Appendix A section, "Windows XP Service Pack 2 (and later)" on page A-1. Otherwise, no other DCOM configuration is required on the client.

Server DCOM Configuration

The following two sections discuss DCOM configuration on the OPC server machine. If the server operating system is Windows XP Service Pack 2, also see "Windows XP Service Pack 2 (and later)" on page A-1.

- Configure OPCEnum
- Configure the OPC server application

Configure OPCEnum

OPCEnum is software provided by the OPC Foundation and is used to discover OPC servers in the host on which it is installed. If the OPC server machine does not have OPCEnum installed, the "Class ID" of the OPC server application will be required for a remote client, or the "Program ID" will be required for a local client (consult your OPC server documentation to obtain these).

On the OPC server, perform the following procedure:

Step 1 Launch DCOMCONFG (see Appendix A, "Launching DCOMCNFG" on page A-1).

Step 2 Find **OPCEnum** and go to its properties.

- Windows 2000: select the **Applications** tab.
- Windows XP: Expand **My Computer**, then **DCOM Config**. OpcEnum is located under **DCOM Config**.

Step 3 **General** tab: Set the "Authentication level" to None.

Step 4 **Security** tab: Do the following:

1. Choose "Use custom access permissions" and make sure the following users are in the list with all permissions allowed:
   - ANONYMOUS LOGON (if possible)
   - Everyone
   - INTERACTIVE
   - NETWORK
   - SYSTEM
2. Choose "Use custom launch permissions," and make sure that the same users are in the list as for access permissions (as above).

Step 5 **Indentity** tab: Select the following:

- "The System Account," if possible.
- If the client and server are on the same domain, select the "The interactive user." Otherwise, select "This user," and enter the credentials of a user valid on the OPC server host.

Configure the OPC server application

The following procedure describe how to configure the OPC server application for anonymous connections:

Step 1 Launch DCOMCONFG (see Appendix A, "Launching DCOMCNFG" on page A-1).

Step 2 Find the OPC Server application, and go to its properties.

- Windows 2000: select from the list in the **Applications** tab.
- Windows XP: Expand **My Computer**, then **DCOM Config**. OpcEnum is located under **DCOM Config**.

Step 3 **General** tab: Set the "Authentication level" to None.

Step 4 **Security** tab: Do the following:

1. Choose "Use custom access permissions" and make sure the following users are in the list with all permissions allowed:
   - ANONYMOUS LOGON (if possible)
   - Everyone
   - INTERACTIVE
   - NETWORK
   - SYSTEM
2. Choose "Use custom launch permissions," and make sure that the same users are in the list as for access permissions (as above).

Step 5 **Indentity** tab: Select the following:

- "The System Account," if possible.
- If the client and server are on the same domain, select the "The interactive user." Otherwise, select "This user," and enter the credentials of a user valid on the OPC server host.

Chapter 2

Opc Quick Start

This section provides a collection of procedures to use the Opc driver to make Opc proxy points, the station interface to OPC data items. As with other NiagaraAX drivers, you can do most configuration from special "manager" views and property sheets using Workbench. These are the main subsections:

- Configure the OpcNetwork
- Create Opc proxy points

Configure the OpcNetwork

To add and configure the OpcNetwork, perform the following main tasks:

- Add an OpcNetwork
- Discover and add OpcDaClients

Add an OpcNetwork

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

To add an OpcNetwork in the station

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 DeviceNetwork dialog. For more details, see "Driver Manager New and Edit" in the *Drivers Guide*.

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

This brings up a dialog to name the network.

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

You should have an OpcNetwork named "OpcNetwork" (or whatever you named it), under your Drivers folder.

Discover and add OpcDaClients

The OpcDaClient is the "device-level" component. Each OpcDaClient represents the client (Niagara) connection to one OPC *server*. For details, see "About the OpcDaClient" on page 3-4.

To discover and add OpcDaClients

Step 1 Double-click the **OpcNetwork**, or:

right-click the OpcNetwork and select **Views > Opc Device Manager**.

This brings up the **Opc Device Manager**.

Step 2 Click the **Discover** button.

A popup **Address** dialog appears, defaulting to "localhost."

Step 3 Type in the IP address for the OPC server you wish to integrate with, and click **OK**.

This launches an Opc Device Discovery job.

A progress bar appears at the top of the view, and updates as the discovery occurs.

Step 4 When the job completes, any discovered OPC server is listed in the *top pane* of the view, in the "Discovered" table (Figure 3-3). The bottom pane, labeled "Database," is a table of OpcDaClients that are currently mapped into the Niagara station—initially, this table will be empty.

Step 5 Double-click the discovered OPC server (or click it, then click the Add button).

The **Add** dialog appears, in which you can typically accept all defaults (see Figure 3-4 on page 4).

Step 6 Click **OK** to add the OpcDaClient component to your station.

See the next section: "Create Opc proxy points".

Create Opc proxy points

As with device objects in other drivers, each OpcDaClient 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 Opc Point Manager). You use it to create Opc proxy points under any OpcDaClient.

This section provides quick start procedures for both tasks, as follows:

- Using online Discover to add Opc proxy points
- Manually adding Opc proxy points

Note: *For general information, see "About the Point Manager" in the* Drivers Guide*.*

Using online Discover to add Opc proxy points

This is the recommended way to accurately add Opc proxy points under an OpcDaClient. Use the following procedures:

- To discover OPC folders and data items
- To add discovered OPC data items as Opc proxy points

To discover OPC folders and data items

Step 1 In the **Opc Device Manager**, in the **Exts** column, double-click the Points icon in the row representing the OpcDaClient you wish to explore.

This brings up its **Opc Point Manager**.

Step 2 Click the **Discover** button to learn the OPC folders and data items that are available.

When the discovery job completes, discovered items are listed in the *top pane* of the view, in an expandable tree in the "Discovered" table. Depending on the structure of the OPC server, each folder is a container for either/both more (subordinate) OPC folders, or perhaps OPC data items.

See "Discovered pane notes" on page 3-8 for additional details.

Note: *You are not required to "mirror" the group structure of the OPC server in the OpcNetwork—use of OpcPointFolders is an organizational utility, and is optional.*

Step 3 To add Opc proxy points, see "To add discovered OPC data items as Opc proxy points".

To add discovered OPC data items as Opc proxy points

Perform this task to add Opc proxy points.

Step 1 Select the OPC data items in the Database pane of the Opc Point Manager.

Step 2 You can map selected points in the station in different ways:

- Drag from the Discovered pane to Database pane (brings up an **Add** dialog).
- Double-click an item in the Discovered pane (also brings up an **Add** dialog).

This works the same as in other driver's Point Manager views.

Step 3 When the **Add** dialog appears, you can optionally edit a number of fields for each OPC data item.

The following brief summaries explain a *few* Add dialog fields:

- **Name** is the Opc proxy point name—you can change this if needed.
- **Type** is the Opc "type," which defaults either to a read-only point (NumericPoint, BooleanPoint, StringPoint) or writable point (NumericWritable, BooleanWritable, StringWritable), depending on the type and mode of data.
  Note: *Unlike other entries in the Add dialog, you cannot edit the point's Type later.*
- Id is the learned data item's full hierarchical address, including folders.

For complete details on all fields, see "Add and Edit dialog fields" on page 3-9.

Step 4 When you have Opc proxy point(s) configured properly for your usage, click **OK**.

The proxy points ares added to the station, and appear listed in the Database pane.

For more details, see "Opc proxy point" on page 3-10.

Manually adding Opc proxy points

You can manually add Opc proxy points, using the **New** button in the Opc Point Manager, or by dragging from the opc palette. However, this method is generally not recommended.

Chapter 3

Niagara Opc Concepts

This section provides conceptual details on the OPC driver and its components, including views. These are the main subsections:

OPC terms

The following list of terms and abbreviations is specific to OPC usage in NiagaraAX, and covers entries used in this document. For the definitive collection of terms found in OPC publications, refer to the OPC Foundation, at http://www.opcfoundation.org.

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

Asynchronous vs. Synchronous IO Asynchronous Input / Output (that is, communications) in the OPC context means server-initiated change messages sent to the client. Asynchronous (or async) IO is the most efficient and scalable integration option, because the client does not have to poll for changes of value (synchronous)—the server simply sends change-of-value messages as they occur.

COM and DCOM Most OPC specifications are embodied as COM interfaces. COM is Microsoft's Common Object Model technology. DCOM is distributed COM, where objects in remote processes appear to be local objects.

OPC Data Access There are several OPC specifications. The OPC Data Access specification provides the baseline functionality for reading and writing data from various networked devices via a standard set of interfaces.

OPC Group An OPC Group is an object that an OPC server uses to manage a collection of items. OPC Group objects are created by OPC clients who specify what server items belong in the group. Client grouping does not have to match the organization of the data on the server.

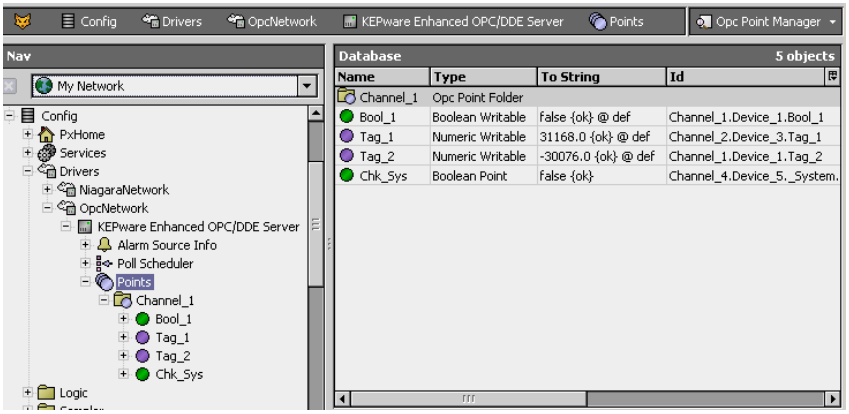
OPC Item OPC refers to individual data points as items.

About Opc Architecture

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

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

Figure 3-1 *Opc driver architecture*



Unlike some other drivers, Points is the *only* "device extension" under an OpcDaClient.

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

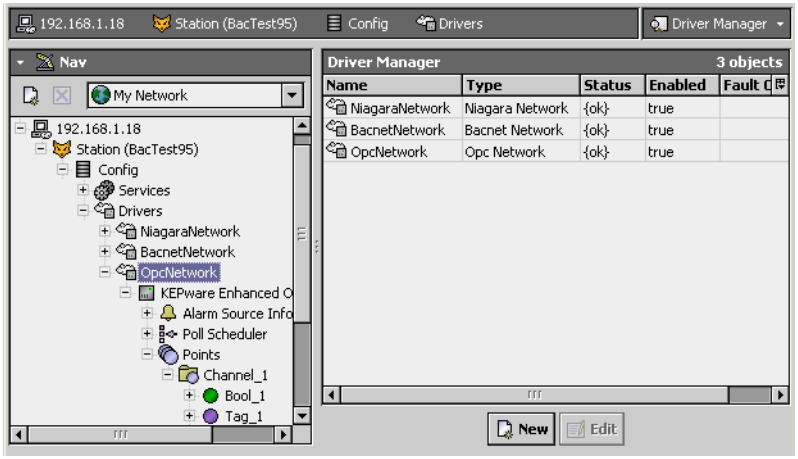
About the Opc Network

The OpcNetwork is the top-level container component for "everything OPC" in a station.

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

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

Figure 3-2 *OpcNetwork in DriverContainer*



Note: *The JACE-NXS or JACE-NX platform must have the Opc module installed. Otherwise, an error occurs explaining that the Opc module is missing. If this occurs, install the* **opc** *module in that platform and repeat the operation.*

The OpcNetwork 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 OpcDaClient (device-level) component.

In addition, the following OpcNetwork property has special importance:

- **Thread Pool**
  Controls the number of threads used to execute all actions of all OPC 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 following sections provide additional OpcNetwork details:

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

Opc Network status notes

As with most "fieldbus" drivers, the status of an OpcNetwork 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 OpcNetwork has an available "Alarm Source Info" container slot you can use to differentiate OpcNetwork alarms from other component alarms in the station. See "About network Alarm Source Info" in the* Drivers Guide *for more details.*

Opc Network monitor notes

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

Opc Network tuning policy notes

The OpcNetwork 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 OpcTuningPolicy exists, however, you can add new tuning policies (duplicate and modify) as needed.

Opc Network views

The OpcNetwork's default view is the **`Opc Device Manager`**, equivalent to the Device Manager in most other drivers. You use this view to discover and add OpcDaClient components to the station. For details, see "Opc Device Manager" on page 3-3.

Other standard views are also available on the OpcNetwork. However, apart from the Opc Device Manager, you typically access only its property sheet.

Opc Device Manager

As the default OpcNetwork view, the Opc Device Manager provides an online "Learn Mode" to find OPC server(s) on a specified host. This lets you add one or more "device-level" OpcDaClient components to the station database (Figure 3-3). For general information, see "About the Device Manager" in the *Drivers Guide*.

Figure 3-3 *Adding OpcDaClient discovered using Opc Device Manager*

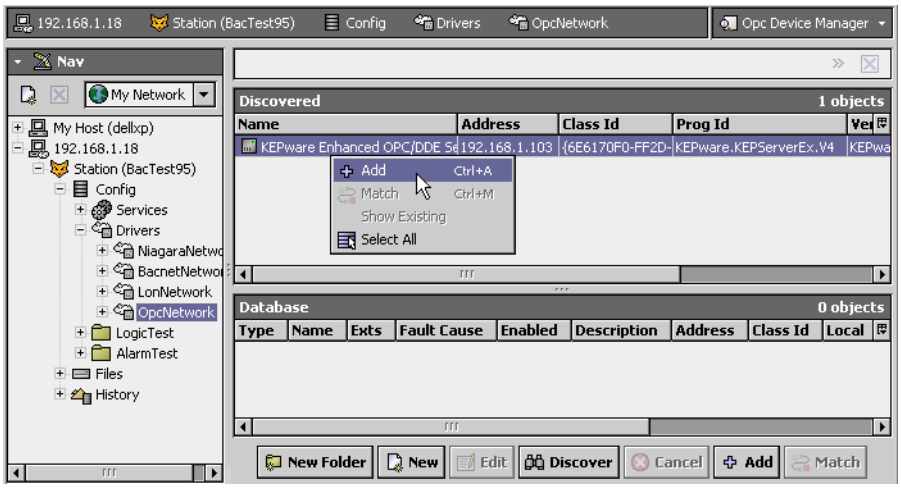
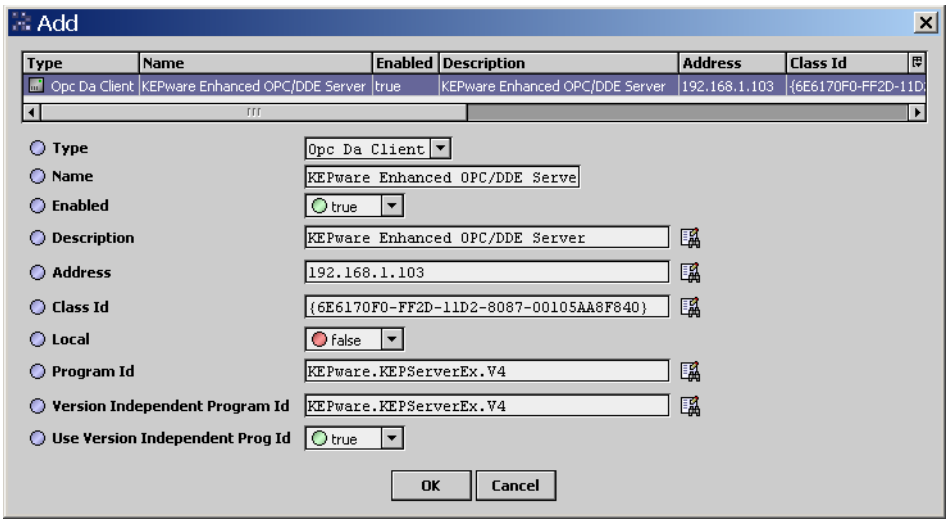


Figure 3-4 *Add dialog for OpcDaClient*



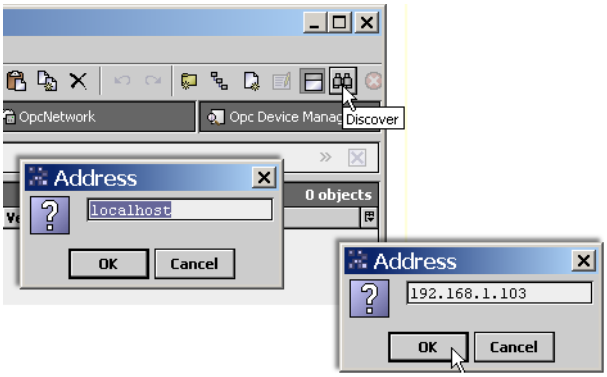
After adding a discovered OPC server, you use the Opc Point Manager view of the **`Points`** container (under its OpcDaClient component) to add Opc proxy points—one for each OPC data item of interest.

- Opc Device Manager usage notes

Opc Device Manager usage notes

Unlike with some other drivers, a device discover job is *not* network-wide—each time you must enter the IP address or hostname of the target OPC server (Figure 3-3), which defaults to "`localhost`." If you have multiple OPC server hosts, you will need to run that many Opc Device Discover jobs.

Figure 3-5 *Discover in Opc Device Manager is directed to one host*



Additional usage notes about the Opc Device Manager include:

- Offline engineering is possible (using New), but is not typically used.
- Typically you do not change any defaults in the Add dialog (Figure 3-4) when adding an OpcDaClient, unless the OPC server does not have OPCEnum installed. In this case, you must enter the Class Id (CLSID) of the server, and possibly other Config properties in the OpcDaClient's property sheet.

About the OpcDaClient

The OpcDaClient represents the client connection to any version of OPC Data Access server. As the "device-level" component in the Opc driver architecture, it is similar to most driver's device components— see "Common device components" in the *Drivers Guide* for general information.

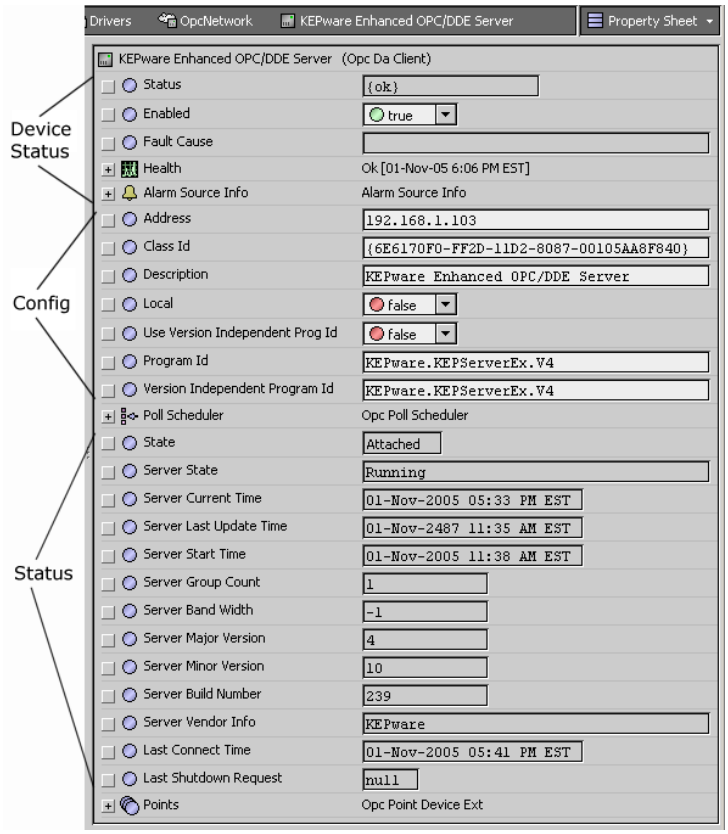
The following main sections provide more details on the OpcDaClient:

- OpcDaClient properties
- OpcPointDeviceExt (Points container)

OpcDaClient properties

Figure 3-6 shows the property sheet for an example OpcDaClient.

Figure 3-6 *OpcDaClient property sheet*



As shown in the figure above, OpcDaClient properties can be categorized into three groups:

- Device status properties (typical)
- Config properties
- Status properties

Device status properties

An OpcDaClient has typical device-level status properties (see "Device status properties" in the *Drivers Guide*). The following notes apply:

- **Status**
  Status of OpcNetwork communications to this OpcDaClient. Possible status flags include:
  - ok - Normal communications, no other status flags.
  - disabled - Enabled property is set to false, either directly or in OpcNetwork. While status is disabled, all child Opc points have disabled status; OpcDaClient polling is suspended.
  - fault - Typically a licensing issue, if seen.
  - down - Error communicating to the OPC server. If status was previously ok (without changing OpcDaClient configuration), this may mean the server host is now unreachable, or the server program is not running.
- **Enabled**
  Either true (default) or false. Can be set directly or in parent OpcNetwork. See Status disabled description above.
- **Health**
  Contains properties including timestamps of last "ok" time and last "fail" time, plus a string property describing last fail cause.
- **Fault Cause**
  If status has fault, describes the cause.

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

OpcDaClient Config properties

In addition to common status properties, each OpcDaClient has the following unique configuration properties:

Note: *These properties are included when you* **`Add`** *or* **`Edit`** *an OpcDaClient from the* **`Opc Device Manager`** *(see Figure 3-4), in addition to being on its property sheet view (Figure 3-6).*

- **Address**
  This is the IP address or hostname of the OPC server.
- **Class Id**
  This is the Windows class ID, or CLSID, of the OPC server instance. It is set automatically when the server is discovered, and is only needed for remote servers.
  Note: *If the OPC server does not support discovery, you must manually enter its CLSID here.*
- **Description**
  (Informational only) On a learned OPC server, this is typically populated with descriptive text.
- **Local**
  Set to True if the OPC server is on the local Niagara platform, or False if on a remote host.
- **Use Version Independent Program Id**
  This is for local servers. Set to True to use the versions independent Program ID. It might be desired to connect to an older version of the server, in which this case, set this to False.
- **Program Id**
  Either this or the Version Independent Program Id are needed to connect to local servers.
- **Version Independent Program Id**
  Either this or the Program Id are needed to connect to local servers.
- **Poll Scheduler**
  See "OpcPollScheduler" on page 3-7.

OpcDaClient Status properties

In addition to device-level Config properties, each OpcDaClient has the following unique (read-only) status properties about the OPC server and its Niagara client connection. These are visible from the property sheet of the OpcDaClient (Figure 3-6).

- **State**
  The current client state, as either: Attached, Attaching, Detached, or Detaching.
- **Server State**
  From the Monitor (ping) result, provides the current server status as one of the following:
  - Running — Server is running normally.
  - Failed — A vendor-specific fatal error occurred within the server.
  - No Configuration — Server is running, but has no configuration information loaded. Thus, it cannot function normally.
  - Suspended — Server has been temporarily suspended, and is not receiving or sending data.
  - Test — Server is in test mode. Outputs are disconnected from the real hardware; but the server otherwise behaves normally.
  - Communications Failure — Server is running properly, but is having problems accessing data from its sources. Expect data items affected by this to have individual fault status.
- **Server Current Time**
  Ping result, as the current time of the server.
- **Server Last Update Time**
  Ping result, the time the server thinks it sent the last data value update to this client.
- **Server Start Time**
  Ping result, as time the server instance started.
- **Server Group Count**
  Ping result, as total number of groups being managed by the server instance.
- **Server Band Width**
  Ping result, the behavior of this field is server-specific. The suggested use is the approximate percent of bandwidth currently in use by the server. Any value over 100% indicates the update rate is too high. The server may also return a value of "0xFFFFFFFF" or 4294967295, if this value is unknown.
- **Last Connect Time**
  The last time the client successfully connected to the server.
- **Last Shutdown Request**
  The last time the server sent a shutdown request to the (Niagara) client.

OpcPollScheduler

Each OpcDaClient has a **`Poll Scheduler`** container slot (OpcPollScheduler), with the standard collection of properties, as described in "About poll components" in the *Drivers Guide*. Note the poll scheduler is "device level" (each OpcDaClient), versus a single poll scheduler at the "network level."

Note: *Polling by the station is used only when asynchronous messaging is not in use—otherwise, the server sends change-of-value messages for updates. See properties of the* **`Points`** *device extension (OpcPointDeviceExt) for related details.*

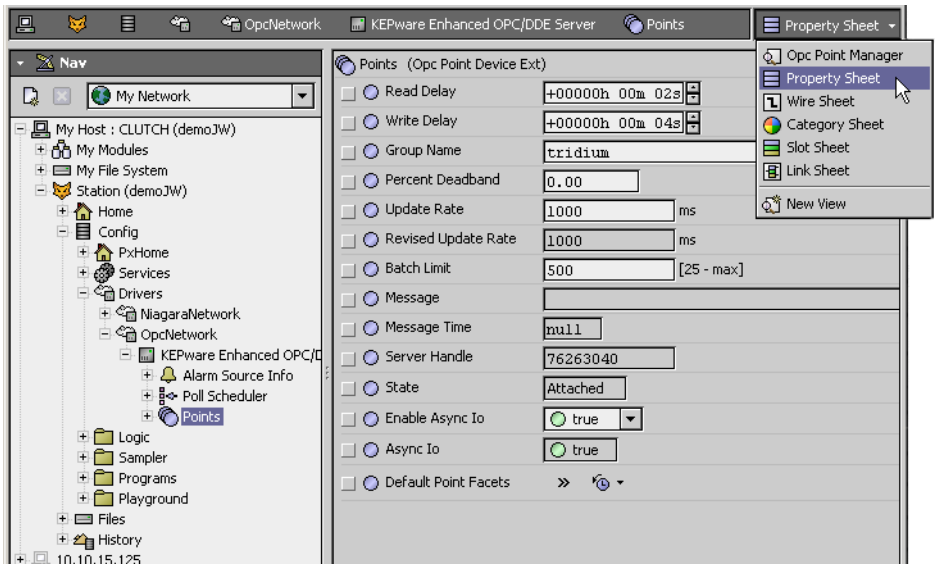
OpcPointDeviceExt

Each OpcDaClient has a **`Points`** device extension (OpcPointDeviceExt). Note that unlike some other drivers, *other device extensions* such as **`Schedules`**, **`Histories`**, and **`Alarms`** are not used.

The **`Points`** extension represents one "OPC Group," where all Opc proxy points contained in the tree rooted by Points are members of this OPC Group. The primary interface to the Points extension is via its default view, the Opc Point Manager.

However, the property sheet of the Points extension is significant too, as shown in Figure 3-7.

Figure 3-7 *OpcPointDeviceExt property sheet*



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

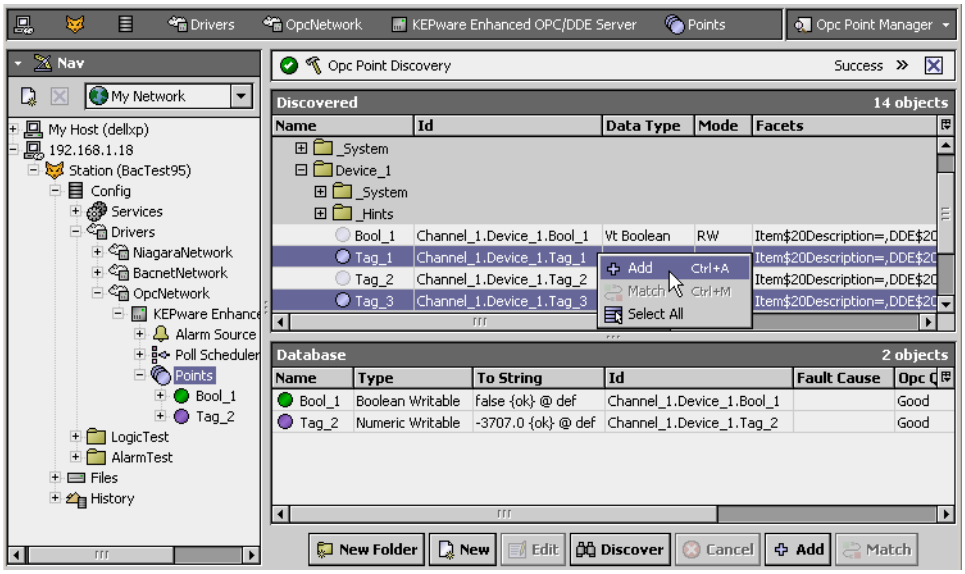
- **Read Delay**
  Amount of time after "device up" before the client creates a subscription and reads any values from the server.
- **Write Delay**
  Amount of time after "device up" before the client attempts to write any values to the server.
- **Group Name**
  Name of the group to create on the server for all points contained under this OpcPointDeviceExt.
- **Percent Deadband**
  Percentage of change in a (numeric) data item value that causes a server-initiated change-of-value message to be sent to the client (Niagara). Note this applies only to numeric Opc proxy points.
- **Update Rate**
  Client requested update rate in milliseconds, this specifies the fastest rate at which data changes may be sent by the server. For very large integrations, this value may need to be increased. Also see the "Server Band Width" property, one of the Status properties.
- **Revised Update Rate**
  (Read only) The server-returned value for the update rate.
- **Batch Limit**
  Specifies the maximum number of items that can be batched together into a single subscribe, unsubscribe, or synchronous read operation.
- **Message / Message Time**
  (Read only) If anything is wrong, check here.

- **Server Handle**
  (Read only) The server-generated handle for this group.
- **State**
  (Read only) Client group state, either `Attached`, `Attaching`, `Detached`, or `Detaching`.
- **Enable Async Io**
  If true, asynchronous change-of-value messages from the server are enabled. Note that Async Io is the most efficient and scalable way to integrate with an OPC server.
- **Async Io**
  (Read only) Whether or not async messaging is being used; some OPC servers do not support it.
- **Default Point Facets**
  Null, by default. If set to a non-null value, these facets are applied by the OpcPointManager to the proxy extension device facets first—they would then be overridden by any facets coming from the OPC server. These facets also affect the control point facets, since they are overridden by the proxy extensions device facets.

Opc Point Manager

As the default view for the Points extension under an OpcDaClient, the Opc Point Manager provides an online "Learn Mode" to find available data items. It operates similar to point manager views in other drivers. For general information, see "About the Point Manager" in the *Drivers Guide*. As shown in Figure 3-8, you use this view to discover and add corresponding Opc proxy points (for data items) and Opc point folders (for organizing, if desired) to the station database.

Figure 3-8 *Adding Opc proxy points discovered using Opc Point Manager*



You are *not* required to mimic the organizational structure of the OPC server when adding data items (Opc proxy points). For more details, see "Discovered pane notes" on page 3-8. For procedures, see "Create Opc proxy points" on page 2-2.

For listings of the available columns in the Discovered and Database panes of the Opc Point Manager, see "opc-OpcPointManager" on page 4-2.

Opc Point Manager usage notes

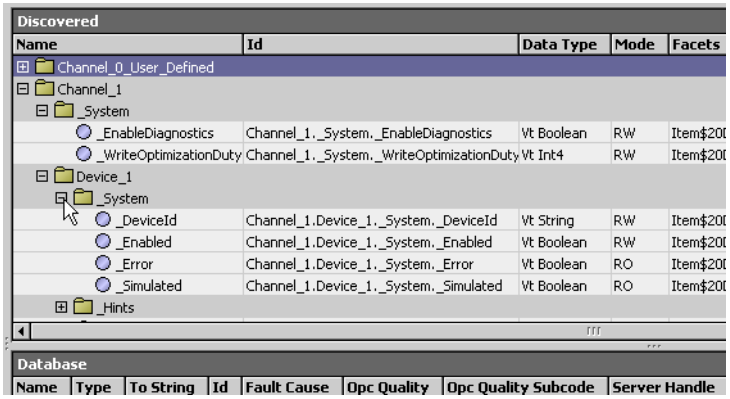
The following notes apply when using the Opc Point Manager:

- Discovered pane notes
- Add and Edit dialog fields

Discovered pane notes

When in "Learn Mode," the Discovered (top) pane of the Opc Point Manager shows the discovered "OPC Address Space" of the server. This is done in a tree fashion by showing the organization levels and child data items differently, as shown in Figure 3-9.

Figure 3-9 *Discovered pane in Opc Point Manager shows OPC Address Space*

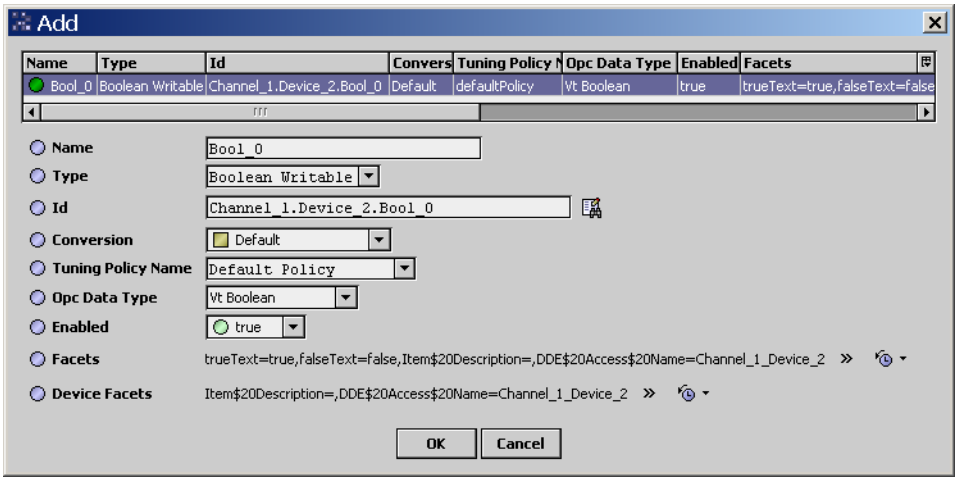


- *Folders* of items are shown in gray rows as expandable *folder icons* with "learned" descriptors. You can add these levels as OpcPointFolders, or simply expand to access child subfolders and/or data items. Adding folders is *optional*—note that any data item added (as a proxy point) automatically includes the server's hierarchy for it (in its OpcProxyExt "Id" property).
- *Data Items* are shown in white rows with *object icons* , again with "learned" descriptors. You can add these as Opc proxy points. If you double-click, or select and click the **Add** button, the Add dialog appears. This lets you select the control point *type* before it is added to the station. See the next section "Add and Edit dialog fields".

Add and Edit dialog fields

When adding Opc proxy points for discovered data items, the following items are available in the **Add** (or **Edit**) dialog (Figure 3-10):

Figure 3-10 *Add dialog in Opc Point Manager*



- **Name**
  Name of the Opc proxy point, as learned from the discovery. Often left at default, unless there are multiple points using this name (equivalent to right-click Rename, can be edited anytime).
- **Type**
  The type of Opc proxy point to create (*not* editable after adding). A default type is pre-selected, based upon discovered OPC "Data Type" and "Mode" (`RO`, `RW`) of the data item. If mode is writable (`R/W`), the default is a *writable* point type, as either:
  - **NumericWritable**
  - **BooleanWritable**
  - **StringWritable**

  Otherwise if mode is `RO`, type selection is limited to read-only point types (NumericPoint, BooleanPoint, or StringPoint).
- **Id**
  The unique ID of the data item on the server, including its full hierarchical folder structure.

- **Conversion**
  The conversion type used between units in the proxy extension's "Device Facets" and the units in the parent point's Facets. Typically, this left at "`Default`."
- **Tuning Policy Name**
  The assigned tuning policy for this proxy point, as configured in the OpcNetwork. See "Opc Network tuning policy notes" on page 3-3.
- **Opc Data Type**
  The Windows variant datatype of the OPC item. In most cases, the server provides this information. See Table 1-1 on page 2 for related details.
- **Facets**
  Facets for the Opc proxy point. Equivalent to accessing facets through the point's property sheet (and can be edited anytime). For more details, see "About point facets" in the *User Guide*.
- **Device Facets**
  Facets for the OPC data item, as known in the server. Will contain OPC "properties" for the point discovered by the Opc Point Manager. Typically, this value is left unchanged.

Note: *Additional OpcProxyExt properties are available in its property sheet—see "Opc Proxy Ext" on page 3-10.*

Opc proxy point

Opc proxy points are similar to other driver's proxy points. You can (and often do) add alarm and history extensions to them, and link them into other station logic as needed. For general information, see "About proxy points" in the *Drivers Guide*.

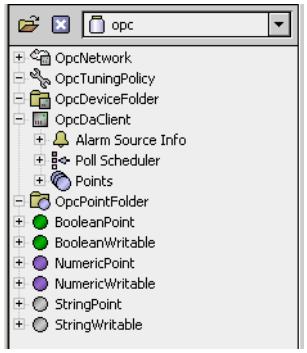
The following sections explain further:

- Opc point types
- Opc Proxy Ext properties

Opc point types

As seen in the **opc** palette (Figure 3-11), there are 6 different Opc proxy point types: BooleanPoint, BooleanWritable, NumericPoint, NumericWritable, StringPoint, and StringWritable.

Figure 3-11 *Opc components as seen in the* opc *palette*

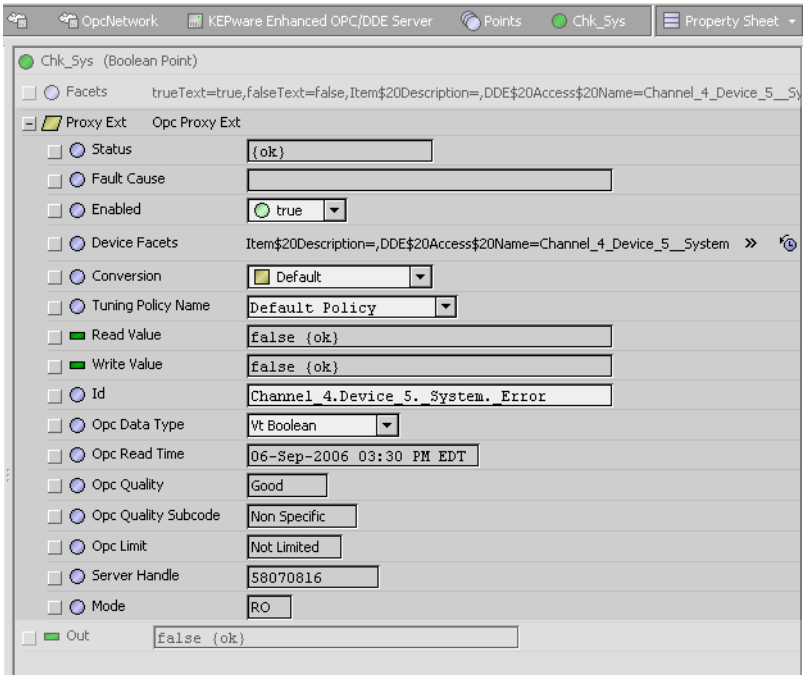


These 6 types are simply a subset of the 8 basic control point types, minus the EnumPoint and EnumWritable. Each point type has an Opc Proxy Ext (*proxy extension*).

Opc Proxy Ext

The OpcProxyExt has standard proxy extension properties such as Status and Enabled, among others (for details, see "ProxyExt properties" in the *Drivers Guide*). Figure 3-12 shows the property sheet of an Opc proxy point with its OpcProxyExt expanded.

Figure 3-12 *Common properties among Opc proxy extensions*



The following properties of the Opc Proxy Ext are unique or have special importance—note that some are also available in the **Add** or **Edit** dialog in the Opc Point Manager (see Figure 3-10):

- **Device Facets**
  Will contain OPC "properties" for points discovered by the Opc Point Manager.
- **Id**
  The ID of the point on the OPC server.
- **Opc Data Type**
  Windows variant datatype of the OPC item. In most cases, the server provides this information.
- **Opc Read Time**
  (read only) The server timestamp of the current value.
- **Opc Quality**
  (read only) The server-supplied status of the value, as either:
  - Good
  - Bad
  - Uncertain
- **Opc Quality Subcode**
  (read only) The server-supplied status subcode (numeric subcodes indicate vendor information):
  - Good
    - Non-specific
    - Local Override
  - Bad
    - Non-specific
    - Configuration Error
    - Not Connected
    - Device Failure
    - Last Known Value — Comm has failed, however the last known value is available.
    - Comm Failure — No last known value.
    - Out of Service
    - Waiting for Initial Data
  - Uncertain
    - Non-specific
    - Last Usable Value — Whatever was writing this value has stopped doing so.
    - Sensor Not Accurate
    - Engineering Units Exceeded
    - Sub-Normal — The value is derived from multiple sources, and has less than the required number of good sources.

- **Opc Limit**
  (read only) One of the following:
  - Not Limited — The value is free to move up or down.
  - Low Limited — The value has pegged at some lower limit.
  - High Limited — The value has pegged at some upper limit.
  - Constant — The value is a constant and cannot move.
- **Server Handle**
  (read only) The server handle used to refer to this item. A handle of 0 means the item has not been registered with the server.
- **Tuning Policy Name**
  If the point is not being updated via asynchronous server callbacks, this controls the tuning policy, and thus the poll frequency at which the point is synchronously polled.

Chapter 4

Opc 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.

Opc Plugin Guides Summary

Summary information is provided on views specific to components in the opc module, as follows:

- OpcDeviceManager
- OpcPointManager

opc-OpcDeviceManager

Use the **Opc Device Manager** to add, edit, and access OPC device components (OpcDaClients). The Opc Device Manager is the default view of a OpcNetwork. As in other device managers, there is a Discovered table (if in Learn mode) and a Database table. For additional details, see "Opc Device Manager" on page 3-3.

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

- **Name**
  Name of the OPC server application.
- **Address**
  Hostname or IP address of the OPC server.
- **Class Id**
  Windows class ID string, or CLSID, of the OPC server instance.
- **Program Id**
  Text string of OPC Program Id for server, typically containing one or more periods.
- **Version Independent Program Id**
  Text string of version-independent OPC Program Id for server, which may match the Program Id.
- **Cat Id**
  OPC specification of the server, for example "OPC DA 2.0" for Data Access 2.0.

Database table The Database table in the OpcDeviceManager view has the following available columns:

- **Type**
  Niagara type of device component, where currently there is only OpcDaClient (Opc Da Client).
- **Name**
  Niagara name of the OpcDaClient component, often left as the (discovered) OPC server application name.
- **Exts**
  Shortcut to the device extensions (only one extension, Points).
- **Fault Cause**
  String describing the cause of the device status fault, if any.
- **Enabled**
  Indicates whether the OpcDaClient is enabled (true) or disabled (false).
- **Description**
  Description property for the OpcDaClient.

- **Address**
  The hostname or IP address of the OPC server.
- **Class Id**
  The Windows class ID, or CLSID, of the OPC server instance.
- **Local**
  Set to true if the OPC server is on the local host, otherwise it is false.
- **Program Id**
  Either this or the Version Independent Program Id are needed to connect to local servers.
- **Version Independent Program Id**
  Either this or the Program Id are needed to connect to local servers.
- **Use Version Independent Program Id**
  Reflects this property of the OpcDaClient, which is typically true. If connecting to an older version of an OPC server, this may need to be set to false.
- **State**
  Reflects the client state, which is either attached, attaching, detached, or detaching.

opc-OpcPointManager

Use the OpcPointManager to add, edit, and access Opc proxy points under the OpcPointDeviceExt extension of a selected OpcDaClient, or in an OpcPointFolder. The OpcPointManager is the default view on both of those components. To view, *double-click* the OpcPointDeviceExt extension or OpcPointFolder, or right-click and select **`Views > Opc Point Manager`**.

As in other point managers, there is a Discovered table (if in Learn mode) and a Database table. For additional details, see "Opc Point Manager" on page 3-8.

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

- **Name**
  Name of the OPC folder or data item.
- **Id**
  The ID of the data item on the OPC server, including full hierarchical folder path.
- **Data Type**
  The (Windows) OPC datatype of an item, as provided by the server—for example: `Vt Real4`. See "Supported OPC data types" on page 1-1 for related information.
- **Mode**
  Either RO (read-only) or RW (read-writable).
- **Facets**
  Reflect default facets for a data item, including its description.

Database table The Database table in the OpcPointManager view has the following available columns:

- **Name**
  Niagara name of the Opc component, often left as the (discovered) folder or data item name.
- **Type**
  Niagara type of component, as either an Opc Point Folder (for a folder) or a type of control point if an Opc proxy point (for example, Boolean Point, Boolean Writable, Numeric Point, and so on).
- **To String**
  Last read value of a data item.
- **Id**
  The ID of the data item on the OPC server.
- **Fault Cause**
  String describing the cause of the point status fault, if any.
- **Opc Quality**
  Server-supplied status of a data item value, as either: `Good`, `Bad`, or `Uncertain`.
- **Opc Quality Subcode**
  Server-supplied status subcodes (numeric subcodes indicate vendor information), as either:
  Bad:
  - Non Specific
  - Configuration Error
  - Not Connected
  - Device Failure
  - Sensor Failure
  - Last Known Value (Comm has failed, however the last known value is available.)
  - Comm Failure (No last known value.)

- Out of Service
- Waiting for Initial Data

Uncertain:

- Non Specific
- Last Usable Value (Whatever was writing this value has stopped doing so.)
- Sensor Not Accurate
- Engineering Units Exceeded
- Sub Normal (Value is derived from multiple sources, with less than required number of good sources)

Good:

- Non Specific
- Local Override

- **Server Handle**
  Numeric server handle used to refer to this data item. A handle of 0 means the item has not been registered with the server.
  Note: *By default, remaining columns (listed below) are not enabled to display in the Database table. However, you can use the table controls to show/hide whichever columns needed.*
- **Conversion**
  Niagara conversion type used by the Opc proxy extension, which is typically Default.
- **Tuning Policy Name**
  Name of the Niagara OpcTuningPolicy that the proxy point is assigned to.
- **Data Type**
  The (Windows) OPC datatype of an item, as provided by the server—for example: `Vt Real4`. See "Supported OPC data types" on page 1-1 for related information.
- **Enabled**
  Reflects whether proxy point is enabled (true) or disabled (false).
- **Facets**
  Reflect the facets in use by the proxy point.
- **Device Facets**
  Reflects the read-only device facets used in the point's proxy extension.
- **Path**
  Reflects the station path for the Opc component, relative to the station's root.
- **Mode**
  Either RO (read-only) or RW (read-writable).
- **Opc Read Time**
  Reflects the server timestamp of the current value.
- **Read Value**
  Reflects current read value in point's OpcProxyExt.
- **Write Value**
  Reflects current write value (if any) in point's OpcProxyExt..
- **Opc Limit**
  Reflects the Opc limit for the data item.

Chapter 5

Opc Component Guides

These component guides provides summary help on Opc components.

Opc Component Reference Summary

Summary information is provided on components in the opc module, listed alphabetically as follows:

- OpcDaClient
- OpcDeviceFolder
- OpcNetwork
- OpcPointDeviceExt
- OpcPointFolder
- OpcPollScheduler
- OpcProxyExt
- OpcTuningPolicy
- OpcTuningPolicyMap

opc-OpcDaClient

OpcDaClient represents the client access to any version OPC Data Access server. It is the "device-level" component in the NiagaraAX OPC driver architecture. You use the OpcPointManager view of its *child* OpcPointDeviceExt extension to discover, add, and edit Opc proxy points.

For more details, see "About the OpcDaClient" on page 3-4.

opc-OpcDeviceFolder

OpcDeviceFolder is the Opc driver implementation of a folder under an OpcNetwork. Usage is optional. Each OpcDeviceFolder has its own OpcDeviceManager view.

You can use the **`New Folder`** button in the OpcDeviceManager view to add an OpcDeviceFolder. It is also available in the **`opc`** palette.

opc-OpcNetwork

OpcNetwork represents a tree of OPC devices and ancillary components, and is the top-level component for the Opc driver in a station. This network object is a NiagaraAX Framework convention, and has no physical correspondence to any OPC systems.

The OpcDeviceManager is the default view the OpcNetwork. For more details, see "About the Opc Network" on page 3-2.

opc-OpcPointDeviceExt

OpcPointDeviceExt (default name `Points`) is the container for Opc proxy points under an OpcDaClient. This component represents an "OPC Group," where all points contained under this component are members of that OPC Group.

The default and primary view for the OpcPointDeviceExt is the OpcPointManager. For more details, see "Opc Point Manager" on page 3-8.

opc-OpcPointFolder

OpcPointFolder is an optional container for Opc proxy points. It provides organizational utility, and has no correspondence with the underlying OPC system. Points can be orgainzed in any fashion under a OpcPointDeviceExt component.

You can use the **`New Folder`** button in the OpcPointManager view to add an OpcPointFolder. It is also available in the **`opc`** palette.

opc-OpcPollScheduler

An OpcPollScheduler is a child component of every (device-level) OpcDaClient. The poll scheduler provides a flexible polling algorthim based on four "buckets."

opc-OpcProxyExt

OpcProxyExt is the proxy extension for any type of Opc proxy point. For more details, see "Opc proxy point" on page 3-10 and "Opc Proxy Ext" on page 3-10.

opc-OpcTuningPolicy

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

opc-OpcTuningPolicyMap

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

Appendix A

Windows Tasks

This appendix provides procedures for Windows configuration tasks that may be needed at the OPC server and/or client. The following main sections are included:

Launching DCOMCNFG

DCOMCNFG is "Distributed COM Configuration Properties," and is used to configure security for DCOM applications. As needed, launch it either from Windows 2000 or Windows XP.

Note: *Starting in AX-3.4, NiagaraAX support for Windows 2000 ended. However, other Windows operating systems are supported, including Windows Server 2003. Please consider Windows XP, described below, as a "representative" Windows OS.*

Windows 2000

Step 1 From the Windows **Start** menu, choose **Run...**.

Step 2 In the **Run** dialog box, type in: `dcomcnfg`

and click **OK**.

Windows XP

Step 1 From the Windows **Start** menu, choose **Run...**.

Step 2 In the **Run** dialog box, type in: `dcomcnfg`

and click **OK**.

A **Component Services** dialog appears, with a **Console Root** tree in the left pane. Under the console root are **Component Services**, **Event Viewer (Local)**, and **Services (Local)**.

Windows XP Service Pack 2 (and later)

The following sections apply to using Windows XP Service Pack 2 and later:

Windows Firewall

Configuration of the Windows Firewall is the same for both client and server. For troubleshooting, it would be best to turn off the firewall, get everything working, and then turn it back on.

If the firewall is required, go to the firewall's **Exceptions** tab and review the following:

- All OPC client and server programs on the local machine need to be added to the Exceptions tab. The program to add for the OPC client is:
  NiagaraRelease`\nre\bin\niagarad.exe`
- Add TCP port 135.

DCOM Enhancements

After launching DOMCNFG in Windows XP, do the following:

Step 1 In the **Console Root** tree, expand **Component Services**, then **Computers**.

Step 2 Right-click **My Computer**, and select **Properties**.
A tabbed dialog **My Computer Properties** appears, including a tab labeled **COM Security**.

Step 3 Select **COM Security**.

Step 4 In **Access Permissions**, click **Edit Limits...**.

Step 5 For the ANONYMOUS LOGON user, check the **Remote Access** box and click **OK**.

Step 6 On the OPC Server, in **COM Security**:
In **Launch and Activation Permissions**, click **Edit Limits...**.

Step 7 Check the Remote boxes for the user Everyone.
If client and server are *not* on the same domain, you must add the ANONYMOUS LOGON user and check all **Allow** boxes.