Technical Document

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

July 8, 2014

Niagara^{AX} BACnet Guide

Confidentiality Notice

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation (“Tridium”). Such information, and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

Trademark Notice

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

Copyright and Patent Notice

Contents

Preface

- Bacnet FAQs
- BACnet terms
- Document Change Log

Bacnet FAQs

The following are frequently asked questions (FAQs) about the NiagaraAX Bacnet driver:

Q: Why are there two spelling variations used: BACnet and Bacnet?

A: The term Bacnet (vs. BACnet) is used throughout this document when referring to NiagaraAX components and the Bacnet driver itself. This reflects the actual Niagara component and view names, such as BacnetNetwork, BacnetDevice, Bacnet Device Manager, and so on. If the term BACnet is used, this refers to BACnet in a context wider than the Niagara Framework, that is the protocol defined by ASHRAE Standard 135.

Q: Why is it when I add proxy points for a device, that some proxy points are disabled by default?

A: By default, when you add a writable proxy point for a "commandable" type BACnet object (any object with a Priority_Array property), the proxy point's "Enabled" property is "`false`." This is simply a "safety technique," to help prevent unintended BACnet writes from Niagara.

Keep in mind if you add a writable proxy point for the Present_Value property of such a (commandable) BACnet object, that once enabled, that point's 16 priority levels map *directly* to the BACnet priority levels for that object. For example, if you invoke an "Emergency Override" action on the proxy point, it is written to the BACnet object at priority level 1 (the highest). Or, if you link the "In4" slot of the writable proxy to a Logic object, then slot 4 of the Priority_Array of the remote BACnet object will be controlled by Niagara (by the output of the Logic object).

For these type point(s) to work at all—including reading the value from the device, or writing the value to it from Niagara, you must manually set Enabled to "`true`." You can do this either in the initial **`Add`** dialog for these point(s), or in subsequent **`Edit`** dialogs (double-click on a point in Bacnet Point Manager).

Also, note that if there is a BACnet object that you want to limit Niagara writes to *one* particular priority level, instead of adding a writable proxy point for Present_Value, you could instead add one for a particular index (level) under the object's Priority_Array property.

Q: How do I make an object in the station appear as a BACnet object?

A: You "expose" objects in a station to BACnet, including chosen components, histories, and files, from one central location: the **`Export Table`** under the **`Local Device`** in the BacnetNetwork. For more details, see "Bacnet server configuration overview" on page 4-60.

Q: Why do some proxy points display differently after upgrading my system to AX-3.2 or later?

A: By default, proxy points no longer poll "additional" BACnet properties of a target object like Status_Flags and Priority_Array—however, you can configure proxy points for these (and other) additional polled properties. For more details, see "About Bacnet Polling" on page 3-17, and "Facets usage to poll additional properties" on page 3-40.

Q: Are there advantages to using "Bacnet virtual points" instead of proxy points?

A: For JACE platforms that have limited storage (Flash) capacity for their hosted station, use of Bacnet virtual points (starting in AX-3.2) can offer advantages over proxy points, at least for simple Px view monitoring. For more details, see "About Bacnet virtual points" on page 3-52.

BACnet terms

The following list of terms and abbreviations is specific to BACnet usage in NiagaraAX, and covers entries used in this document. For the definitive BACnet glossary of terms, refer to the ASHRAE publication: *Standard 135-2008 – BACnet® – A Data Communication Protocol for Building Automation and Control Networks*. For general NiagaraAX terms, see the Glossary in the *User Guide*.

APDU Application (layer) protocol data unit. Basically, a BACnet message—that is, a unit of data specified within any of the BACnet link-layer protocols, consisting of protocol control information and possibly application user data.

BACnet Building Automation and Control NETworking protocol (ANSI/ASHRAE Standard 135-2004). An open communication protocol standard conceived by a consortium of manufacturers and system users under the auspices of ASHRAE. Data is modeled as a common set of "objects", which can be accessed using a standard set of "services." See "Bacnet FAQs" about the spelling variation "Bacnet."

BACnet client Operation mode for a BACnet system or device, where it makes use of a BACnet device for some particular purpose via a service request instance.

BACnet server Operation mode for a BACnet system or device, where it provides a service to a requesting client.

BACnet/Ethernet Or B/ETH. BACnet over Ethernet, one of the original BACnet link-layer types. Since the introduction of BACnet/IP, its usage has become less common.

BACnet/IP Or B/IP. BACnet over Ethernet IP. Introduced in "Annex J" of the BACnet standard, it has become the most popular BACnet link-layer protocol (except in lowest-cost devices, where MS/TP is used).

BACnet MS/TP Or MSTP. BACnet link-layer protocol used by lower-cost devices, using master slave / token passing over RS-485 multidrop networks. QNX-based JACEs support direct MS/TP (network) trunks, one per RS-485 port (if licensed for MS/TP). See "Bacnet MS/TP licensing" on page 1-2.

BBMD BACnet/IP Broadcast Management Device. A device that receives and redistributes broadcast-type BACnet messages (Who-Is, I-Am, etc.) to other B/IP devices on its own subnet, and sends broadcast-type messages to BBMDs on other subnets. By having one BBMD on each subnet, a B/IP network can span subnets (between IP routers, which otherwise typically *block* broadcast type messages). A Niagara station supports operation as a BBMD. See "About BBMDs" on page C-3 for more details.

BTL BACnet Testing Laboratories, established by BACnet International to support compliance testing and interoperability testing of BACnet products. The `BACnet AWS Supervisor` and `BACnet OWS Supervisor` are each BTL-certified. See "BACnet AWS and OWS Supervisors" on page A-1.

Config object NiagaraAX can model BACnet objects in a client BACnet device as "Config" objects, where you can see all properties of the object together. For details, see "About Bacnet Config objects" on page 3-34.

COV Change-of-Value. BACnet provides services for COV reporting, on both the client and server sides. NiagaraAX supports COV reporting services on both sides.

Device object Or device component. In general NiagaraAX terms, any component representing an external device. Specific to a BacnetNetwork, each BacnetDevice is a container that represents a particular BACnet device. It has several device extensions, one being the Config Device Ext. By default, this Config extension contains a (frozen) Device Object component, which represents the single BACnet Device Object in that device. This component cannot be deleted, but other Config objects can be added and deleted as needed.

On the BACnet server side, a station's Device Object is represented by the configuration of its single Local Device component under its BacnetNetwork. See "Local Device notes" on page 4-78.

foreign device A BACnet/IP term for a BACnet device that exists on an IP subnet without a BBMD, where the device can "register" with a BBMD on another (remote) subnet as a "foreign device" to explicitly receive BACnet broadcast messages. In no way does it imply any reduced functionality. For more details, see "About BBMDs" on page C-3.

internetwork Two or more BACnet networks connected by a BACnet router, or essentially "everything on your site that can be accessed via BACnet", or whatever you can get to with a BACnet request.

The single BacnetNetwork in a station often represents an internetwork, for example if any external BACnet routers exist, or if the BacnetNetwork has multiple ports under its BacnetComm, Network component. If the latter, the station can act as a BACnet router between its local networks.

In a BACnet internetwork, each network must have a unique Network_Number, from 0 to 65534. Each BACnet device must have a Device object with a unique Object_ID, from 0 to 4194302.

For more details, see "Internetworks and BACnet/IP" on page C-1.

MS/TP Master Slave / Token Passing. See BACnet MS/TP.

object identifier Or object ID. A BACnet method to identify a particular object within a device, using a combination of its *object type* and an *instance number* (unique for that type, within that device). In the case of the single Device Object (type) per BACnet device, it must have a unique instance number across the BACnet internetwork on which it is installed.

Document Change Log

Updates (changes/additions) to this *Niagara^{AX} BACnet Guide* document are listed below.

- Updated: June 26, 2014
  Updated to describe changes in the BACnet driver made in AX-3.8 and AX-3.7 (since AX-3.6), and also to include previously omitted information on BACnet alarming, as well any reported corrections. The following document sections were modified or added:
  - In this "Document Change Log", removed entries prior to 2008.
  - In the "Bacnet Quick Start" subsection "Configure the Network ports", a correction was made in the procedure "To configure a BACnet/MSTP port" on page 2-6 ("Max Master" should be set to the *highest* known master on the network, with room for possible expansion).
  - In the main section "Niagara Bacnet Client Concepts", various subsections were updated, including "About bacnet palette components" on page 3-12 (shows AX-3.8 `bacnet` palette, with "WorkerPools" folder). Under the "About Bacnet Comm" section, most screen captures were updated to the AX-3.8/AX-3.7 look, and a new "Bacnet Comm: Server configuration" was added about possible configuration changes. Other updates were made in "About Bacnet Polling", primarily in subsection "Current Bacnet polling (AX-3.2 and later)". A new section "Bacnet Workers" was added, which includes "Location of Bacnet workers" on page 3-19 and "Worker pool expansion" on page 3-20. The section "About Bacnet Tuning Policies" had updates, including new subsections "Client COV notes" and "Changes in COV statusFlags reporting", the latter which explains changes since AX-3.7u1. A new section also describes a new property for a BacnetNetwork in AX-3.8. See "BacnetNetwork "uploadOnStart" property" on page 3-26. The section "About Bacnet Device Find Parameters" on page 3-27 was updated to show and describe "Select All" and "Clear All" buttons introduced in AX-3.7. The section "BacnetDevice properties" was updated, including new sections "Notes on other BacnetDevice properties" and "BacnetDevice "skipUpload" slot", the latter of which applies to AX-3.8 and AX-3.7 stations. Under the "Bacnet proxy points" section, in "Bacnet ProxyExt properties" on page 3-38, it was noted to *not change* the "Property Id" value of any existing proxy point. The subsection "Status merger for Bacnet proxy points" on page 3-39 section was updated, including mention of changed status behavior for proxy points using COV instead of polling. This was also mentioned in an update to the subsection "Facets usage to poll additional properties" on page 3-40.
  - In the main section "Niagara Bacnet Server Operation", a footnote was added to the "Bacnet server configuration overview" section, explaining that in some cases configuration outside of the **`LocalDevice`** may be necessary.
  - A new main "Niagara Bacnet Alarming" section was added, to provide basic information on alarm integration between NiagaraAX and BACnet. A main section is "Receiving BACnet alarms", including a subsection "Configuring to receive BACnet alarms" on page 5-84. The other main section is "Sending Alarms to BACnet", including a subsection "Configuring to send alarms to BACnet" on page 5-86.
  - In the "Bacnet Component Guides" subsection "Components in bacnet module" on page 7-93, summary entries were added for the BacnetAlarmDeviceExt component and new components (in AX-3.7u1) OutOfServiceExt, and in (in AX-3.8) BacnetWorkerPool. Component summaries support context-sensitive Workbench Help "Guide on Target", and often have links to more in-depth, contextual details in main sections.

- Updated: December 5, 2011
  Updated document to describe various changes made in the BACnet driver in AX-3.4, AX-3.5, and AX-3.6, as well as the new **BACnet AWS Supervisor** and **BACnet OWS Supervisor** products introduced in the 3.6.40.1 build of BACnet-related modules, as well as in a later AX-3.6 maintenance release. The following document sections were affected:
  - In the section "Bacnet Quick Start" on page 2-3, a Note was added explaining that quick start procedures also apply to any of the "BACnet Supervisor" products, with links to the appropriate sections that describe these products. Related Notes were added in quick start procedures.
  - In the main section"Niagara Bacnet Client Concepts" on page 3-11, a more comprehensive table of contents to subsections is provided, with various content changes or additions in some of these subsections. Changed areas include "About Bacnet Comm: Network: Router Table" on page 3-14, "About Bacnet Comm: Network: Ip Port" on page 3-14, and reordered content in "About Bacnet Polling" on page 3-17. In the "Bacnet Local Device" section, the subsection "About Local Device slots" on page 3-22 was edited, and a new section was added: "Time synchronization properties" on page 3-22. New "Bacnet Device Manager" subsections were added, including "Devices discovered in fault (orange)" on page 3-27, and AX-3.5 and later features "Timesynch (TSynch) function" on page 3-29 and "Device ID function" on page 3-29. Starting in build 3.6.40.1 of the BACnet driver, client support was added for Trend Log Multiple objects. This affected sections "About Bacnet Trend Logs (Histories)" on page 3-47, including subsections "About the Bacnet History Import Manager" on page 3-48, and new subsection "Bacnet Trend Multiple View" on page 3-51.
  - In the main section"Niagara Bacnet Server Operation" on page 4-59, a more comprehensive table of contents to subsections is provided, with various content changes or additions in some of these subsections. Changed areas include "About export folders" on page 4-61 and "Bacnet Export Manager" on page 4-63, including new subsections "Using export folders" on page 4-64, "Structured View Object server support" on page 4-64, "Adding new Bacnet export folders" on page 4-64, and "Svo Subordinate Manager view" on page 4-65. Other changes were made in "Local Device container slots" on page 4-79.
  - In the section"Bacnet Plugin Guides" on page 6-89, more plugin (view) summary descriptions were added, including "Views in bacnetAws module" on page 6-91 and "Views in bacnetOws module" on page 6-91. Summaries support context-sensitive Workbench "Help on View", and contain links to more in-depth, contextual details in main sections.
  - In the section"Bacnet Component Guides" on page 7-93, more component summary descriptions were added, including "Components in bacnetAws module" on page 7-104 and "Components in bacnetAws module" on page 7-104. Summaries support context-sensitive Workbench Help "Guide on Target", and have links to more in-depth, contextual details in main sections.
  - A new appendix "BACnet AWS and OWS Supervisors" was added, with main sections "BACnet OWS and AWS Supervisors overview" on page A-2, "BACnet AWS Supervisor features" on page A-4, "BACnet OWS Supervisor features" on page A-15, and "Converting to a BACnet AWS or OWS Supervisor station" on page A-18.
  - The previous appendix "BACnet Supervisor" (about a `bacnetws` module -based Supervisor) was retained, but now contains Notes about the BACnet Supervisor being deprecated starting in AX-3.6. A few additional Notes were added about BacnetDevice support starting in AX-3.5.
  - A new appendix "Internetworks and BACnet/IP" was added, providing details similar to those in the (R2) *Niagara BACnet Integration Reference*, but relative to a NiagaraAX station. Main sections include "BACnet Network Numbers" on page C-1, "Internetwork = BACnet Routers" on page C-2, "BACnet/IP and BBMDs" on page C-3, and "Example Internetwork Diagrams" on page C-8.
- Updated: February 26, 2008
  Changed document to reference the new *NiagaraAX Drivers Guide*, which was created from information formerly found in the NiagaraAX *User Guide*. Reworked and expanded the section "About Bacnet virtual points" on page 3-52, to describe the Bacnet virtual component redesign in AX-3.3 and later. Added new corresponding summary descriptions in the "Bacnet Component Guides" and "Bacnet Plugin Guides" sections.

CHAPTER 1

Bacnet Driver Installation

To use the NiagaraAX Bacnet driver, you must have a target JACE host that is licensed for the feature "`bacnet`," or a PC host acting as a BACnet Supervisor (meaning it is also licensed for `bacnet`). See the next section "Bacnet licensing considerations" for further details.

Note: *Link layer types BACnet/IP and BACnet/Ethernet are supported by any NiagaraAX platform. However, please note that BACnet MS/TP trunks are directly supported by QNX-based JACEs only, for example a JACE-6. Separate Bacnet MS/TP licensing is also required. Other platforms can indirectly support MS/TP devices through BACnet routers.*

The following sections provide details about Bacnet driver licensing and software installation:

- Bacnet licensing considerations
- Bacnet software installation

Bacnet licensing considerations

NiagaraAX Bacnet licensing is explained in the following sections:

- Bacnet client limits
- Bacnet server (export) ability
- Bacnet MS/TP licensing

Bacnet client limits

As in various other NiagaraAX drivers, the host's license may contain limits within the "`bacnet`" feature portion of your license. Maximum limits apply to the total number of devices, points, schedules, and histories supported. If any entry value is `"none"`, no set limits apply for that item type.

All limits apply directly to the "client" operation of the Bacnet driver, meaning the total number of BacnetDevices, Bacnet proxy points, imported Bacnet Schedules and Calendars, and imported Bacnet Trend Logs supported under the (single) BacnetNetwork. Therefore limits, if any, apply across all BacnetDevices and related child components in the station, regardless of the BacnetComm, Network port used to reach any device (IpPort, EthernetPort, MstpPort).

Limits do *not* apply to "server" operation of the Bacnet driver. See "Bacnet server (export) ability".

Bacnet server (export) ability

A single entry in the "`bacnet`" license feature determines whether a station can provide BACnet server functions. If enabled for server operation, this entry is:

```
export="true"
```

Note: *Any JACE ordered with the "Enterprise Connectivity Pack," required for Niagara communications to other JACEs and/or a Supervisor, should also have this entry enabling BACnet server operation.*

If instead the export value is `"false"`, no components, files, or histories are exported to BACnet, apart from the single BACnet Device Object that represents the station (as configured by the Bacnet Local Device). Although you can still use the various views of the Local Device's Export Table to add Bacnet export descriptors (and file and history descriptors), all descriptors will have a "fault status," with a fault cause of "Server capability not licensed."

For more details on BACnet server functions, see "Niagara Bacnet Server Operation" on page 4-59.

Bacnet MS/TP licensing

Bacnet MS/TP is a separately-licensed feature, requiring a feature named "mstp" in the JACE's license file. For example:

```
<feature name="mstp"
    expiration="never"
    port.limit="1"
    parts="DR-MSTP-AX"/>
```

Where port.limit defines the number of MS/TP trunks (JACE RS-485 ports) that can be used. This ranges from 1 to 4, varying on the type of the host JACE controller, and the license purchased.

The "EIA-485 (or RS-485) load factor" of connected MS/TP devices determines how many devices are physically supported, per trunk, due to electrical considerations. This ranges from 31 ("full load") to up to 127 ("quarter load") devices. Note that any other device limits in the license's bacnet feature, or platform limits (e.g. JACE-403, 27 total devices) also apply.

Bacnet software 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 ensures your Workbench has the needed distribution files (*.dist* files) for commissioning various models of remote JACE platforms. The dist files are located under your Niagara install folder in a "sw\dist" subfolder.

Apart from installing the 3.*n.nn* version of the Niagara distribution file in the JACE, make sure to install the bacnet module too (if not already present, or upgrade if an older revision). For more details, see the appropriate "Install and Startup Guide" for the target JACE platform.

Following this, the remote JACE is now ready for Bacnet configuration in its running station, as described in the rest of this document. See "Niagara Bacnet Client Concepts" on page 3-11 and "Niagara Bacnet Server Operation" on page 4-59, for details on how the Bacnet driver works.

Note: *Basic procedures for using the Bacnet client features of the driver, including online discovery of BACnet devices and contained objects (and their addition to your Niagara station) is in the next section, "Bacnet Quick Start".*

CHAPTER 2

Bacnet Quick Start

This section provides a collection of procedures to use the NiagaraAX Bacnet driver's client functions—the most typical usage of the NiagaraAX Bacnet driver. Like other NiagaraAX drivers, you can do most configuration from special "manager" views and property sheets using Workbench. These are the main subsections:

- "Configure the BacnetNetwork" on page 2-3
  - "Add a BacnetNetwork" on page 2-3
  - "Configure the LocalDevice" on page 2-4
  - "Configure the Network ports" on page 2-4
- "Create BacnetDevices" on page 2-6
  - "Using online Discover to add BacnetDevices" on page 2-6
  - "Manually adding BacnetDevice components" on page 2-7
- Create Bacnet proxy points
  - "Using online Discover to add Bacnet proxy points" on page 2-8
  - "Manually adding Bacnet proxy points" on page 2-9

Note: *These quick start procedures apply to a setting up a* **`BACnet AWS Supervisor`**, **`BACnet OWS Supervisor`**, *or (now deprecated)* **`BACnet Supervisor`** *station too.*

In these cases, however, the root network component is different, and not from the `bacnet` *module. The BacnetAwsNetwork is from the* `bacnetAws` *module, the BacnetOwsNetwork is from the* `bacnetOws` *module, and the BacnetWsNetwork is from the* `bacnetws` *module. For related details, see the appendixes in this document "BACnet AWS and OWS Supervisors" on page A-1 and "BACnet Supervisor" on page B-1.*

Configure the BacnetNetwork

To configure the BacnetNetwork, perform the following main tasks:

- Add a BacnetNetwork
- Configure the LocalDevice
- Configure the Network ports

Add a BacnetNetwork

Note: *As an alternative to the procedure below, you can simply copy (drag and drop) a* **`BacnetNetwork`** *from the* `bacnet` *palette in Workbench, placing it in the station's* **`Config`**, **`Drivers`** *container.*

This applies to a **`BACnet AWS Supervisor`** *station or* **`BACnet OWS Supervisor`** *station too, except in those cases you would copy either a* **`BacnetAwsNetwork`** *(from the* `bacnetAws` *palette) or* **`BacnetOwsNetwork`** *(from the* `bacnetOws` *palette). Only* one *Bacnet network of any type is supported.*

To add a BacnetNetwork in the station

Use the following procedure to add a BacnetNetwork under the station's Drivers container.

To add a BacnetNetwork 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 dialog. For more details, see "Driver Manager New and Edit" in the *Drivers Guide*.

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

This brings up a dialog to name the network.

Step 4 Click **`OK`** to add the BacnetNetwork to the station.

You should have a BacnetNetwork named “BacnetNetwork” (or whatever you named it), under your Drivers folder.

Configure the LocalDevice

To configure the Bacnet LocalDevice

You must define one essential Bacnet LocalDevice property—the Device object ID used by the station (see “An essential Local Device property” on page 3-21).

To configure this essential property, in the station with the BacnetNetwork (or BacnetAwsNetwork, BacnetOwsNetwork, or BacnetWsNetwork):

Step 1 Right-click the BacnetNetwork and select **`Views > Property Sheet`**.

This produces the network’s property sheet.

Step 2 In the property sheet, click to expand the **`Local Device`** (see Figure 3-14 on page 21).

Note: *You can alternately use the Nav side bar, and simply double-click Local Device.*

Step 3 In the numerical **`Object Id`** field, change the entry from “`-1`” to a valid, BACnet Device instance number, unique across the entire BACnet internetwork (range is 0 to 4,194,302).

Note: *Make sure you are using an instance number that is NOT being used by any other BACnet device on the job!*

Step 4 Click the **`Save`** button.

Above the **`Object Id`**, the **`Status`** should change from `{fault}` to `{ok}`, and the **`Fault Cause`** should change from “Invalid Object ID” to blank.

Step 5 In the property sheet, click to collapse the **`Local Device`**.

See the next section: “Configure the Network ports”.

Configure the Network ports

By default, a new BacnetNetwork contains a **`Bacnet Comm`** component supporting BACnet/IP. Regardless of BACnet port usage, some minimal configuration is needed. For more details than given in the following procedures, see “About Bacnet Comm” on page 3-12.

Use the following quick start procedures to configure Bacnet network ports:

- To access the Bacnet Comm network port(s)
- To add a Bacnet Comm network port(s)
- To configure a BACnet/IP port
- To configure a BACnet/Ethernet port
- To configure a BACnet/MSTP port

To access the Bacnet Comm network port(s)

Do this to add one or more additional ports, or to access the default BACnet/IP port.

To access the Bacnet Comm network ports:

Step 1 In the BacnetNetwork property sheet, click to expand **`Bacnet Comm`**, then **`Network`**.
Or, select these items using the Nav side bar (see Figure 3-2 on page 13).

Step 2 Depending on your BACnet installation, you can do several things:

- If you are connecting to a BACnet/IP network, you can use the default IP port. See “To configure a BACnet/IP port” on page 2-5.
- If you are connecting to a BACnet/Ethernet or BACnet/MSTP network, you must add the appropriate NetworkPort for that type of network. “To add a Bacnet Comm network port(s)”.

To add a Bacnet Comm network port(s)

If needed, in addition to an IpPort, you can add one EthernetPort (per adapter), and/or up to four MstpPorts.

You can also add additional IpPorts, to enable communications on additional UDP ports, such as 47809 (0xBAC1), or whatever UDP ports may also be used. Note in this case, the station is acting as BACnet/IP router between two logical BACnet/IP networks (that actually exist on the same physical network).

To add a Bacnet Comm network port:

Step 1 Open the Bacnet Comm, Network. See “To access the Bacnet Comm network port(s)”).

Step 2 Open the **`bacnet`** palette in the Workbench palette side bar (Figure 3-1 on page 12). See “Using the palette side bar” in the *User Guide* for general details.

Step 3 In the bacnet palette, expand the **NetworkPorts** folder.

Step 4 *Drag* the port type you want from palette into the Network container (or, use copy and paste instead). In the popup **Name** dialog, you can rename the port—or, simply use the default name.

Step 5 Click **OK** to add the network port.

If needed, repeat to add additional Bacnet network ports.

As needed, see the following related procedures:

- To configure a BACnet/IP port
- To configure a BACnet/Ethernet port
- To configure a BACnet/MSTP port

To configure a BACnet/IP port

You need to perform a few essential configuration items.

To configure a BACnet/IP port:

Step 1 Expand the **Ip Port** in the property sheet view.

Step 2 Set the **Network Number** from "-1" to the BACnet network number for the network segment to which you are connecting.

- If this is an existing BACnet installation, make sure to use the same network number already in use.
- If this is a new BACnet installation, you can choose this number (for example: 1).

Step 3 Expand the **Link** component in the property sheet.

Step 4 Under **Link**, use the drop-down list for **Adapter** to select the network adapter to use. These adapters are automatically discovered and validated by Niagara.

Step 5 Review the **Udp Port** setting being used for BACnet communications.

By default, UDP port 47808 decimal (0xBAC0 in hexadecimal) is used. If your BACnet/IP installation is using another UDP port, enter this port number in the Udp Port property.

Step 6 Review the **Ip Device Type** setting for the station. By default, the station will operate as a "Standard" BACnet/IP device. Alternatively, you can specify the station to function as either a "Foreign Device" or a "Bbmd" (BBMD), to allow BACnet/IP operation through standard IP routers.

Note: *Only one BACnet/IP device on the same IP subnet should operate as a BBMD. If a BBMD already exists on the host's subnet, the "Standard" device type is appropriate—leave remaining properties at defaults. The existing BBMD will distribute all BACnet broadcast messages to/from the remote BBMDs.*

- If configuring the station as either a Foreign Device or BBMD, you must know the IP address and UDP port used by one other (*remote* subnet) BBMD.
  Enter that combination in the **Bbmd Address** property. For example, a **Bbmd Address** value of 172.16.8.26:47808 (using a ":" between IP address and UDP port.)
- Also, if configuring as a Foreign Device, review the **Registration Lifetime** value, which defaults to 15 minutes. Typically this value is ok; however, some BBMDs may require another value.

For details about BACnet/IP BBMDs and Foreign Devices, see "BACnet/IP and BBMDs" on page C-3.

Step 7 Click the **Save** button to save **Ip Port** changes.

Step 8 Right-click the **Ip Port** (in property sheet or Nav side bar), and select **Actions > Enable**.

To configure a BACnet/Ethernet port

You need to perform a few essential configuration items.

To configure a BACnet/Ethernet port:

Step 1 Expand the **EthernetPort** in the property sheet view.

Step 2 Set the **Network Number** from "-1" to the BACnet network number for the network segment to which you are connecting.

- If an existing BACnet installation, make sure to use the same network number already in use.
- If a new BACnet installation, you can choose this number (for example: 2).

Step 3 Expand the **Link** component in the property sheet.

Step 4 Under Link, use the drop-down list for **Adapter Title** to select the Ethernet adapter to use. These adapters are automatically discovered and validated by Niagara. The properties "Adapter Description" and "Adapter Name" provide additional information about a selected adapter.

Step 5 Click the **Save** button to save EthernetPort changes.

Step 6 Right-click the **EthernetPort** (in property sheet or Nav side bar), and select **Actions > Enable**.

To configure a BACnet/MSTP port

Note: *BACnet MSTP is supported only with a station running on a QNX-based JACE controller.*

You need to perform a few essential configuration items.

To configure a BACnet/MSTP port:

Step 1 Expand the **MstpPort** in the property sheet view.

Step 2 Set the **Network Number** from "-1" to the BACnet network number for the network segment to which you are connecting.

- If an existing BACnet installation, make sure to use the same network number already in use.
- If a new BACnet installation, you can choose this number (for example: 3).

Step 3 Expand the **Link** component in the property sheet.

Step 4 Under Link, perform the following configuration:

- Set the **Port Name** to the RS-485 port to be used on the JACE. For example, if JACE with a dual RS-485 port option card, i.e. three total RS-485 ports), one of the following: COM3, COM4 or COM5.
- Set the **Mstp Address** to a unique BACnet MAC address on that MSTP trunk, in decimal, with valid range from 0 (default) to 127. Each BACnet device on the MS/TP network segment must have a unique MAC address.
  Note: *Typically, you leave the Mstp Address at 0 (the default), and verify that no other MS/TP device on the trunk is addressed the same. If there is ever a "lost token," the device with the lowest MAC address regenerates the token (and in this case it will be the station).*
- Set **Max Master** to the *highest* known master device on the network, with possible room for expansion if needed. Also, the **Max Info Frames** property controls how many messages are sent before passing the token, and may be increased up to 50 to increase performance in some cases.

Step 5 Click the **Save** button to save MstpPort changes.

Step 6 Right-click the **MstpPort** (in property sheet or Nav side bar), and select **Actions > Enable**.

Create BacnetDevices

After configuring the BacnetNetwork network port(s), you can use online discovery to find and create BacnetDevice components under the BacnetNetwork. Or, you can add BacnetDevice components to the station manually. You use the BacnetNetwork's default **Bacnet Device Manager** view.

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

- Using online Discover to add BacnetDevices
- Manually adding BacnetDevice components

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

Using online Discover to add BacnetDevices

If the JACE is connected to the BACnet device network(s), this is the easiest way to accurately populate the station with the necessary configured BacnetDevice objects. Use the following procedures:

- To discover BACnet Devices
- To add discovered BACnet Devices

To discover BACnet Devices

Perform this task to discover BACnet devices.

To discover BACnet devices:

Step 1 In the Nav side bar, right-click the **BacnetNetwork** and select **Views > Bacnet Device Manager** to bring up the **Bacnet Device Manager**.

Step 2 Click the **Discover** button to automatically learn what devices are on the network.

A popup **Configure Device Discovery** dialog appears. By default, discovery occurs for all possible BACnet devices connected on locally connected networks. If needed, you can make changes before initiating the discovery. See "About Bacnet Device Find Parameters" on page 3-27.

Step 3 Click **OK** to initiate the discovery process.

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

Step 4 When the discovery job completes, discovered BACnet devices are listed in the *top pane* of the view, in the "Discovered" table. The bottom pane, labeled "Database," is a table of devices that are currently mapped into the Niagara station—initially, this table will be empty.

Note: *This works the same as in most Device Manager views. For details, see the section "About Device Discover, Add and Match (Learn Process)" in the* Drivers Guide*.*

To add learned devices, see the next section: "To add discovered BACnet Devices".

To add discovered BACnet Devices

Perform this task to add discovered BACnet devices to your station database.

To add discovered BACnet devices:

Step 1 You can map a discovered device in the station in a number of ways:

- Drag it from the Discovered pane to Database pane (brings up an **Add** dialog).
- Double-click it in the Discovered pane (also brings up an **Add** dialog).
- Click to highlight in the Discovered, then press "a". ("Quick Add", meaning *no* **Add** dialog).

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

Step 2 When the **Add** dialog appears, you can edit the configuration of the BacnetDevice object before it is added in the Niagara station. Initial property values are determined from the device (and are typically acceptable).

- For Bacnet-specific details, see "BacnetDevice properties" on page 3-31.
- For general details, see the section "About Device Discover, Add and Match (Learn Process)" in the *Drivers Guide*.

Step 3 When you have a BacnetDevice component configured properly for your usage, click **OK**.

The BacnetDevice is added to the station, and appears listed in the Database pane—and is now dimmed in the Discovered pane.

Manually adding BacnetDevice components

To manually add a BacnetDevice, you can use the Device Manager's **New** device wizard, or drag (or copy) a BacnetDevice from the **bacnet** palette. You can also use the *Match* function in the Device Manager to match a manually added (or duplicated device) with a discovered device.

The following procedures explain how:

- To add a BacnetDevice using New device wizard
- To copy a BacnetDevice from the bacnet palette
- To match a BacnetDevice to a discovered device

To add a BacnetDevice using New device wizard

You can perform this task even when the JACE station is not communicating on BACnet networks.

Note: *This works the same as in other driver's Device Manager views. For general details, see the section "Device New Folder and New" in the* Drivers Guide*.*

To add BacnetDevices using New device wizard:

Step 1 In the Bacnet Device Manager, click the **New** button.

This brings up the first New device wizard dialog.

- Select `BacnetDevice` in "Type to Add."
- Enter the number of devices to add (default value is `1`).
- Click **OK**.

Step 2 When the next **New** dialog appears, you can edit the configuration of the BacnetDevice object before it is added in the Niagara station. This New dialog is like the **Add** and **Edit** device dialog, but in this case values are *not pre-populated* from any discovery.

Configure these properties according to the device you are attempting to connect with. For specific details, see "BacnetDevice properties" on page 3-31.

Step 3 When you have a BacnetDevice component configured properly for your usage, click **OK**.

The BacnetDevice is added to the station, and appears listed in the Database pane.

To copy a BacnetDevice from the bacnet palette

You can perform this task even when the JACE station is not communicating on BACnet networks. See "Using the palette side bar" in the *User Guide* for general palette details.

To copy a BacnetDevice from the bacnet palette:

Step 1 Open the **bacnet** palette in the Workbench palette side bar (Figure 3-1 on page 12).

Step 2 Drag the BacnetDevice from the palette into the Database pane of the Bacnet Device Manager.

A popup **Name** dialog prompts for the Niagara name for this BacnetDevice.

Step 3 Enter whatever name you need, then click **OK**.

The BacnetDevice is added to the station, and appears listed in the Database pane.

Step 4 To edit the device, double-click it for the **Edit** dialog. This dialog is like the **Add** device dialog, but in this case values are *not pre-populated* from any discovery.

Configure these properties according to the device you are attempting to connect with. For specific details, see "BacnetDevice properties" on page 3-31.

To match a BacnetDevice to a discovered device

If online with the BACnet network, you can *match* a manually added BacnetDevice to a discovered device using the Device Manager's **Match** button. This copies essential configuration (learned in the auto-discovered device) into the selected BacnetDevice component.

This is useful if you previously manually added the BacnetDevice (using New device, or copied from the bacnet palette). Also, this can be useful if you saved a fully-engineered device locally in your Workbench, and wish to *duplicate* it for a network of identical devices. For more details, see the section "Match (Device)" in the *Drivers Guide*

To match an existing BacnetDevice (in Database pane) to a discovered device:

Step 1 If the Discovered device table is empty, perform a Discover. See "To discover BACnet Devices" on page 2-6.

Step 2 Click to select (highlight):

- *One* device in the *Discovered* table.
- *One* device in the *Database* table.

Step 3 Click the **Match** button, or type Ctrl + m, to match the devices.

This brings up the **Match** dialog, which is like the **Add** device dialog, where values are pre-populated from the discovery. The only difference is that the Type field cannot be edited.

Note: *Alternately, you can simply type m for a "Quick Match," whereby the* **Match** *dialog is bypassed and the match is made in the station database.*

Step 4 As needed in the **Match** dialog, edit any property required. For specific details, see "BacnetDevice properties" on page 3-31.

Step 5 Click **OK** to match the discovered device to the BacnetDevice component.

The discovered device now appears dimmed, indicating it is already represented in the station.

Create Bacnet proxy points

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

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

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

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

Using online Discover to add Bacnet proxy points

If the JACE is connected to the BACnet device network(s), this is the easiest way to accurately add Bacnet proxy points under a BacnetDevice. Use the following procedures:

- To discover BACnet objects
- To add discovered BACnet objects as Bacnet proxy points

Note: *This works the same as in most Point Manager views. For details, see the section "About Point Discover, Add and Match (Learn Process)" in the* Drivers Guide*.*

To discover BACnet objects

Perform this task to discover Bacnet objects as proxy point candidates.

To discover Bacnet objects in a device:

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

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

Step 2 Click the **Discover** button to automatically learn what BACnet objects are in the device. Niagara interrogates the device for its object list, and a progress bar appears at the top of the view.

When the discovery job completes, discovered BACnet objects are listed in the *top pane* of the view, in the "Discovered" table. Each BACnet object in the device occupies one row.

You can click to expand objects (by default, "presentValue" is on top of any object that has it). See "Bacnet Point Manager "Discovered" notes" on page 3-34 for more details.

Step 3 To proxy learned objects, see the next section: "To add discovered BACnet objects as Bacnet proxy points".

To add discovered BACnet objects as Bacnet proxy points

Perform this task to add discovered BACnet objects to your station database as proxy points.

To add discovered BACnet objects as Bacnet proxy points:

Step 1 Select the property of the discovered object you wish to proxy. Typically, this is the default (top) "presentValue" property, but you may wish to create one or more additional proxy points for other properties (expand the discovered object). Possible examples include "eventState," or for a priority-type object, a particular "priorityArray" index (level).

Step 2 You can map selected items in the station in a number of 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).
- Click to select in Discovered, then press "a". ("Quick Add", meaning *no* **Add** dialog).

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

Step 3 When the **Add** dialog appears, you can edit the configuration of the proxy point's BacnetProxyExt before it is added in the Niagara station. Initial property values are determined from the learned object (and are typically acceptable).

Note the following about entries in the Add dialog:

- **Name** is the Object_Name as reported by the BACnet object, plus Property_Indentifier to ensure a unique point name. For example: "`AHU1-Frz_Stat-eventState`"
  The exception to this is for "presentValue" points (default), where only Object_Name is used in point name. For example: "`AHU1-Frz_Stat`"
  Note: *Each slash ("/") character, if any, is replaced by a period ("*.*") in the Niagara name.*
  This is the Niagara point name only—any change is not written to the BACnet object.
- **Type** is the Niagara control point type to use for the proxy point.
  Note: *Unlike other entries in the Add dialog, you cannot edit Type later.*
- **Enabled** is whether the proxy point is enabled for polling, writing, etc.
  Note: *By default, any writable Type point (BooleanWritable, NumericWritable, etc.) for a "commandable" type object (any with a priority_array property, such as a Binary_Output, Analog_Output, etc), will have Enabled initially set to* `false`*. To allow for point operation, you must set this property to* `true`*. See "Bacnet FAQs" on page 2-3 for more details.*
- Object ID, Property ID, and Index are how the point references its value in the remote BACnet device.
- **Tuning Policy Name** specifies the Bacnet tuning policy to use for the proxy point.
- **Data Type** specifies the ASN data type of the property in the BACnet object. Values are automatically converted to the appropriate Niagara type for the point. For example, Analog Input Present_Value is an ASN REAL, but can be interpreted by a StringPoint as a character string.
- Read and Write display the read and write status of the point.
- **Device Facets** represent the facets learned from the device.
- **Facets** represent the parent Niagara proxy point's facets, for how the value should be displayed in Niagara.

Step 4 When you have a Bacnet 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 "Bacnet Point Manager "Database" notes" on page 3-36.

Manually adding Bacnet proxy points

You can manually add Bacnet proxy points, using the New button in the Bacnet Point Manager, or by dragging from the bacnet palette.

CHAPTER 3

Niagara Bacnet Client Concepts

This section describes the “client side” operation of the NiagaraAX Bacnet driver, the most commonly used NiagaraAX implementation with BACnet. These are the main subsections:

About Bacnet Network Architecture

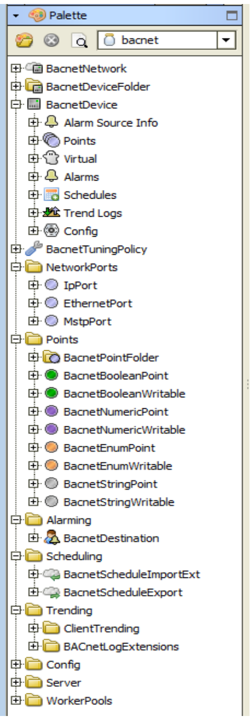
The Bacnet driver uses the standard NiagaraAX network architecture. See “About Network architecture” in the *Drivers Guide* for more details. However, a station’s BacnetNetwork is unique because of simultaneous support for *multiple* BACnet link layer types, using different BACnet communications protocols: BACnet/IP, BACnet/Ethernet, (and if QNX-based JACE host platform, BACnet MS/TP). Thus, depending on configuration, a BacnetNetwork may proxy BACnet devices that not only reside in *different* BACnet networks, but are also accessed using different physical ports on the host JACE controller.

For more details, see “About Bacnet Comm” on page 3-12.

About bacnet palette components

In Workbench, open the **bacnet** palette to see the various Bacnet components (Figure 3-1).

Figure 3-1 *Components in bacnet palette (AX-3.8 shown)*



Palette

bacnet

- BacnetNetwork
- BacnetDeviceFolder
- BacnetDevice
  - Alarm Source Info
  - Points
  - Virtual
  - Alarms
  - Schedules
  - Trend Logs
  - Config
- BacnetTuningPolicy
- NetworkPorts
  - IpPort
  - EthernetPort
  - MstpPort
- Points
  - BacnetPointFolder
  - BacnetBooleanPoint
  - BacnetBooleanWritable
  - BacnetNumericPoint
  - BacnetNumericWritable
  - BacnetEnumPoint
  - BacnetEnumWritable
  - BacnetStringPoint
  - BacnetStringWritable
- Alarming
  - BacnetDestination
- Scheduling
  - BacnetScheduleImportExt
  - BacnetScheduleExport
- Trending
  - ClientTrending
  - BACnetLogExtensions
- Config
- Server
- WorkerPools

Even without showing the many other components under the **Config** and **Server** folders, this demonstrates that Bacnet is one of the more comprehensive NiagaraAX field bus drivers. However (like most NiagaraAX drivers), you *seldom* need to work from the palette. Instead, the various Bacnet manager views simplify component creation, enforcing proper component hierarchy.

The few scenarios where you *may* need to work from the `bacnet` palette including the following:

- When adding another network port, found under folder `NetworkPorts`.
- When adding a specialized history extension (BacnetLogExtension) to any point in the station, in order to make its Niagara history exportable as a BACnet-compliant Trend Log object. See “About BacnetTrendLogExt extensions” on page 4-74 for more details.
- When adding a BacnetDestination to the station’s AlarmService, found under folder `Alarming`.
- When adding a “WorkerPool” under a “Worker” slot (AX-3.8 only). See “Bacnet Workers” on page 3-19 for related details.

About Bacnet Comm

Under the BacnetNetwork, the Bacnet Comm container configures the “protocol stack” used by Niagara for all BACnet communications. Bacnet Comm has a **Comm Control** property that you must enable for any BACnet communications.

In addition, Bacnet Comm has several child components, discussed in the following subsections:

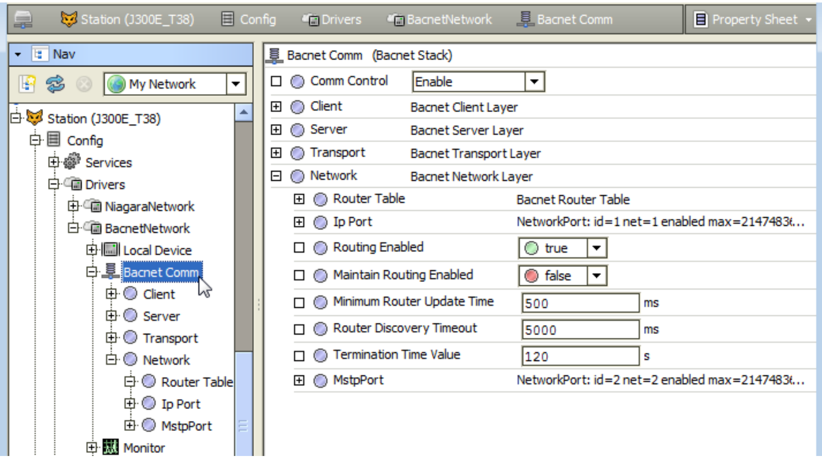
- About Bacnet Comm: Network
- About Bacnet Comm: Network: Router Table

- About Bacnet Comm: Network: Ip Port
- About Bacnet Comm: Network: Ethernet Port
- About Bacnet Comm: Network: Mstp Port
- About Bacnet Comm: Client, Server, and Transport
- Bacnet Comm: Server configuration
- About Bacnet Polling

About Bacnet Comm: Network

The Network container under Bacnet Comm determines BACnet network-layer configuration for the station. This includes whatever port/protocols will be used (BACnet/IP, BACnet/Ethernet, and if a QNX-based host, BACnet MS/TP), as well as operation as a BACnet router (if multiple network ports will be used). You can access Bacnet Comm directly in the Nav side bar, as shown in Figure 3-2.

Figure 3-2 *Bacnet Comm access using Nav side bar*



Bacnet Comm (Bacnet Stack)

Comm Control	Enable
Client	Bacnet Client Layer
Server	Bacnet Server Layer
Transport	Bacnet Transport Layer
Network	Bacnet Network Layer
Router Table	Bacnet Router Table
Ip Port	NetworkPort: id=1 net=1 enabled max=2147483...
Routing Enabled	true
Maintain Routing Enabled	false
Minimum Router Update Time	500 ms
Router Discovery Timeout	5000 ms
Termination Time Value	120 s
MstpPort	NetworkPort: id=2 net=2 enabled max=2147483...

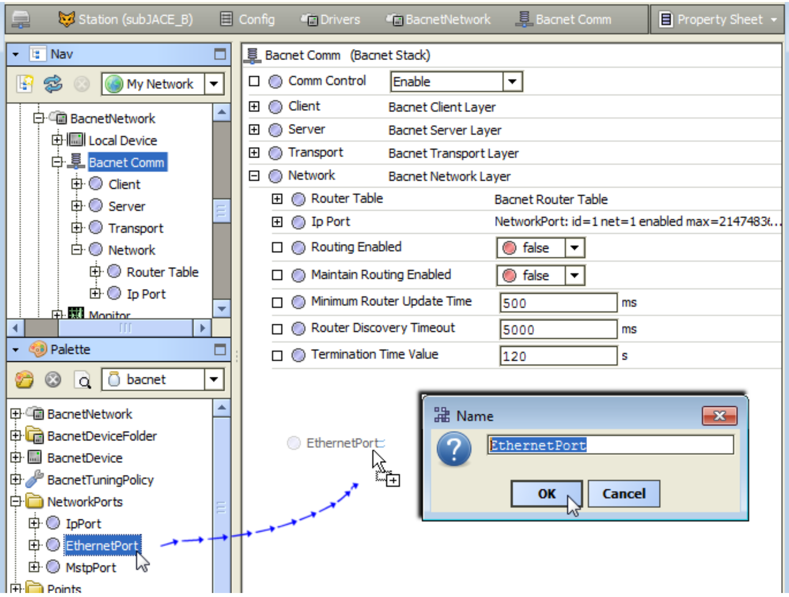
Configuration properties determine if routing is enabled, as well as other routing-specific parameters.

By default, a newly added BacnetNetwork will have an `Ip Port` under its Bacnet Comm—the most popular BACnet protocol (BACnet/IP).

If needed, you can copy another network port into the Network container from the `bacnet` palette, as shown being done in Figure 3-3.

Note: *Each port has "Status" and "Fault Cause" properties that can help when troubleshooting.*

Figure 3-3 *Adding Network port from bacnet palette*



Bacnet Comm (Bacnet Stack)

Comm Control	Enable
Client	Bacnet Client Layer
Server	Bacnet Server Layer
Transport	Bacnet Transport Layer
Network	Bacnet Network Layer
Router Table	Bacnet Router Table
Ip Port	NetworkPort: id=1 net=1 enabled max=2147483...
Routing Enabled	false
Maintain Routing Enabled	false
Minimum Router Update Time	500 ms
Router Discovery Timeout	5000 ms
Termination Time Value	120 s

Name: EthernetPort

Note: *Whatever port type (default IP included), all ports require a unique Network Number across the entire BACnet internetwork. The default Network Number value is -1 (inoperative). Ensure that each network port has a unique, positive Network Number for proper operation.*

The valid BACnet Network_Number *range is from 1 to 65534.*

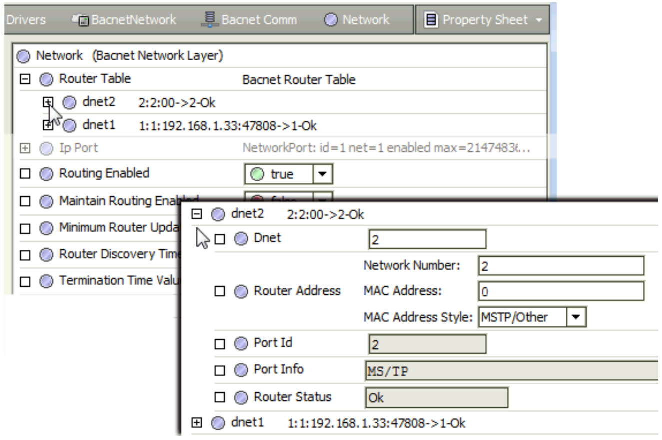
Existing BACnet network(s) note:

- If adding a station on an existing BACnet internetwork, specify the established BACnet/IP Network_Number and/or BACnet/Ethernet Network_Number currently in use.
- If a JACE with one or more enabled MS/TP ports, specify a previously *unused* network number for each MstpPort (RS-485) port.

About Bacnet Comm: Network: Router Table

If using more than one **`Bacnet Comm`** network port, and "Routing Enabled" is true in the parent **`Bacnet Comm`**, **`Network`** slot, corresponding router table entries automatically populate under this container. Router entries are listed by "dnet" (destination network). See Figure 3-4.

Figure 3-4 *Router Table in property sheet example*



Network (Bacnet Network Layer)
⊟ Router Table — Bacnet Router Table
⊞ dnet2 2:2:00->2-Ok
⊞ dnet1 1:1:192.168.1.33:47808->1-Ok
⊞ Ip Port — NetworkPort: id=1 net=1 enabled max=2147483...
□ Routing Enabled — true
□ Maintain Routing Enab
□ Minimum Router Upda
□ Router Discovery Tim
□ Termination Time Valu

⊟ dnet2 2:2:00->2-Ok
□ Dnet — 2
□ Router Address — Network Number: 2; MAC Address: 0; MAC Address Style: MSTP/Other
□ Port Id — 2
□ Port Info — MS/TP
□ Router Status — Ok
⊞ dnet1 1:1:192.168.1.33:47808->1-Ok

After device discovery from the **`Bacnet Device Manager`**, where the "All Networks" option was chosen, *additional* router table entries may exist. These reflect other discovered remote BACnet routers. Such entries may populate following an "All Networks" discovery even if "Routing Enabled" is false. Entries like this may also appear resulting from unsolicited messages received from remote networks.

After 24 hours of no traffic directed to a network, router table entries are *cleared*, to prevent the table from being cluttered with old entries. As needed, router table entries are discovered again.

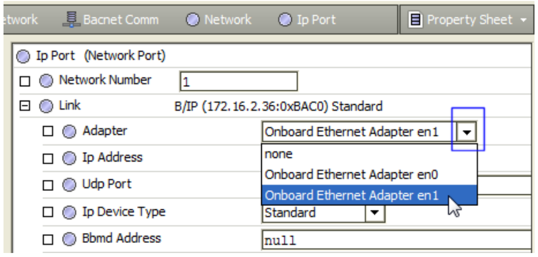
If the station detects a BACnet "router loop", the BacnetComm, Network's "Routing Enabled" may be automatically set to false. If this case, the station stops acting as a BACnet router for its enabled network ports. Corresponding entries are generated in the stations's LogHistory, noting detected misconfiguration and disabled BACnet router functionality.

If needed, you can override this behavior by setting the "Maintain Routing Enabled" property to true, and then resetting "Routing Enabled" back to true.

About Bacnet Comm: Network: Ip Port

The **`Ip Port`** container provides for BACnet Annex J communications as a BACnet/IP device. Note that most JACE controllers provide *two* Ethernet adapters: LAN1 (en0) or LAN2 (en1), therefore you should select the desired one in the Adapter property, as shown in Figure 3-5.

Figure 3-5 *Selecting Ethernet adapter to use for BacnetComm: Network: IpPort*

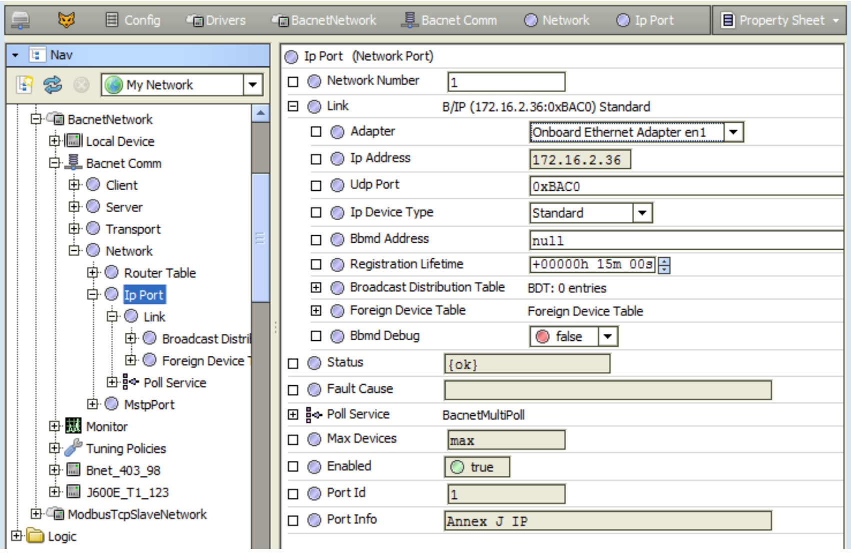


Ip Port (Network Port)
□ Network Number — 1
⊟ Link — B/IP (172.16.2.36:0xBAC0) Standard
□ Adapter — Onboard Ethernet Adapter en1
□ Ip Address
□ Udp Port
□ Ip Device Type — Standard
□ Bbmd Address — null

Link binding is automatic to the IP address and the selected Ethernet adapter of the host Niagara platform. By default, the “conventional” UDP port 0xBAC0 (decimal 47808) is used, however, you can specify using another UDP port.

Figure 3-5 shows the **`Ip Port`** property sheet with its **`Link`** container expanded.

Figure 3-6 *Property sheet view of Ip Port, Link container expanded*



Ip Port (Network Port)

Network Number	1
Link	B/IP (172.16.2.36:0xBAC0) Standard
Adapter	Onboard Ethernet Adapter en1
Ip Address	172.16.2.36
Udp Port	0xBAC0
Ip Device Type	Standard
Bbmd Address	null
Registration Lifetime	+00000h 15m 00s
Broadcast Distribution Table	BDT: 0 entries
Foreign Device Table	Foreign Device Table
Bbmd Debug	false
Status	{ok}
Fault Cause	
Poll Service	BacnetMultiPoll
Max Devices	max
Enabled	true
Port Id	1
Port Info	Annex J IP

As with other Bacnet network ports, the Ip Port provides its own, independent Poll Scheduler (**`Poll Service`**). See “About poll components” in the *Drivers Guide* for more details. Also see “About Bacnet Polling” on page 3-17.

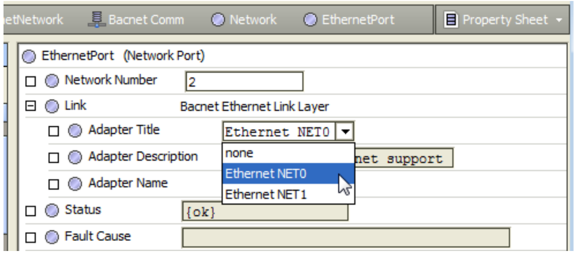
Link property “IP Device Type” allows for B/IP operation as a BBMD (BACnet Broadcast Management Device) or BACnet Foreign Device, or a standard BACnet device (the default). Remaining properties support further configuration as a BBMD or Foreign Device. For related BBMD and Foreign Device details, see “BACnet/IP and BBMDs” on page C-3.

About Bacnet Comm: Network: Ethernet Port

If needed, you can add an Ethernet Port from the `bacnet` palette (see Figure 3-3 on page 13). The EthernetPort container provides for BACnet/Ethernet communications.

Note that most JACE controllers provide *two* Ethernet adapters: LAN1 (NET0) or LAN2 (NET1), therefore you should select the desired one in the Adapter property, as shown in Figure 3-7.

Figure 3-7 *Selecting Ethernet adapter to use for BacnetComm: Network: EthernetPort*

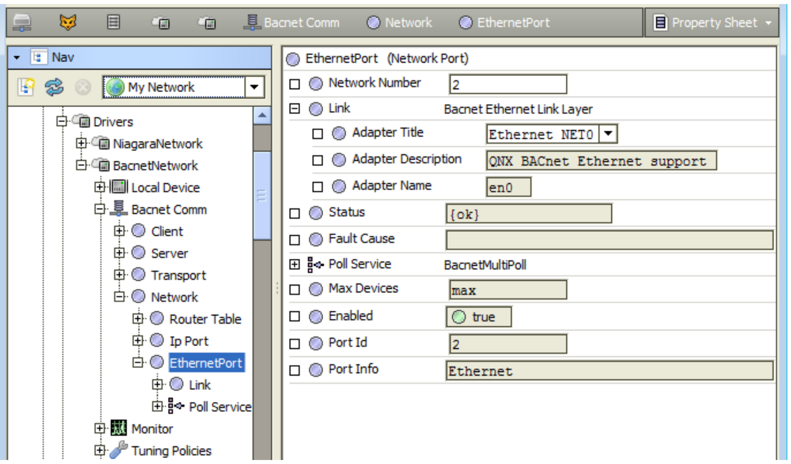


EthernetPort (Network Port)

Network Number	2
Link	Bacnet Ethernet Link Layer
Adapter Title	Ethernet NET0
Adapter Description	net support
Adapter Name	
Status	{ok}
Fault Cause	

Link binding is automatic to the selected Ethernet adapter of the host Niagara platform. Figure 3-8 shows the **`Ethernet Port`** property sheet with its **`Link`** container expanded.

Figure 3-8 *Property sheet view of Ethernet Port, Link container expanded*



EthernetPort (Network Port)

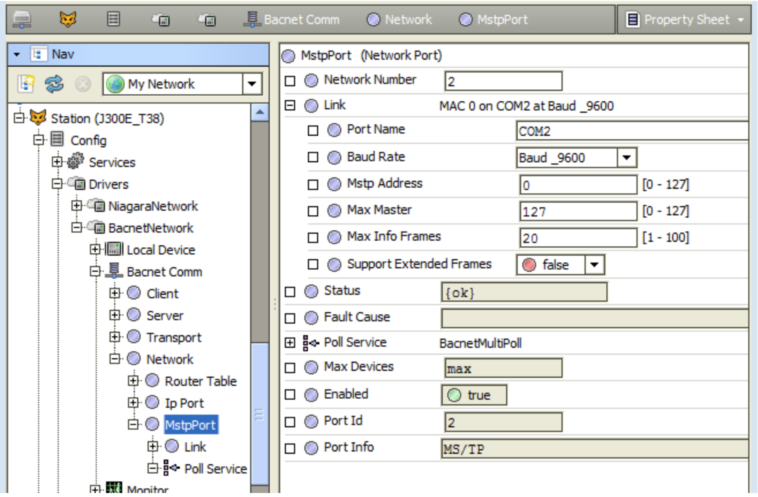
Network Number	2
Link	Bacnet Ethernet Link Layer
Adapter Title	Ethernet NET0
Adapter Description	QNX BACnet Ethernet support
Adapter Name	en0
Status	{ok}
Fault Cause	
Poll Service	BacnetMultiPoll
Max Devices	max
Enabled	true
Port Id	2
Port Info	Ethernet

As with other Bacnet network ports, the Ethernet Port has its own, independent Poll Scheduler (`Poll Service`). See “About poll components” in the *Drivers Guide* for more details. Also see “About Bacnet Polling” on page 3-17.

About Bacnet Comm: Network: Mstp Port

If a QNX-based JACE station, and if licensed for BACnet MS/TP (see “Bacnet MS/TP licensing” on page 1-2), you can add one or more **`Mstp Ports`** from the `bacnet` palette (see Figure 3-3 on page 13). Each **`Mstp Port`** container provides for BACnet MS/TP communications as a “master” type device, and supports one network *trunk* of BACnet MS/TP devices. See Figure 3-9 for a property sheet.

Figure 3-9 *Property sheet view of Mstp Port, Link container expanded*



MstpPort (Network Port)

Network Number	2	
Link	MAC 0 on COM2 at Baud _9600	
Port Name	COM2	
Baud Rate	Baud _9600	
Mstp Address	0	[0 - 127]
Max Master	127	[0 - 127]
Max Info Frames	20	[1 - 100]
Support Extended Frames	false	
Status	{ok}	
Fault Cause		
Poll Service	BacnetMultiPoll	
Max Devices	max	
Enabled	true	
Port Id	2	
Port Info	MS/TP	

You must configure **`Link`** properties to specify the JACE serial port used, as well as other serial communications parameters. In addition, other MS/TP master parameters may require configuration.

Up to 4 Mstp Ports are supported, depending on host JACE platform, as well as licensing considerations. As with other Bacnet network ports, each Mstp Port has its own, independent Poll Scheduler (**`Poll Service`**). See “About poll components” in the *Drivers Guide* for more details. Also see “About Bacnet Polling” on page 3-17.

About Bacnet Comm: Client, Server, and Transport

Relative to other Bacnet Comm child containers, properties under the **`Client`**, **`Server`**, and **`Transport`** containers typically require little configuration (from default values).

Possible exceptions apply to the Bacnet Comm **`Server`** component. These exceptions are not typical, and can be skipped whenever the station’s BACnet driver operates only in “client mode” —meaning it is no t licensed and configured to operate as a BACnet server as well).

Bacnet Comm: Server configuration

If licensed and running as a BACnet server (typically in *addition* to a BACnet client), in a few cases it may be needed to make some configuration changes under the **Bacnet Comm**: **Server** component.

- Server Worker changes
- Remote reinitialization or backup/restore

Server Worker changes

The **Bacnet Comm**: **Server** component has a "Worker" child to queue and process server messages (responses to requests from BACnet clients). In AX-3.8, an additional "Confirmed Worker" child now separately handles "confirmed type" messages, for example event (alarm) notifications. A "Worker Pool" component is also available in the AX-3.8 `bacnet` palette, which you can paste as a child under a Worker.

If a station operates as a BACnet server with a large number of BACnet clients, configuration changes to one or both workers may be beneficial. For related details, see "Bacnet Workers" on page 3-19.

Remote reinitialization or backup/restore

In some cases, job circumstances require a remote BACnet client to have the ability to *reboot the Niagara host* (running the station). If so, you must change the "Reinitialize Allowed" property of the Bacnet Comm: Server component—which is *hidden*, by default. If you clear the "Hidden" config flag for this property from the Server's slot sheet, you can change its value from `false` to `true`.

In the **Server** layer component, this "Reinitialize Allowed" property must be set to true to support any of the following:

- A remote BACnet client to reinitialize the station (in most cases, this results in a host reboot).
- Server-side support of "backup and restore" functions from a remote BACnet Workstation client, conforming to BACnet DM-BR-B BIBB.
  Note: *This requires AX-3.2 or higher. In addition, you may wish to adjust related properties in the BacnetNetwork's LocalBacnetDevice. See "Local Device backup and restore properties" on page 4-80.*

About Bacnet Polling

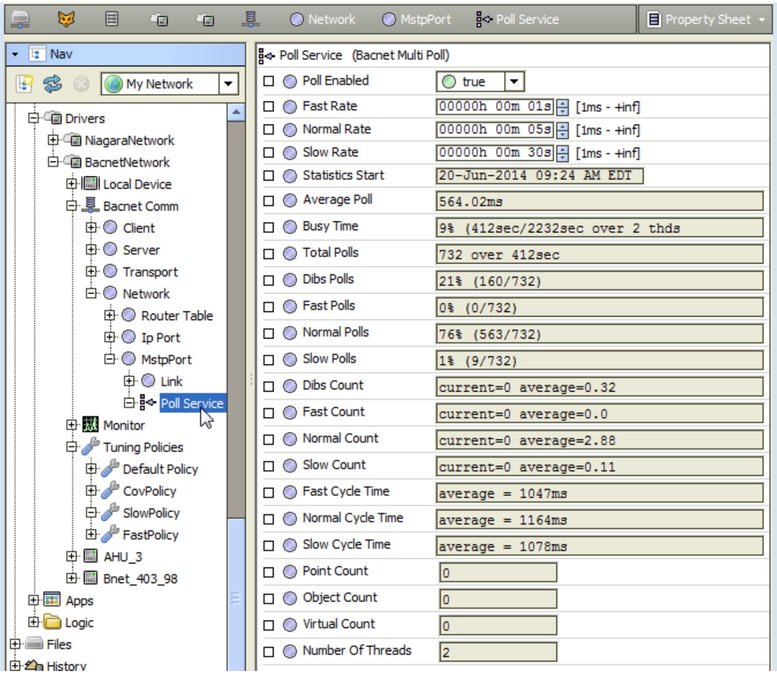
Since the initial (AX-3.0) release, polling improvements have been made in the NiagaraAX Bacnet driver, culminating in a reworked polling mechanism starting in AX-3.2. This section describes the different Bacnet driver polling mechanisms by different release levels, and notes related areas of possible configuration changes and default behaviors.

- Basic Bacnet polling design
- Current Bacnet polling (AX-3.2 and later)
- Notes on AX-3.1 and earlier Bacnet polling

Basic Bacnet polling design

Regardless of NiagaraAX release level, the Bacnet driver is unique among all drivers in that there are multiple communication port *types* available (IpPort, EthernetPort, MstpPort), along with multiple *instances* of ports supported, all under one network-level component (BacnetNetwork). Thus, the driver design has always used a separate "BacnetMultiPoll" (**Poll Service**) component for *each network port*, where each "poll scheduler" component is found under **BacnetComm**, **Network**, <*port*>, as shown in Figure 3-10.

Figure 3-10 Separate Poll Service for each port under BacnetComm, Network



Poll Service (Bacnet Multi Poll)

□ Poll Enabled	true
□ Fast Rate	00000h 00m 01s [1ms - +inf]
□ Normal Rate	00000h 00m 05s [1ms - +inf]
□ Slow Rate	00000h 00m 30s [1ms - +inf]
□ Statistics Start	20-Jun-2014 09:24 AM EDT
□ Average Poll	564.02ms
□ Busy Time	9% (412sec/2232sec over 2 thds
□ Total Polls	732 over 412sec
□ Dibs Polls	21% (160/732)
□ Fast Polls	0% (0/732)
□ Normal Polls	76% (563/732)
□ Slow Polls	1% (9/732)
□ Dibs Count	current=0 average=0.32
□ Fast Count	current=0 average=0.0
□ Normal Count	current=0 average=2.88
□ Slow Count	current=0 average=0.11
□ Fast Cycle Time	average = 1047ms
□ Normal Cycle Time	average = 1164ms
□ Slow Cycle Time	average = 1078ms
□ Point Count	0
□ Object Count	0
□ Virtual Count	0
□ Number Of Threads	2

You can adjust the basic poll settings used for each port in its own poll scheduler, as well as see individual statistics. The basic operation of each Poll Service, including the four "buckets" (3 rate group + "dibs" stack) is explained in the *Drivers Guide*, in the section "About poll components". Note that a "multi-thread" capability exists in all Bacnet Poll Services, as this is sometimes needed even in a (typically speedy) IpPort or EthernetPort link layer, where point polling might involve a BACnet router (say, going between B/IP and slower MS/TP devices).

Also unique to Bacnet polling (in any release) is the interplay with BACnet COV services, where some server devices may offer COV for point updates—typically, a desirable alternative to polling. This affects Tuning Policy configurations, and how you should assign tuning policies for points. For related details, see "Bacnet Tuning Policy notes" on page 3-24, "BacnetDevice properties" on page 3-31, and "Bacnet ProxyExt properties" on page 3-38.

And now for the release-level differences in Bacnet point polling:

- Current Bacnet polling (AX-3.2 and later)
- Notes on AX-3.1 and earlier Bacnet polling

Current Bacnet polling (AX-3.2 and later)

Starting in AX-3.2, polling of Bacnet proxy points uses the Basic Bacnet polling design, with the following key points:

- By *default*, any Bacnet proxy point *polls only the* (one) *single, configured property* for polling. Thus, if you specify "presentValue," that is all the point's out displays from BACnet—no native object status, or currently active level ("bac=*n*", if a commandable object), affect the point display.
  However, *as needed* you can add either or both of these extra polled properties, and/or others, by editing the *facets* of proxy points. See "Facets usage to poll additional properties" on page 3-40. Also, note that if adding statusFlags, there is a point "status merge"—see "Status merger for Bacnet proxy points" on page 3-39.
- Using an "added slot" method on a Bacnet ProxyExt, it is now possible to poll for data outside the normal boundaries of a single object poll, such as sometimes needed in "Event Enrollment" scenarios. For more details, see "Advanced "add DOPR slot" to ProxyExt method" on page 3-41.
- Efficiencies in polling come about from a "PollListEntries" method, where any "pollable" component (proxy points, Config objects) that is subscribed is added by its poll service into aggregate poll list objects (by containing device), where they are better processed. In addition, "dibs" polling can make more efficient use of bandwidth by using ReadPropertyMultiple to poll objects.
  For the most part, these efficiencies do not affect station polling configuration (however, starting in AX-3.2, the "maxDevicePollSize" property of a BacnetDevice, if still present, has no significance).

Notes on AX-3.1 and earlier Bacnet polling

Note that in any AX-3.0 or AX-3.1 system, Bacnet proxy point polling uses the Basic Bacnet polling design, but works differently than current Bacnet polling, as follows:

- All proxy points (regardless of property chosen for polling), poll the target object's Status_Flags property (if present). This reflects in the proxy point's status, as it is "merged" with the native Niagara status for the point itself. For details, see "Status merger for Bacnet proxy points" on page 3-39.
- Prioritized points (for objects with Priority_Array property) *always* poll that property in addition to the chosen property, such that this extra information is seen in the point display, using a "`bac=`*n*" display convention to show the currently active level.
- Points polled from the "dibs" stack in any Poll Service (highest priority) are always polled using the ReadProperty service instead of ReadPropertyMultiple. This individual poll technique does not always make best use of polling bandwidth.
- BacnetDevice components use a "maxDevicePollSize" property, a dynamic property that is sometimes added and (initially) auto-calculated when the device is learned or created, going by the property "maxAPDULengthAccepted" in its corresponding Device object. The purpose of this property was to allow tunable "group polling" (ReadPropertyMultiple service, if supported by device), specifying the number of points per poll—where you may adjust it downwards, if found necessary. This was sometimes needed if polling was for points with larger, unbounded data types, such as string or octet-string. Also, this property could automatically decrement on its own, in certain scenarios where polling issues occurred. In these cases, this resulted in more poll list entries (using single ReadProperty service).

Bacnet Workers

NiagaraAX "Worker" components queue and process threads in a running Niagara station. In the BACnet driver, Workers manage the queue and thread processing for reads and writes to BACnet.

Often, the default configuration of Workers provides good results. However, in some scenarios, configuration changes may improve performance. These topics provide more details:

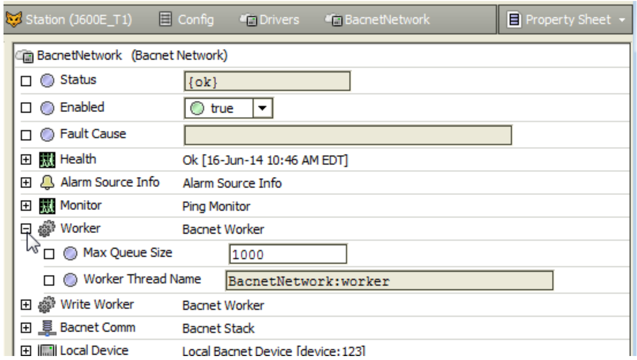
- "Location of Bacnet workers"
- "Bacnet worker properties"
- "Worker pool expansion"

Location of Bacnet workers

There are *at least* three frozen Worker slots under a BacnetNetwork, as follows:

- At least two Worker slots directly for a **`BacnetNetwork`**.

Figure 3-11 *Worker and Write Worker in BacnetNetwork property sheet (after unhiding)*



BacnetNetwork (Bacnet Network)

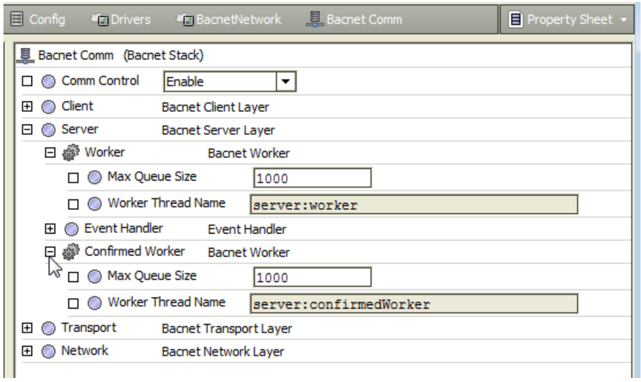
□ Status	{ok}
□ Enabled	true
□ Fault Cause	
⊞ Health	Ok [16-Jun-14 10:46 AM EDT]
⊞ Alarm Source Info	Alarm Source Info
⊞ Monitor	Ping Monitor
⊟ Worker	Bacnet Worker
□ Max Queue Size	1000
□ Worker Thread Name	BacnetNetwork:worker
⊞ Write Worker	Bacnet Worker
⊞ Bacnet Comm	Bacnet Stack
⊞ Local Device	Local Bacnet Device [device:123]

Note: *By default, these* **`BacnetNetwork`** *worker slots have the "hidden" flag set. To see/access them as shown above, you must first unhide (clear this flag) from the network's slot sheet.*

- **`Worker`** — "Worker Thread Name" property value of `BacnetNetwork:worker`
  This worker processes most BACnet client read messaging.
- **`Write Worker`** — "Worker Thread Name" property value of `BacnetNetwork:writeWorker`
  This worker processes most BACnet client write messaging.
- **`COV Worker`** (AX-3.8 only) "Worker Thread Name" property value of `BacnetNetwork:covWorker`. This worker processes most BACnet client COV messaging.

- At least one *visible* slot under the **Bacnet Comm**, **Server** component.

Figure 3-12 *Worker in BacnetNetwork property sheet (after unhiding)*



Bacnet Comm (Bacnet Stack)	
□ Comm Control	Enable
⊞ Client	Bacnet Client Layer
⊟ Server	Bacnet Server Layer
⊟ Worker	Bacnet Worker
□ Max Queue Size	1000
□ Worker Thread Name	server:worker
⊞ Event Handler	Event Handler
⊟ Confirmed Worker	Bacnet Worker
□ Max Queue Size	1000
□ Worker Thread Name	server:confirmedWorker
⊞ Transport	Bacnet Transport Layer
⊞ Network	Bacnet Network Layer

- **Worker** — has “Worker Thread Name” property value of `server:worker`
  This worker processes most BACnet server messaging.
- **Confirmed Worker** (AX-3.8 only) — has “Worker Thread Name” property value of `server:confirmedWorker`

In a AX-3.8 station with both Server workers, “Worker” processes *unconfirmed* server messaging, e.g. “Who-Is” and “Who-Has” responses or unconfirmed COV server messaging. Whereas, “Confirmed Worker” handles all *confirmed* server messaging, including event (alarm) notifications, confirmed COV, and so on. Separate workers help to prevent certain confirmed event notifications from blocking the processing of all incoming requests to the BACnet server layer.

Bacnet worker properties

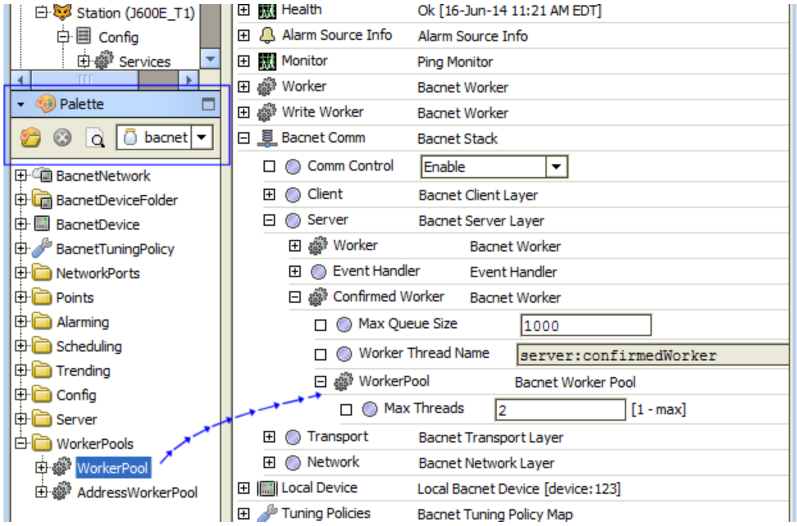
All Worker components have the same two properties:

- **Max Queue Size**
  Specifies the maximum number of items that can be queued for worker processing. The default is 1000. In a few cases, particularly in a large system, increasing this size may be beneficial.
  Note: *Queue size does not allocate a fixed memory size. Rather, the amount of memory used is dynamic, changing as message items are added and removed from the queue.*
- **Worker Thread Name**
  (Read only) Identifiers for threads managed by the worker.

Worker pool expansion

In AX-3.8, a “BacnetWorkerPool” (**Worker Pool**) component is available, found on the `bacnet` palette. In cases where Bacnet “queue full” messages have occurred, or to possibly increase messaging performance, it may help if you copy and paste a **Worker Pool** under one or more Worker slots.

Figure 3-13 *Worker Pool copied from palette and pasted under Bacnet Comm, Server, Confirmed Worker*



⊞ Health	Ok [16-Jun-14 11:21 AM EDT]
⊞ Alarm Source Info	Alarm Source Info
⊞ Monitor	Ping Monitor
⊞ Worker	Bacnet Worker
⊞ Write Worker	Bacnet Worker
⊟ Bacnet Comm	Bacnet Stack
□ Comm Control	Enable
⊞ Client	Bacnet Client Layer
⊟ Server	Bacnet Server Layer
⊞ Worker	Bacnet Worker
⊞ Event Handler	Event Handler
⊟ Confirmed Worker	Bacnet Worker
□ Max Queue Size	1000
□ Worker Thread Name	server:confirmedWorker
⊟ WorkerPool	Bacnet Worker Pool
□ Max Threads	2 [1 - max]
⊞ Transport	Bacnet Transport Layer
⊞ Network	Bacnet Network Layer
⊞ Local Device	Local Bacnet Device [device:123]
⊞ Tuning Policies	Bacnet Tuning Policy Map

- If a worker directly under the **`BacnetNetwork`**, *first* you must access the **`slot sheet`** of the network and *edit config flags* for its "worker" and/or "writeWorker" slots to "*unhide*" these slots. This lets you see/access the Worker components on the network's property sheet, where you can paste a Worker Pool component on (under) them.
  Note: *At the time of this document, the (AX-3.8) BacnetWorker "COV Worker" does not support a "Worker Pool" child. An error results if you attempt to add one under it.*
- Between the two **`Bacnet Comm`**, **`Server`** workers, typically the "**`Confirmed Worker`**" benefits *most* from adding a Worker Pool under it. This is the example shown in Figure 3-13.

Worker Pool properties

WorkerPool has a single configuration property:

- **`Max Threads`** — Specifies the maximum number of threads in the worker pool; the default is 2. Often this is sufficient, but may be increased if necessary. However, large numbers may prove to be detrimental, especially if a JACE platform with limited resources.

Bacnet Local Device

The Bacnet **`Local Device`** is a frozen slot under the BacnetNetwork that represents the station externally. In general, this means providing BACnet *server* responses to remote client requests (from other BACnet devices on a connected network).

Note: *With few exceptions, every component outside the Bacnet Local Device represents BACnet client operation. Client operation is done with BacnetDevice objects, child proxy points, and so on.*

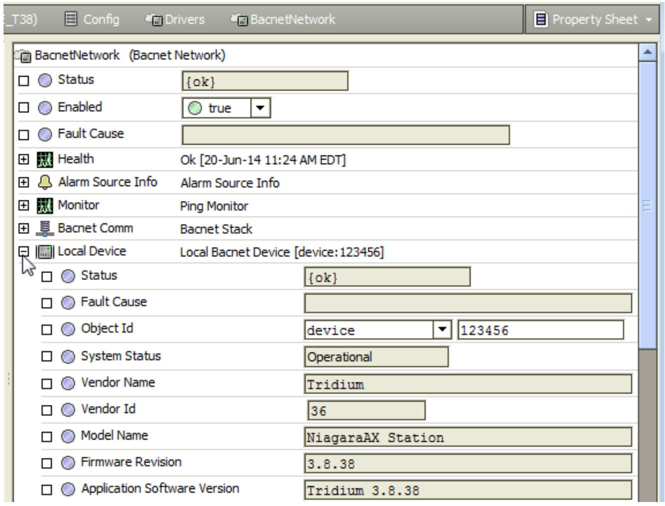
The Local Device has many properties, plus other container slots, briefly described in the following sections:

- An essential Local Device property
- About Local Device slots
  - "Time synchronization properties" on page 3-22 (AX-3.5 and later)

An essential Local Device property

Perhaps the most important Local Device property is the device object ID for the station—by default, this is -1 (driver inoperative). *To permit any BACnet operation in the station*, you must reassign **`Object Id`** to a valid, unique, device ID on the BACnet internetwork. Range is from 0 to 4194302.

Figure 3-14 *Bacnet Local Device property Object Id is critical for driver operation*



BacnetNetwork (Bacnet Network)

□ Status	{ok}
□ Enabled	○ true
□ Fault Cause	
⊞ Health	Ok [20-Jun-14 11:24 AM EDT]
⊞ Alarm Source Info	Alarm Source Info
⊞ Monitor	Ping Monitor
⊞ Bacnet Comm	Bacnet Stack
⊟ Local Device	Local Bacnet Device [device:123456]
□ Status	{ok}
□ Fault Cause	
□ Object Id	device 123456
□ System Status	Operational
□ Vendor Name	Tridium
□ Vendor Id	36
□ Model Name	NiagaraAX Station
□ Firmware Revision	3.8.38
□ Application Software Version	Tridium 3.8.38

Figure 3-14 shows the BacnetNetwork property sheet with an expanded Local Device container. In this example, the station is operating as BACnet Device 123456.

Most other properties are status (read-only) types reflective of the station's BACnet device capabilities. However, several are configuration properties to specify location, description, and various APDU parameters. External from Niagara (to other BACnet devices), these properties determine how the BACnet Device object appears that represents the station. See "Local Device properties" on page 4-78 for more details.

About Local Device slots

The **`Export Table`** under the BacnetNetwork's Local Device container is the central location to export station data as BACnet objects. The default view on the Export Table is the **`Bacnet Export Manager`**, in which you select and expose station components. For complete details, see "Niagara Bacnet Server Operation" on page 4-59, as well as "Local Device notes" on page 4-78.

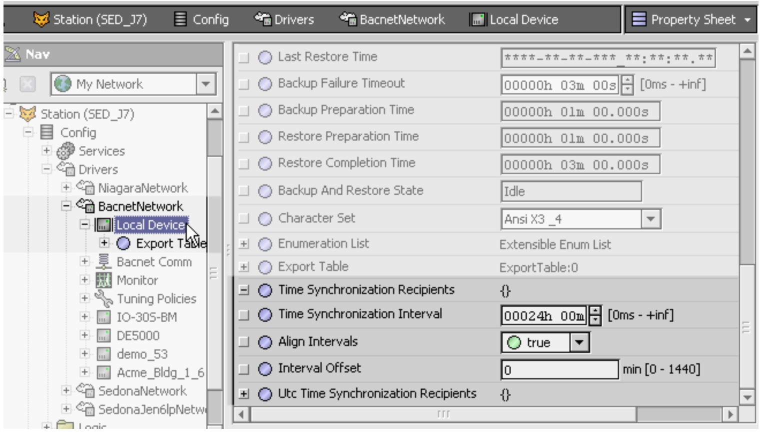
Starting in AX-3.5, a complete collection of "time synchronization" properties are among the slots of the Local Device of a BacnetNetwork. See the next section "Time synchronization properties".

Time synchronization properties

An AX-3.5 or later BacnetNetwork's LocalDevice component has a collection of "Time synchronization server" properties, formerly available only in the LocalDevice of a **`BACnet Supervisor`** station (BacnetWsNetwork). The LocalDevice in a **`BACnet AWS Supervisor`** (BacnetAwsNetwork) or **`BACnet OWS Supervisor`** (BacnetOwsNetwork) also uses this *same collection* of properties.

These properties are at the bottom of the network's LocalDevice property sheet (Figure 3-15).

Figure 3-15 *Time Synchronization properties in BacnetNetwork LocalDevice's property sheet*

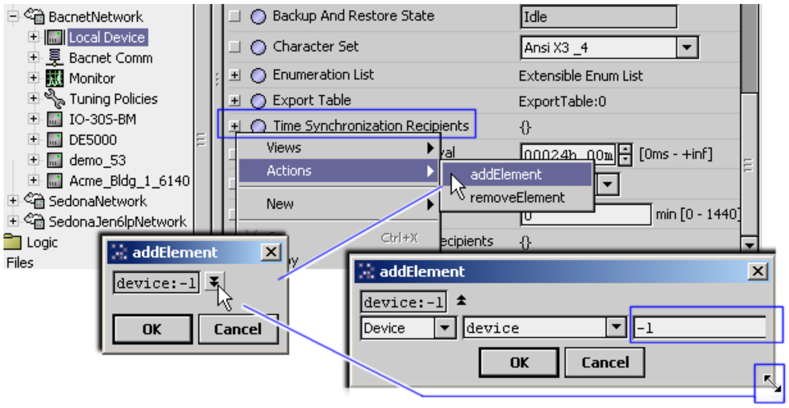


Last Restore Time	****-**-**-*** **:**:**.**
Backup Failure Timeout	00000h 03m 00s [0ms - +inf]
Backup Preparation Time	00000h 01m 00.000s
Restore Preparation Time	00000h 01m 00.000s
Restore Completion Time	00000h 03m 00.000s
Backup And Restore State	Idle
Character Set	Ansi X3 _4
Enumeration List	Extensible Enum List
Export Table	ExportTable:0
Time Synchronization Recipients	{}
Time Synchronization Interval	00024h 00m [0ms - +inf]
Align Intervals	true
Interval Offset	0 min [0 - 1440]
Utc Time Synchronization Recipients	{}

Configuration allows periodic BACnet time synchronization messages to be sent *from the station* to all defined time synchronization recipients (devices), or to UTC time synchronization recipients (devices), at the interval specified in the properties. The following properties are in this collection:

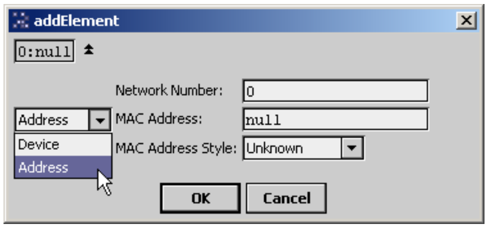
- **Time Synchronization Recipients**
  Contains the list of recipients for periodic Time Synchronization messages. You can add and remove entries to this list using the right-click **`addElement`** and **`removeElement`** actions. Entries in this list are BacnetRecipients. They can be either a Device Object Identifier, or a BACnet Address. See Figure 3-16 for an example of the popup dialog from invoking the addElement action.

Figure 3-16 *Adding time synchronization recipient with right-click "addElement" action*



As shown in Figure 3-16 above, the default **`addElement`** popup dialog is for Device object instance number (click on the small ▾ drop-down control, and then drag the popup corner out to access the instance number field.) Edit from the default -1 to the device's unique Device instance number. Or after resizing, click the ▾ drop-down control beside "Device" and change to "Address." This changes the **`addElement`** popup dialog to have different fields, as shown in Figure 3-17 below.

Figure 3-17 *Dialog to specify time synchronization recipient by Address instead of Device instance number*



In this case you must know the BACnet device's network number, MAC address, and MAC address style (choices are `Ethernet`, `IP`, `MSTP/Other`, and `Unknown`).

- **Time Synchronization Interval**
  Configures how frequently Niagara will send the time synchronization messages. The default value is 24 hours.
- **Align Interval**
  Specifies if the periodic time synchronization messages should be aligned to the beginning of the next interval. For example, if Time Synchronization Interval is exactly 24 hours and Align Interval is `true`, then the time synchronization messages will be sent out exactly at the beginning of the day, at 12:00 midnight.
- **Interval Offset**
  Specifies an offset from the beginning of the interval at which to send the periodic time synchronization messages. If Align Interval is `false`, this property is ignored.
- **Utc Time Synchronization Recipients**
  Contains the list of recipients for periodic UTC Time Synchronization messages. Works the same as the Time Synchronization Recipients property, where you can add and remove entries to this list using the `addElement` and `removeElement` actions (see Figure 3-16 and Figure 3-17). The entries in this list are BacnetRecipients. They can be either a Device Object Identifier, or a BACnet Address.

About Bacnet Tuning Policies

Bacnet Tuning Policies are set at the network level, similar to other NiagaraAX drivers. For general information, see "About Tuning Policies" in the *Drivers Guide*.

- Bacnet Tuning Policy notes
  - Bacnet-only Tuning Policy properties
  - Special tuning notes about "stale"
- Client COV notes
  - Changes in COV statusFlags reporting

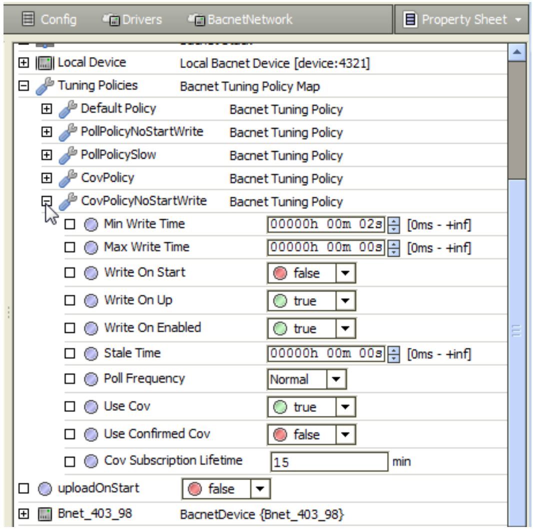
Bacnet Tuning Policy notes

Typically, you create *multiple* tuning policies in a BacnetNetwork—especially if you have multiple Bacnet network ports enabled. This allows you to reference *different* tuning policies from Bacnet proxy points under *different* networks (often with vastly different performance capabilities). For example, points under a Bacnet MSTP network would likely use different tuning policies than those under a Bacnet IP network.

Note: *You can add a tuning policy in the BacnetNetwork's property sheet by simply right-clicking an existing policy, then choosing* **`Duplicate`** *in the shortcut menu.* **`Name`** *it as desired, and edit its properties as needed. Or, drag a* **`BacnetTuningPolicy`** *from the palette, dropping it in the network's TuningPolicies container, and edit as needed.*

Also, if you have client BacnetDevices that provide support for COV notifications, you typically add at least one tuning policy that has its "Use Cov" property set to True (and possibly with other adjustments), see Figure 3-18.

Figure 3-18 *Example Tuning Policy used for points under a COV-capable device*



⊞ Local Device	Local Bacnet Device [device:4321]
⊟ Tuning Policies	Bacnet Tuning Policy Map
⊞ Default Policy	Bacnet Tuning Policy
⊞ PollPolicyNoStartWrite	Bacnet Tuning Policy
⊞ PollPolicySlow	Bacnet Tuning Policy
⊞ CovPolicy	Bacnet Tuning Policy
⊟ CovPolicyNoStartWrite	Bacnet Tuning Policy
□ Min Write Time	00000h 00m 02s [0ms - +inf]
□ Max Write Time	00000h 00m 00s [0ms - +inf]
□ Write On Start	false
□ Write On Up	true
□ Write On Enabled	true
□ Stale Time	00000h 00m 00s [0ms - +inf]
□ Poll Frequency	Normal
□ Use Cov	true
□ Use Confirmed Cov	false
□ Cov Subscription Lifetime	15 min
□ uploadOnStart	false
⊞ Bnet_403_98	BacnetDevice {Bnet_403_98}

Then, when adding/editing proxy points under such COV-capable devices, you can specify to use this (COV-specific) tuning policy.

Also see the two next sections:

- Bacnet-only Tuning Policy properties
- Special tuning notes about "stale"

Bacnet-only Tuning Policy properties

In addition to properties typical to most Tuning Policies, the Bacnet driver provides these properties:

- Use Cov
  Default is False. If set to True, and assigned proxy points are under a BacnetDevice that supports the Subscribe COV service, any necessary updates (Niagara proxy subscriptions) are attempted using BACnet COV subscriptions to the device. If the subscription attempt succeeds, the "Read Status" property of the point's BacnetProxyExt displays "`COV`". If the subscription attempt for a proxy point fails, normal polling is used and the "Read Status" property shows "`Polled`".
- Use Confirmed Cov
  If Use Cov is enabled (True) and assigned proxy points are under a BacnetDevice that supports Confirmed COV notifications, any necessary updates (Niagara proxy subscriptions) are attempted using BACnet Confirmed COV subscriptions to the device. If Use Cov is disabled (False), it makes no difference what this property value is. The default value is True.
- Cov Subscription Lifetime
  The lifetime, in minutes, for which Niagara will subscribe for COV notifications, then (if necessary) periodically re-subscribe. A value of zero means an indefinite lifetime, although this is not guaranteed to persist across resets of the server device. The default value is 15 minutes.

Note: *Also see "Client COV notes" on page 3-25, which explains recent changes in COV status reporting.*

Special tuning notes about “stale”

In any Tuning Policy, the default “Stale Time” is zero (0), effectively disabling the stale timer. This default is often used with good results. However, if setting “Stale Time” to a non-zero value, you should never set it shorter than the poll cycle time, or else points will go stale in the course of normal polling. Instead, set the stale timer to be longer than the largest expected poll cycle time. This period can vary depending on how many Px graphics are being viewed, and so on.

Also, note that for each point, the stale timer is measured from the *last time the point was updated*. This means if you have a 10-minute stale timer, and an 8-minute poll cycle time, you will have *some* points with values nearly 8 minutes old. If you stop polling, those points will begin going stale roughly 2 minutes after polling has stopped, and *not* 10 minutes.

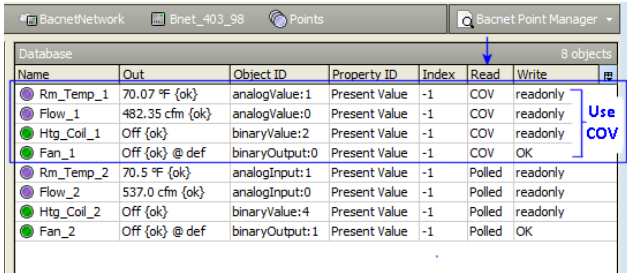
This has resulted in some confusion—where the expectation was that after viewing a graphic, any points in it should stay up for the 10 full minutes (or the configured Stale Time) before going stale. However, the actual time depends on how long ago they were last polled.

Client COV notes

Niagara offers client support for BACnet COV services—for devices that support COV. Receiving COV (change-of-value) notifications for proxy point data is typically more efficient than polling for data.

To take advantage of this, under the BacnetNetwork you need to add at least one additional TuningPolicy (apart from the DefaultPolicy), with its “Use Cov” property set to `true`. Then assign Bacnet proxy points under a COV-capable BacnetDevice to such a “CovPolicy”. The driver then attempts to *subscribe* to that BACnet device for COV notifications.

Figure 3-19 *Example proxy points subscribed for COV*



Name	Out	Object ID	Property ID	Index	Read	Write
Rm_Temp_1	70.07 °F {ok}	analogValue:1	Present Value	-1	COV	readonly
Flow_1	482.35 cfm {ok}	analogValue:0	Present Value	-1	COV	readonly
Htg_Coil_1	Off {ok}	binaryValue:2	Present Value	-1	COV	readonly
Fan_1	Off {ok} @ def	binaryOutput:0	Present Value	-1	COV	OK
Rm_Temp_2	70.5 °F {ok}	analogInput:1	Present Value	-1	Polled	readonly
Flow_2	537.0 cfm {ok}	analogInput:0	Present Value	-1	Polled	readonly
Htg_Coil_2	Off {ok}	binaryValue:4	Present Value	-1	Polled	readonly
Fan_2	Off {ok} @ def	binaryOutput:1	Present Value	-1	Polled	OK

If successful, such a proxy point will list in the **`Bacnet Point Manager`** “**`Read`**” column showing “COV” instead of “Polled”, as shown above in Figure 3-19. If the COV subscription is unsuccessful, the point will fall back to polling and show “Polled”, like points configured with a polling TuningPolicy. For related details, see “Bacnet-only Tuning Policy properties” on page 3-24. Also see the next section, “Changes in COV statusFlags reporting”.

Changes in COV statusFlags reporting

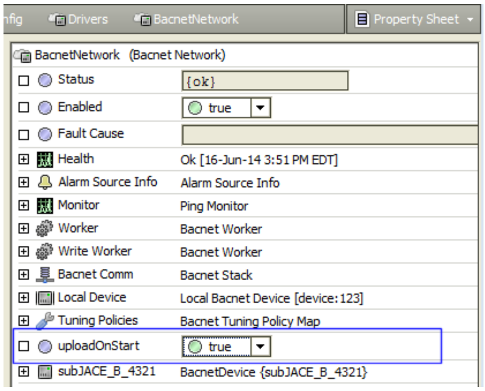
Until recent releases of the BACnet driver, the default behavior *varied* between *polled* Bacnet proxy points and proxy points *that use COV*, as follows:

- Polled points, by default, reflect only the selected BACnet property (typically “Present_Value”), but not any BACnet “Status_Flags” that be set on source objects. Only Niagara point status is seen. Thus, a source BACnet object may be in a native alarm state, but this is not indicated by the proxy point. Any alarm state requires a Niagara alarm extension on the point. If needed, you can change this behavior at the point level. See “Facets usage to poll additional properties” on page 3-40.
- COV points, by default, *did reflect* any BACnet “Status_Flags” on the source objects, in addition to the selected BACnet property (typically “Present_Value”). Therefore, a Bacnet proxy point might have an alarm status, yet there would be no Niagara alarm extension on the point. A “status merger” was used (sometimes causing confusion). See “Status merger for Bacnet proxy points” on page 3-39.

Starting in AX-3.7u1, the default COV proxy point statusFlags behavior *changed* to match polled proxy point behavior. In other words, only Niagara point status is seen by default. If needed, you can change this back to the previous COV proxy point behavior, at the point level, by adding a “`statusFlags`” facet to points. See “Facets usage to poll additional properties” on page 3-40.

BacnetNetwork "uploadOnStart" property

In AX-3.8 the BacnetNetwork has a new dynamic Boolean property “uploadOnStart”. Note this property appears only after a station with a BacnetNetwork has been started (e.g. not seen in the `bacnet` palette).



BacnetNetwork (Bacnet Network)

Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [16-Jun-14 3:51 PM EDT]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Worker	Bacnet Worker
Write Worker	Bacnet Worker
Bacnet Comm	Bacnet Stack
Local Device	Local Bacnet Device [device:123]
Tuning Policies	Bacnet Tuning Policy Map
uploadOnStart	true
subJACE_B_4321	BacnetDevice {subJACE_B_4321}

Figure 3-20 *BacnetNetwork “uploadOnStart” property (AX-3.8)*

This property determines how newly-added BacnetDevice components under the network are created.

- If “uploadOnStart” is `true`, added BacnetDevice components will have no “skipUpload” slot. Note the device-level “skipUpload” slot, new in AX-3.7u1, lets you specify to *not* automatically re-upload all BACnet Device object property values upon each station startup. In AX-3.7u1 (prior to AX-3.8), you have to manually add this Boolean slot to a BacnetDevice, if needed.
- If “uploadOnStart” is `false`, added BacnetDevices are automatically created with a “skipUpload” slot (*hidden*), with value of `true`. For devices that do not support message segmentation and/or with small APDU sizes (often MSTP devices), skipping the upload of these properties can greatly speed the startup of the station’s BacnetNetwork, and allow more important and dynamic point values to be read and written without undue delay.

Note: *Changing the value of the network’s* uploadOnStart *property does* not *affect any BacnetDevice components that already exist in the station. It only determines how newly-added BacnetDevices will be created.*

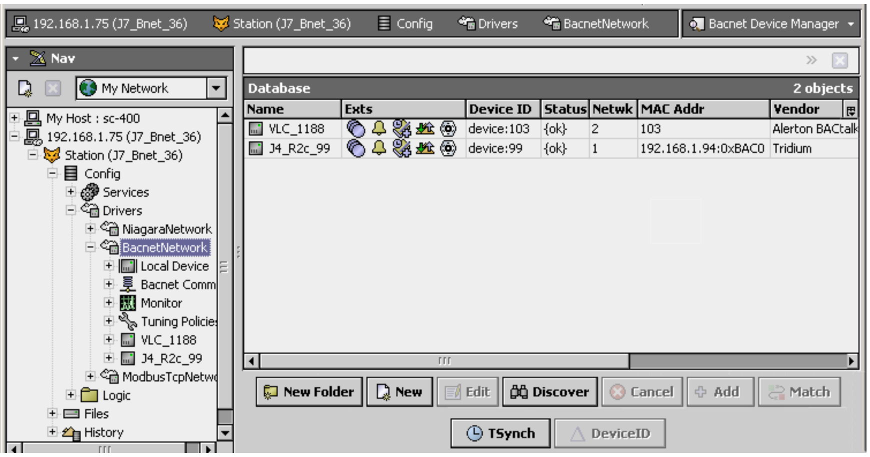
For related details, see “BacnetDevice "skipUpload" slot” on page 3-32.

Bacnet Device Manager

The Bacnet Device Manager (Figure 3-21) is the default view for the BacnetNetwork, and works similar to other device managers that support online device discovery. See “About the Device Manager” in the *Drivers Guide* for general information.

Note: *Starting in AX-3.5, the “*`Who Has`*” button at the bottom of the Bacnet Device Manager was removed, and two other buttons were added: “*`Tsynch`*” and “*`DeviceID`*”. Related details are in following sections.*

Note the “`Who Has`*” function is still available—from the toolbar or “*`Manager`*” menu on the menubar.*



Database 2 objects

Name	Exts	Device ID	Status	Netwk	MAC Addr	Vendor
VLC_1188		device:103	{ok}	2	103	Alerton BACtalk
J4_R2c_99		device:99	{ok}	1	192.168.1.94:0xBAC0	Tridium

Figure 3-21 *Bacnet Device Manager is default view for BacnetNetwork*

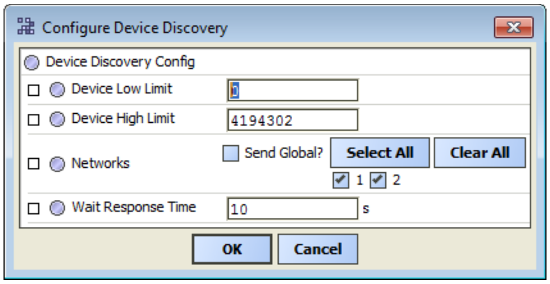
The following sections provide Bacnet-related details:

- About Bacnet Device Find Parameters
- Bacnet Device Manager discover notes
- Who Has function
- Timesynch (TSynch) function (AX-3.5 and later)
- Device ID function (AX-3.5 and later)

About Bacnet Device Find Parameters

When you click **Discover** in the Bacnet Device Manager, a **Configure Device Discovery** dialog appears, as shown in Figure 3-22.

Figure 3-22 *Bacnet Configure Device Discovery*



Configure Device Discovery

Device Discovery Config

□ Device Low Limit 0

□ Device High Limit 4194302

□ Networks □ Send Global? ☑ 1 ☑ 2

□ Wait Response Time 10 s

Note: *Starting in AX-3.7, buttons* **Select All** *and* **Clear All** *were added for the* **Networks** *field. These can be useful in some cases to limit a device discover to certain BACnet networks.*

Depending on the size of the BACnet internetwork you are learning, you may wish to configure settings from defaults. By default, all known networks are selected to learn (each network number matches an entry in the router table), “Send Global” (or “All Networks”) is *not* selected, and the full range of device IDs (0 to 4194302) is specified.

Fields in the **Configure Device Discovery** dialog work as follows:

- Device Low Limit
  Devices with an instance ID lower than this are not discovered. Default value is 0.
- Device High Limit
  Devices with an instance ID higher than this are not discovered. Default value is 4194302 (the maximum valid BACnet device instance number).
- Networks
  Allows selection of either:
  - Send Global? (All Networks) — Niagara will search on all possible BACnet networks segments, including all locally connected networks, as well as segments accessed over BACnet routers.
  - <*Known networks (by network number)*> — Choices reflect the number of network ports available under **Bacnet Comm**, **Network** component, plus any additional remote networks discovered by receiving “I-am-router” messages. Click to include/exclude as needed. By default, all known networks are individually pre-selected.
- Wait Response Time
  Determines how many seconds Niagara waits before determining that all BACnet devices that exist have responded to the discovery request. The default value is 10 seconds.

Often, you may wish to further limit the range of device IDs, or perhaps enable the device learn globally (“Send Global?” or “All Networks”). In the case of MSTP networks, or complex BACnet networks (that span several routers, you may need to set the “Wait Response Time” upwards (from 10 seconds) in order to learn all BACnet Device objects.

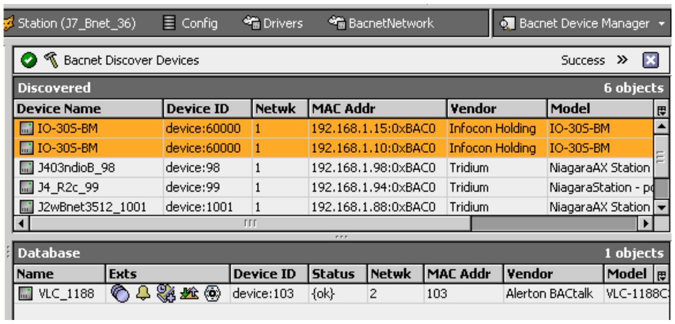
Bacnet Device Manager discover notes

You can rerun a device discover as many times as needed. If your BACnet network is very large, you may wish to add multiple BacnetDeviceFolders (using **New Folder** button), and run a series of *differently* configured device discovers (Figure 3-22) from each one, using the Bacnet Device Manager view available with each folder.

Devices discovered in fault (orange)

Sometimes discovered BACnet devices appear highlighted orange, indicating a fault.

Figure 3-23 Discovered devices in fault because of duplicate Device IDs



Discovered 6 objects

Device Name	Device ID	Netwk	MAC Addr	Vendor	Model
IO-30S-BM	device:60000	1	192.168.1.15:0xBAC0	Infocon Holding	IO-30S-BM
IO-30S-BM	device:60000	1	192.168.1.10:0xBAC0	Infocon Holding	IO-30S-BM
J403ndioB_98	device:98	1	192.168.1.98:0xBAC0	Tridium	NiagaraAX Station
J4_R2c_99	device:99	1	192.168.1.94:0xBAC0	Tridium	NiagaraStation - pd
J2wBnet3512_1001	device:1001	1	192.168.1.88:0xBAC0	Tridium	NiagaraAX Station

Database 1 objects

Name	Exts	Device ID	Status	Netwk	MAC Addr	Vendor	Model
VLC_1188		device:103	{ok}	2	103	Alerton BACtalk	VLC-1188C

As shown in Figure 3-23, often this from duplicate Device IDs (same Object_ID in the Device object of multiple devices). This is illegal in BACnet; each device must have a unique Object_ID for its Device object.

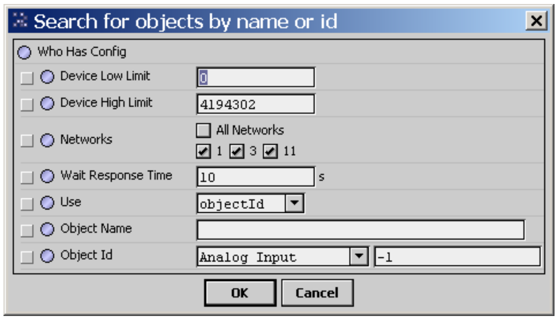
You can still add such devices to the station, however, typically all but one will have a "-1" (unassigned) Device ID, and be unreachable. Devices must be locally re-configured to have unique Device IDs. Then in the station, each BacnetDevice's corresponding (Config) Device Object must have its "Object Id" property set to match accordingly. Or, delete the devices, re-discover, and add again.

Note: *Starting in AX-3.5, a "Device ID" function was added in the* **`Bacnet Device Manager`** *that may provide a simple "one-step" solution, providing that affected BACnet devices allow Device ID change. For more details see "Device ID function" on page 3-29.*

Who Has function

The Bacnet Device Manager provides a **`Who Has`** item (in toolbar or "Manager" menu if AX-3.5 or later) or a **`Who Has`** button at bottom of manager view, if AX-3.4 or earlier. When selected, this provides a popup dialog to set parameters for the BACnet Who-Has service. You can use this to locate objects by name or object ID (type and instance number). The default dialog is shown in Figure 3-24.

Figure 3-24 Default Who Has popup dialog

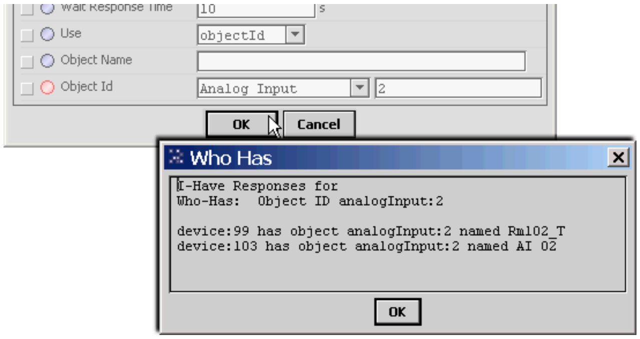


○ Who Has Config

□ ○ Device Low Limit	0
□ ○ Device High Limit	4194302
□ ○ Networks	□ All Networks ☑ 1 ☑ 3 ☑ 11
□ ○ Wait Response Time	10 s
□ ○ Use	objectId
□ ○ Object Name	
□ ○ Object Id	Analog Input -1

As in the device discover dialog, you can set device low and high limits, as well as select which BACnet networks (by number) receive the Who-Has broadcast (or all networks). When you click **`OK`**, a Bacnet Who Has job is started, and when the job completes the results (including "I-Have" responses) appear in another popup window, as shown in Figure 3-25.

Figure 3-25 Example result from Who Has



```
I-Have Responses for
Who-Has:  Object ID analogInput:2

device:99 has object analogInput:2 named Rm102_T
device:103 has object analogInput:2 named AI 02
```

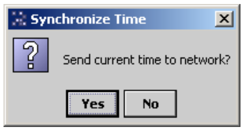
Timesynch (TSynch) function

Starting in AX-3.5, the Bacnet Device Manager provides a 🕒 **Tsynch** button that lets you send a manual (one-time) time synchronization message, from the station to all BACnet devices on the network.

Note: *Prior to AX-3.5, this function was available only from a BACnet Supervisor's* **Bacnet Ws Device Manager** *view. This functionality was moved from the* bacnetws *module into the core* bacnet *module.*

When you click 🕒 **Tsynch**, you are first asked to confirm this action, as it will send the current station time to all controllers on the network.

Figure 3-26 *Time Synch pre-confirmation dialog*

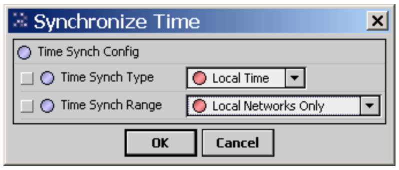


Synchronize Time

Send current time to network?

If you choose **Yes**, you see a **Synchronize Time** dialog, shown in Figure 3-27.

Figure 3-27 *Time Synchronization configuration dialog*



Synchronize Time

○ Time Synch Config

□ ○ Time Synch Type: Local Time

□ ○ Time Synch Range: Local Networks Only

As shown above, this dialog has two fields:

- **Time Synch Type**
  Select either Local Time (default), or UTC Time.
- **Time Synch Range**
  Select either Local Networks Only (default), or All Connected Networks.

When you click OK, a time synchronization request is then broadcast to the network. This is a one-time message.

If you wish to schedule *periodic* BACnet time synchronization sent from the station, configure this in the LocalBacnet Device (AX-3.5 and later). See "Time synchronization properties" on page 3-22 for details.

Note: *Such LocalDevice configuration also applies to a* **BACnet AWS Supervisor** *(BacnetAwsNetwork) or* **BACnet OWS Supervisor** *(BacnetOwsNetwork).*

Device ID function

Starting in AX-3.5, the Bacnet Device Manager provides a △ **DeviceID** button that *may* let you change the Device ID (Object_ID property of its Device object) in one or more selected remote BACnet devices. To work, devices must be in the station and reachable (OK status), *and accept a change to Device ID*. If a device does not accept a change to Device ID, the function fails with a popup describing why, typically "write access denied".

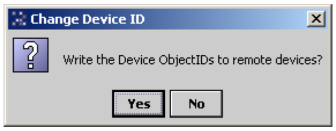
This function can be useful where "out of the box" BACnet controllers have been installed, all with some identical Device ID. In this case, such controllers appear in fault (orange) in the Discover pane (see Figure 3-23 on page 28).

When added, such devices will remain in fault (or down) unless Device ID is changed in the device, with a corresponding change in the (Config) Device Object's "Object Id" property of the BacnetDevice in the station. This function provides a way to *do all this all in one step*—again, providing that the BACnet device accepts changes to Device ID.

Note that merely changing the "Object Id" property in a BacnetDevice's Config, Device Object *does not write to the actual device*, it only determines how the station communicates with the device.

The △ **DeviceID** button becomes enabled with one or more devices selected in the Database pane. When you click it, a pre-confirmation dialog appears (Figure 3-28).

Figure 3-28 *Change Device ID pre-confirmation dialog*

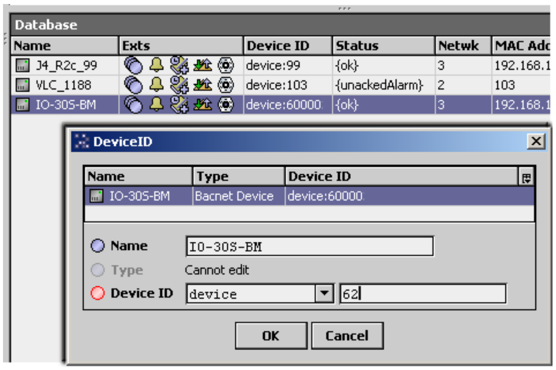


Change Device ID

Write the Device ObjectIDs to remote devices?

If you choose **Yes**, a popup **DeviceID** dialog appears, shown in Figure 3-29.

Figure 3-29 *DeviceID dialog (to change Device ID in a device)*



Database

Name	Exts	Device ID	Status	Netwk	MAC Adc
J4_R2c_99		device:99	{ok}	3	192.168.1
VLC_1188		device:103	{unackedAlarm}	2	103
IO-30S-BM		device:60000	{ok}	3	192.168.1

DeviceID

Name	Type	Device ID
IO-30S-BM	Bacnet Device	device:60000

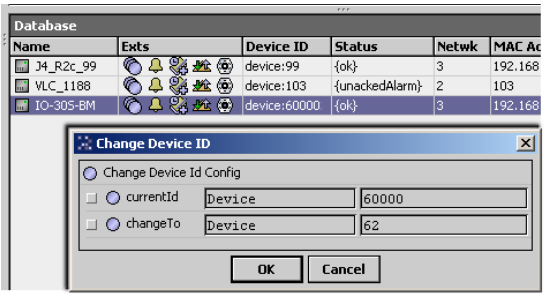
Name: IO-30S-BM
Type: Cannot edit
Device ID: device 62

As shown above, this dialog has two fields:

- **Name**
  The Device object's Object_Name in the BACnet device. Editable if only one device selected.
- **Device ID**
  The Device object's Object_ID in the BACnet device.
  In this second field you can enter a new numerical Object ID value, which must be unique across the job's BACnet internetwork. The valid range is from 0 to 4194302.
  Note: *If you've selected multiple BacnetDevices, this function will assign new Device IDs sequentially, starting with this number. For example if you have three devices selected and "62" entered, it will attempt to change Device IDs to 62, 63, and 64 amongst the three devices.*

Clicking **OK** shows a summary of the Device ID changes to be made, as shown in Figure 3-30.

Figure 3-30 *DeviceID summary*



Database

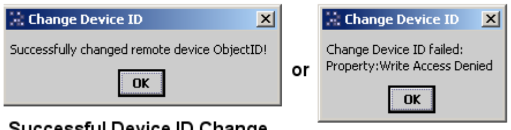
Name	Exts	Device ID	Status	Netwk	MAC Ac
J4_R2c_99		device:99	{ok}	3	192.168
VLC_1188		device:103	{unackedAlarm}	2	103
IO-30S-BM		device:60000	{ok}	3	192.168

Change Device ID

Change Device Id Config
currentId: Device 60000
changeTo: Device 62

Clicking **OK** launches a "Change Device Id Job", with an entry in the station's JobService. The job ends with a popup notification that explains whether the Device ID change was successful or not (Figure 3-31).

Figure 3-31 *Example Results Notifications for Change Device ID*



Change Device ID

Successfully changed remote device ObjectID!

Change Device ID

Change Device ID failed:
Property:Write Access Denied

Successful Device ID Change

Unsuccessful Attempt

Bacnet Device components

BacnetDevice components and properties are similar to most driver's device components—see "Common device components" in the *Drivers Guide* for general information.

In addition to common components, views, and device extensions, the BacnetDevice contains a "Config" device extension container. It provides a "Device Object" with a complete list of standard BACnet Device properties for the remote device.

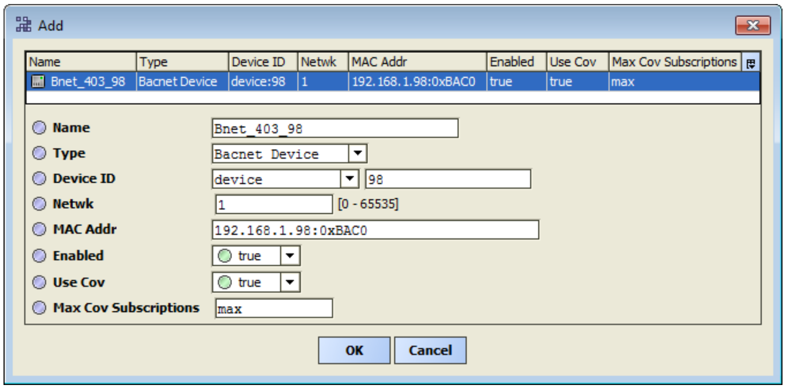
The following main sections provide more details:

- BacnetDevice properties
- About the Config Device Ext container

BacnetDevice properties

When you add (or edit) a BacnetDevice from the **`Bacnet Device Manager`**, several key device properties appear in that dialog, as shown in Figure 3-32.

Figure 3-32 *Add dialog in Bacnet Device Manager*



Name	Type	Device ID	Netwk	MAC Addr	Enabled	Use Cov	Max Cov Subscriptions
Bnet_403_98	Bacnet Device	device:98	1	192.168.1.98:0xBAC0	true	true	max

○ **Name** Bnet_403_98
○ **Type** Bacnet Device
○ **Device ID** device 98
○ **Netwk** 1 [0 - 65535]
○ **MAC Addr** 192.168.1.98:0xBAC0
○ **Enabled** ○ true
○ **Use Cov** ○ true
○ **Max Cov Subscriptions** max

These properties are described as follows:

- Name
  As learned, this reflects the "Object_Name" property from the device's sole Device object.
  Note: *Each slash ("/") character (if any) is replaced by a period (". ") in the Niagara name.*

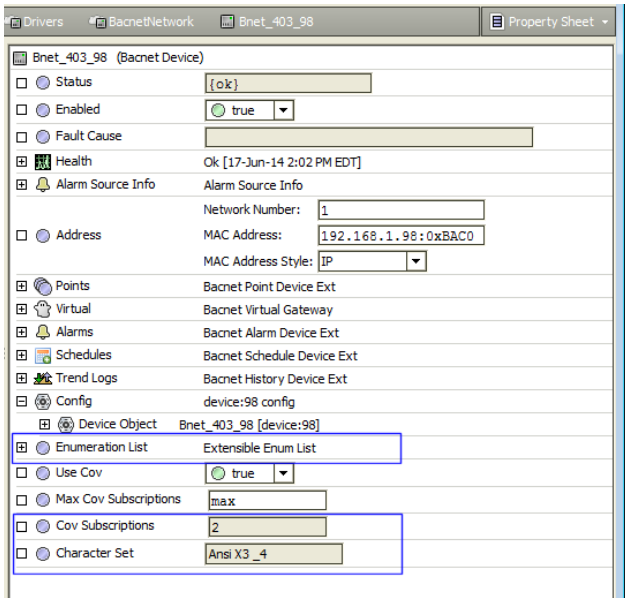
  This is only the Niagara component name for this device—if desired, you can edit it to whatever you want. It does not affect the name in the remote BACnet device.
- Type
  Type is specific type of BacnetDevice. Typically, is single selection only of `Bacnet Device`.
- Device ID
  The BACnet Object_Identifier reported (if learned) by this device. Editing does *not affect* the device's ID, only the ID used by Niagara to access this device.
- MAC Addr
  The data link layer MAC address for this device. If a B/IP device, this is *`IPaddress`*`:`*`UDPport`*, similar as shown in Figure 3-32. If an MS/TP device, this is an 8-bit address (0-254) that is unique on its RS485 trunk.
- Enabled
  A flag indicating whether Niagara has enabled communications to the device.
- Use Cov
  A flag specifying whether Niagara attempts to subscribe for COV notifications, as a way to monitor proxy point values.
  - If the device was learned, and Niagara determined that the device indicates support for server-side COV, this flag defaults to *true*.
  - Otherwise, this flag is *false*. In this case, no proxy points under the device will use COV.

  Note: *If true, then individual proxy points under the device may use COV subscriptions, depending on their assigned tuning policy. See "Bacnet Tuning Policy notes" on page 3-24 and "Client COV notes" on page 3-25.*
- Max Cov Subscriptions
  (Applies only if "Use Cov" is true.) Specifies the maximum number of COV subscriptions that Niagara will attempt to use with this device. This allows the station to restrict itself from consuming all of the available subscription space in that device.

Notes on other BacnetDevice properties

In addition to properties shown in the **Add** or **Edit** dialog from the **Bacnet Device Manager**, a BacnetDevice has a few other properties, as shown in Figure 3-33.

Figure 3-33 *Additional BacnetDevice properties*



Bnet_403_98 (Bacnet Device)	
□ Status	{ok}
□ Enabled	true
□ Fault Cause	
⊞ Health	Ok [17-Jun-14 2:02 PM EDT]
⊞ Alarm Source Info	Alarm Source Info
□ Address	Network Number: 1 MAC Address: 192.168.1.98:0xBAC0 MAC Address Style: IP
⊞ Points	Bacnet Point Device Ext
⊞ Virtual	Bacnet Virtual Gateway
⊞ Alarms	Bacnet Alarm Device Ext
⊞ Schedules	Bacnet Schedule Device Ext
⊞ Trend Logs	Bacnet History Device Ext
⊟ Config	device:98 config
⊞ Device Object	Bnet_403_98 [device:98]
⊞ Enumeration List	Extensible Enum List
□ Use Cov	true
□ Max Cov Subscriptions	max
□ Cov Subscriptions	2
□ Character Set	Ansi X3 _4

Typically inconsequential, these other BacnetDevice properties are as follows:

- **Enumeration List**
  An expandable list of extensible BACnet enumerations that are defined and applicable for this specific device. In rare cases, adding or editing items in this list may provide utility for proprietary data items in the BACnet device—perhaps property IDs.
- **Cov Subscriptions**
  (Read Only) Number of active COV client subscriptions to this device.
- **Character Set**
  (Read Only) The character set encoding used by this BACnet device, e.g. ANSI X3.4 (`Ansi X3_4`).

Starting in AX-3.7u1, another possible slot has importance. See "BacnetDevice "skipUpload" slot".

BacnetDevice "skipUpload" slot

Upon station startup, the BACnet driver has always automatically uploaded all "Device object" properties *for each represented BACnet device*, after which the station's BacnetNetwork is fully initialized. Typically, such property uploads are inconsequential if there are mostly BACnet/IP or BACnet Ethernet devices.

However, if a BacnetNetwork that includes many MS/TP devices, *this can cause a long delay at station startup*. During all these Device object uploads, more critical point data reads and writes cannot occur.

Starting in AX-3.7u1, any BacnetDevice with a Boolean slot named "skipUpload" (with a value of `true`), *skips* this upload at station startup. Note at any time, you can still manually invoke the Upload action on any BacnetDevice to refresh these values—typically, most Device object properties have static values.

If necessary, you can manually add this slot and value by doing the following:

1. Go to the slot sheet of the **BacnetDevice** (typically, one for an MS/TP device).
2. Add a slot with **Name**: `skipUpload` and choose **Type**: `baja, Boolean`
3. Go to the property sheet of the **BacnetDevice** and set the value of this new property to: `true`
4. **Save** these changes.

Alternatively, you can use the **Batch Editor** of the station's **ProgramService** to do all the above on *multiple* selected BacnetDevices. For related information, see the *NiagaraAX Batch Editor* engineering notes document.

Note: *Starting in AX-3.8, the BacnetNetwork has new "*uploadOnStartup*" property, which you can set from* `false` *(default) to* `true` *in order to have child BacnetDevices (from that point on) to be automatically created with the "*skipUpload*" slot, with a value of* `true`*. In this case, note that the added* skipUpload *slot on BacnetDevices is* hidden*. For related details, see "BacnetNetwork "uploadOnStart" property" on page 3-26.*

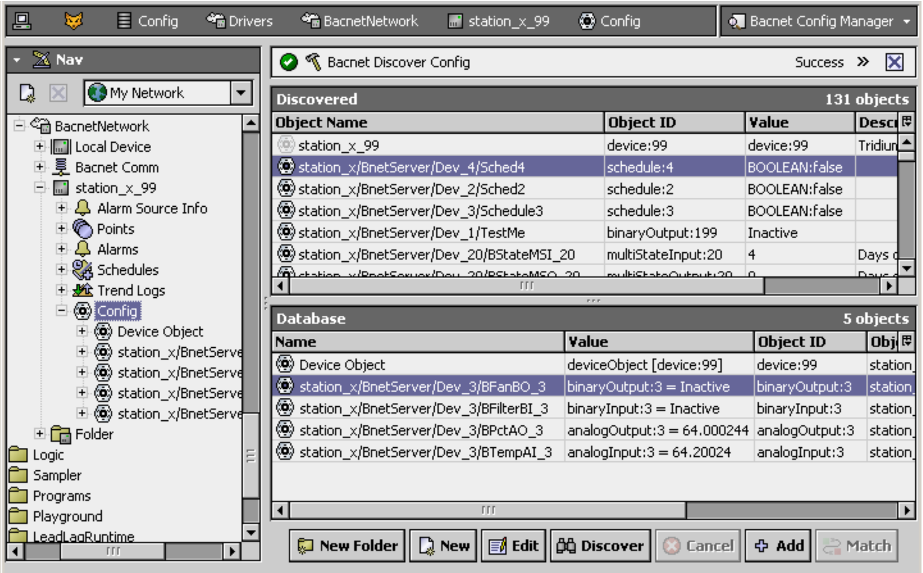
When engineering a BacnetNetwork with mostly MS/TP devices in AX-3.8, a possible tactic is to set the network-level "uploadOnStart" property to `false` *before* adding BacnetDevices. Then if desired, you could go to the slot sheet of BacnetDevices for B/IP and/or B/Ethernet devices and either delete (or unhide to change from the property sheet) the hidden "skipUpload" slot.

About the Config Device Ext container

Properties of the Device Object under the Bacnet **`Config`** Device Ext are populated when the device was learned. These represent BACnet Device object properties. Included are several properties that show the various protocol services and objects supported by that device.

The default view of the Bacnet Config Device Ext is the **`Bacnet Config Manager`** (Figure 3-34).

Figure 3-34 *Bacnet Config Manager*



Bacnet Discover Config — Success

Discovered — 131 objects

Object Name	Object ID	Value	Desc
station_x_99	device:99	device:99	Tridiun
station_x/BnetServer/Dev_4/Sched4	schedule:4	BOOLEAN:false	
station_x/BnetServer/Dev_2/Sched2	schedule:2	BOOLEAN:false	
station_x/BnetServer/Dev_3/Schedule3	schedule:3	BOOLEAN:false	
station_x/BnetServer/Dev_1/TestMe	binaryOutput:199	Inactive	
station_x/BnetServer/Dev_20/BStateMSI_20	multiStateInput:20	4	Days c

Database — 5 objects

Name	Value	Object ID	Obj
Device Object	deviceObject [device:99]	device:99	station_
station_x/BnetServer/Dev_3/BFanBO_3	binaryOutput:3 = Inactive	binaryOutput:3	station_
station_x/BnetServer/Dev_3/BFilterBI_3	binaryInput:3 = Inactive	binaryInput:3	station_
station_x/BnetServer/Dev_3/BPctAO_3	analogOutput:3 = 64.000244	analogOutput:3	station_
station_x/BnetServer/Dev_3/BTempAI_3	analogInput:3 = 64.20024	analogInput:3	station_

From this view, you can learn *other* config-type objects, each representing a BACnet object in that device. You might do this to "evaluate" a BACnet device, rather than for permanent station engineering. In this case, remember to *delete* the config-type objects afterwards, as they consume station resources. See "About Bacnet Config objects" on page 3-34.

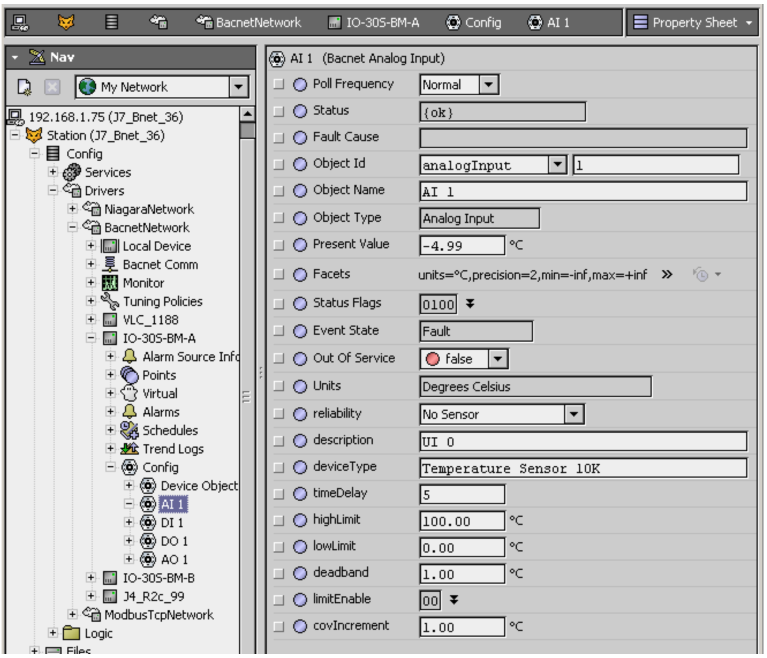
Note: *An alternate method to evaluate the objects in a device is via its "Virtual Gateway." This may actually be easier, as the resulting "virtual components" are automatically removed after browsing. For more details, see "About Bacnet virtual points" on page 3-52.*

If a **`BACnet AWS Supervisor`** based on the AX-3.6 or later `bacnetAws` module, the equivalent **`Bacnet Aws Config Manager`** view on the **`Config`** device extension of each BacnetAwsDevice provides two *additional* buttons at the bottom (apart from those shown in Figure 3-34). These buttons allow you to *create* and/or *delete* BACnet objects in the device, providing that the device supports these services. For more details, see "Bacnet Aws Config Manager" on page A-11.

About Bacnet Config objects

A config object provides a view of a BACnet object in its native format—where all properties of that object are presented as a whole. Figure 3-35 shows the property sheet of an Analog Input config object.

Figure 3-35 *Example config object for a Analog Input object*



AI 1 (Bacnet Analog Input)

Poll Frequency	Normal
Status	{ok}
Fault Cause	
Object Id	analogInput 1
Object Name	AI 1
Object Type	Analog Input
Present Value	-4.99 °C
Facets	units=°C,precision=2,min=-inf,max=+inf
Status Flags	0100
Event State	Fault
Out Of Service	false
Units	Degrees Celsius
reliability	No Sensor
description	UI 0
deviceType	Temperature Sensor 10K
timeDelay	5
highLimit	100.00 °C
lowLimit	0.00 °C
deadband	1.00 °C
limitEnable	00
covIncrement	1.00 °C

Unlike Bacnet proxy points that you create using the Bacnet Point Manager, when subscribed, config objects poll *all* properties—not as efficient as the selective proxy point model. In addition, slots of Bacnet config objects use primitive or special data types, not compatible with normal linking logic.

Config objects are expected to be useful for one-time commissioning, or for proprietary objects for which the proxy point interaction may be insufficient.

Bacnet Point Manager

The Bacnet Point Manager is the default view for the **`Points`** extension (or Points Folder) under any BacnetDevice, and works similar to other point managers that support online point discovery. See “About the Point Manager” in the *Drivers Guide* for general details.

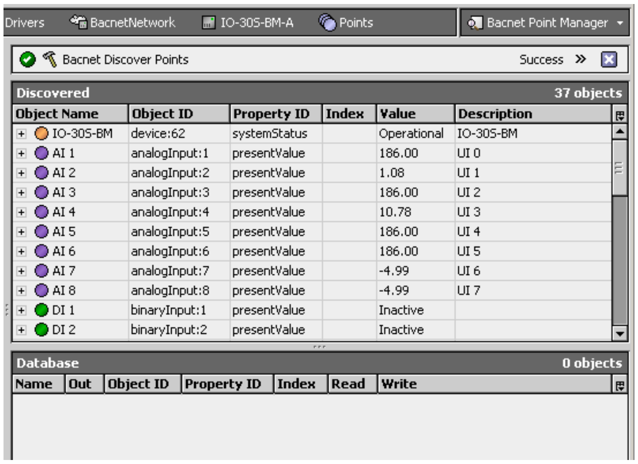
The following sections provide Bacnet-related details:

- Bacnet Point Manager “Discovered” notes
- Bacnet Point Manager “Database” notes

Bacnet Point Manager “Discovered” notes

By default, the “Discovered” table for Bacnet objects lists the single BACnet Device object at top, and other objects underneath, with each object initially occupying a single row. See Figure 3-36.

Figure 3-36 Discovered Bacnet objects show present value



Discovered — 37 objects

Object Name	Object ID	Property ID	Index	Value	Description
IO-30S-BM	device:62	systemStatus		Operational	IO-30S-BM
AI 1	analogInput:1	presentValue		186.00	UI 0
AI 2	analogInput:2	presentValue		1.08	UI 1
AI 3	analogInput:3	presentValue		186.00	UI 2
AI 4	analogInput:4	presentValue		10.78	UI 3
AI 5	analogInput:5	presentValue		186.00	UI 4
AI 6	analogInput:6	presentValue		186.00	UI 5
AI 7	analogInput:7	presentValue		-4.99	UI 6
AI 8	analogInput:8	presentValue		-4.99	UI 7
DI 1	binaryInput:1	presentValue		Inactive	
DI 2	binaryInput:2	presentValue		Inactive	

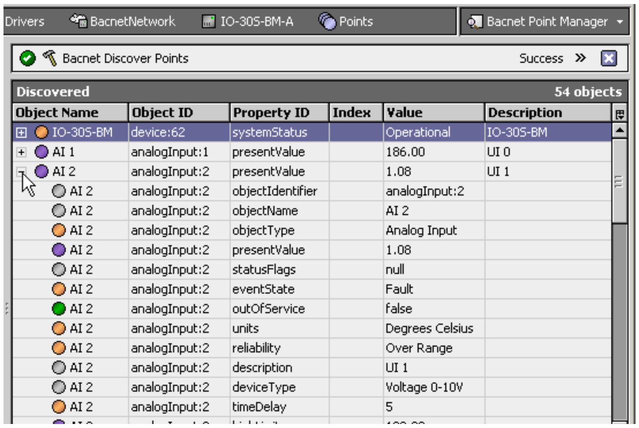
Database — 0 objects

Name	Out	Object ID	Property ID	Index	Read	Write

You can resort objects by clicking on any column header. Often, this is useful to sort by “Object ID” (BACnet object type), to group like type objects together.

Also, you can click to expand any object (Figure 3-37). This lets you select other properties (apart from Present_Value) as a candidate to proxy.

Figure 3-37 Click plus (+) icon beside any discovered object to see all properties



Discovered — 54 objects

Object Name	Object ID	Property ID	Index	Value	Description
IO-30S-BM	device:62	systemStatus		Operational	IO-30S-BM
AI 1	analogInput:1	presentValue		186.00	UI 0
AI 2	analogInput:2	presentValue		1.08	UI 1
AI 2	analogInput:2	objectIdentifier		analogInput:2	
AI 2	analogInput:2	objectName		AI 2	
AI 2	analogInput:2	objectType		Analog Input	
AI 2	analogInput:2	presentValue		1.08	
AI 2	analogInput:2	statusFlags		null	
AI 2	analogInput:2	eventState		Fault	
AI 2	analogInput:2	outOfService		false	
AI 2	analogInput:2	units		Degrees Celsius	
AI 2	analogInput:2	reliability		Over Range	
AI 2	analogInput:2	description		UI 1	
AI 2	analogInput:2	deviceType		Voltage 0-10V	
AI 2	analogInput:2	timeDelay		5	

The following sections provide more details about the Discovered pane in the Bacnet Point Manager:

- Discovered object table columns
- Discovered object usage notes

Discovered object table columns

By default, the following columns appear in the `Discovered` table, from left-to-right:

Object NameObject IDProperty IDIndexValueDescription

Object Name The Object_Name property of the discovered BACnet object. This name should be unique within this specific device. When you select an object to add as a proxy point, this is the default (Niagara) name in the station for the Bacnet proxy point.

If BACnet objects in the device are organized in some hierarchical folder scheme, the Object Name reflects this, using a forward slash (“/”) between folders. For example:

```
J4_R2c/BNetServer/Dev_20/BFan_BO_20
```

Object ID The Object_Identifier property of the discovered BACnet object, which is a *combination* of BACnet *object type* and *instance number* (unique within that type). In this column, these two fields appear separated by a “:”, using Niagara descriptors for type. For example:

- analogInput:3
- multiStateValue:3
- binaryOutput:3

Property ID The property shown for the discovered BACnet object. By default, all objects that have a Present_Value property are listed with this value “on top,” as shown in Figure 3-36. BACnet objects without a Present_Value property (for example, a Device object or Trend Log object), list showing another property as the top Property ID.

Typically, present value is the most useful piece of data from any BACnet object. However, you can expand any discovered object (Figure 3-37) to see *all its other properties* as children in the discovered pane, where each one is a separate proxy point candidate. Additionally, properties that are “arrays” are further expandable (see Index column).

Index This is a numeric index into an “arrayed” property, if selected, otherwise is *blank*. For example, if you expand a “priority-type” object (e.g: object type “binaryOutput”) and expand again on its “prioritArray” property, each child row displays with a unique index number (1—16).

Note: *Use this to proxy a point to write to only one specific level of the “Priority_Array” property of the target BACnet object (instead of accessing all levels, by proxy of only the default “presentValue” property). This may be useful in your control scheme, if you have a number of possible sources in the station you wish to evaluate (on a Niagara priority basis), to write to one BACnet priority level (only) in the target BACnet object.*

Value The (static) value of the associated property, captured when Niagara retrieved the object list (or, for any “non-default” properties, when you expanded the “top” property—see Property ID).

Discovered values display showing any descriptors associated with the BACnet object’s related properties (Units, Active_Text, Inactive_Text, and so on).

Description Any available character string Description for the BACnet object. In many cases, this value may be blank.

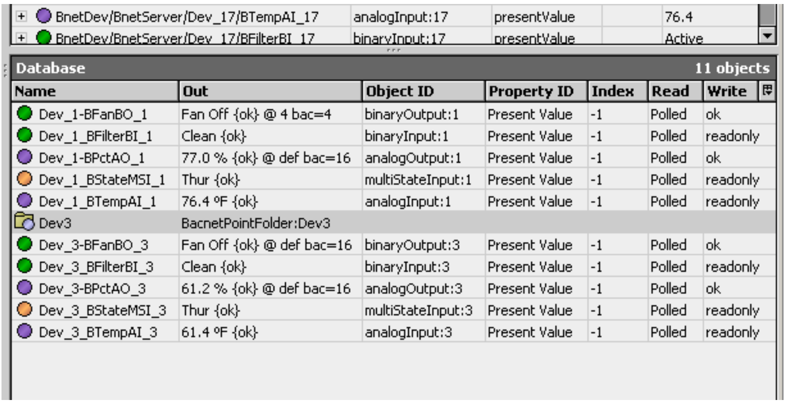
Discovered object usage notes

You can rerun a point discover as many times as needed. If the BACnet device contains many objects, you may wish to add multiple BacnetPointFolders (using **`New Folder`** button), and add discovered points differently into each one, using the Bacnet Point Manager view available with each folder.

Bacnet Point Manager “Database” notes

The “Database” table contains existing Bacnet proxy points, where each appears as a row in the table. Each proxy point represents one data item from a specific BACnet object in that device (Figure 3-38).

Figure 3-38 *Database shows Bacnet proxy points in the station*



BnetDev/BnetServer/Dev_17/BTempAI_17	analogInput:17	presentValue	76.4
BnetDev/BnetServer/Dev_17/BFilterBI_17	binaryInput:17	presentValue	Active

Database 11 objects

Name	Out	Object ID	Property ID	Index	Read	Write
Dev_1-BFanBO_1	Fan Off {ok} @ 4 bac=4	binaryOutput:1	Present Value	-1	Polled	ok
Dev_1_BFilterBI_1	Clean {ok}	binaryInput:1	Present Value	-1	Polled	readonly
Dev_1-BPctAO_1	77.0 % {ok} @ def bac=16	analogOutput:1	Present Value	-1	Polled	ok
Dev_1_BStateMSI_1	Thur {ok}	multiStateInput:1	Present Value	-1	Polled	readonly
Dev_1_BTempAI_1	76.4 ºF {ok}	analogInput:1	Present Value	-1	Polled	readonly
Dev3	BacnetPointFolder:Dev3					
Dev_3-BFanBO_3	Fan Off {ok} @ def bac=16	binaryOutput:3	Present Value	-1	Polled	ok
Dev_3_BFilterBI_3	Clean {ok}	binaryInput:3	Present Value	-1	Polled	readonly
Dev_3-BPctAO_3	61.2 % {ok} @ def bac=16	analogOutput:3	Present Value	-1	Polled	ok
Dev_3_BStateMSI_3	Thur {ok}	multiStateInput:3	Present Value	-1	Polled	readonly
Dev_3_BTempAI_3	61.4 ºF {ok}	analogInput:3	Present Value	-1	Polled	readonly

You can resort points by clicking on any column header. Often, this is useful to sort by “Object ID” (BACnet object type), or perhaps by name.

Also, if you created BacnetPointFolders under a device’s Points container, you can see all proxy points in the device from the main (root) Points Manager using “All Descendants” tool. For details, see the section “Points New Folder and New” in the *Drivers Guide*.

The following sections provide more details about the Database pane in the Bacnet Point Manager:

- Database proxy point table columns
- Modifying the Database table

Database proxy point table columns

By default, the following columns appear in the **`Database`** table, from left-to-right:

NameOutObject IDProperty IDIndexReadWrite

Note: *You can also modify columns shown, see “Modifying the Database table” on page 3-37.*

Name The Niagara name for the Bacnet proxy point. If you added the point from a discover (selecting the default “presentValue” property), and did not edit Object Name, this will be identical to the BACnet object’s name. As needed, you can edit when adding, or at any later time.

Out The current out value, including any point facets. For (presentValue) proxy points *before* AX-3.2, this includes both the BACnet “Present_Value” and the logical “OR” status *merger* of the BACnet object’s “Status_Flags” (in_alarm, fault, overridden, out_of_service) along with Niagara point status.

Note: *BACnet status flags “in_alarm” and “out_of_service” OR with Niagara status “alarm” and “disabled,” respectively. Status flags “overridden” and “fault” map OR to identically named Niagara statuses. For more details, see “Status merger for Bacnet proxy points” on page 3-39.*

Also, (before AX-3.2) if a *writable* proxy point for a “priority type” BACnet object (Binary_Output, Analog_Output, etc.), the out value facets always include the active BACnet priority array level (1—16), as the object’s “Priority_Array” is also automatically polled. By default, this is formatted as “`bac=n`”. For example, a (presentValue) BooleanWritable point for a Binary_Output object may have an out of:

```
On {ok} @ 16 bac=16
```

Where here, the active Niagara *point* priority level (`@ 16`) agrees with the current BACnet priority level.

Note: *Starting in AX-3.2, out display of a proxy point defaults to only the single (configured) property value, along with Niagara status for the proxy point. However, you can edit point facets to poll for additional properties, such as the native “statusFlags” and/or “priorityArray” level. For details, see “Facets usage to poll additional properties” on page 3-40.*

Object ID Just as in the Discovered table, this is the “Object_Identifier” property of the proxied BACnet object, which is a *combination* of BACnet *object type* and *instance number* (unique within that type). In this column, these two field appear separated by a “:”, using Niagara descriptors for type. For example:

- analogInput:3
- multiStateValue:3
- binaryOutput:3

Property ID The BACnet property name proxied by the point. For example “Present Value” or “Event State,” depending on selection.

Note: Do not change *Property ID on an existing Bacnet proxy point. This often produces undesirable results, especially if a different control point type would apply. Instead, make a new proxy point.*

Index Applies only if an “arrayed” property, like “Priority Array” (otherwise it is “`-1`” for no index). Provides a numeric index into the property array.

Read The read-only “Read Status” of the ProxyExt, which is typically either “Polled,” “COV,” or “unsubscribed,” depending on a number of factors. If a read error occurs, other descriptive text may appear instead.

Write The read-only “Write” status of the ProxyExt, which is typically “`readonly`” if the proxy point is not writable, or if a writable point type, either “`writable`” or “`ok`” (last Niagara write occurred within effective Tuning Policy period). Or, if a write operation fails, the write status provides some descriptive text.

For example, if you create a writable point for a readonly object (say, a NumericWritable for presentValue of an Analog_Input object), and attempt to write from Niagara, following this the write status may show: “`Property:Write Access Denied`”. If the error is actually a BACnet Error, then this colon-separated format will contain the Error Class and Error Code returned by the device.

Modifying the Database table

You can modify which data columns appear in the Bacnet Point Manager database table. For the options menu, simply click the small “table options” control in the upper right corner. See the section “Manager table features” in the *Drivers Guide* for general details.

Non-default selections for data columns in the Bacnet Point Manager Database table include various properties of both the parent proxy point and the Bacnet ProxyExt, and are the following:

- Path — Station path to the proxy point.
- Type — Type of Niagara point the proxy is based upon (BooleanWritable, NumericPoint, etc.).
- Enabled — Whether the Niagara proxy point is currently enabled for communications.
- Tuning Policy Name — Name of the network’s TuningPolicy component assigned to the point.
- Data Type — The ASN data type for the property (ENUMERATED, REAL, and so on).

- Device Facets — Learned facets from the source BACnet object.
- Facets — Facets in use by the parent proxy point.
- Conversion — Conversion used between device facets and point facets (typically "Default").
- Read Value — Last value read from device, expressed in device facets.
- Write Value — (Applies to writable types only) Last value written, using device facets.

Bacnet proxy points

Bacnet proxy points are similar to other driver's proxy points. Refer to "About proxy points" in the *Drivers Guide* for general information.

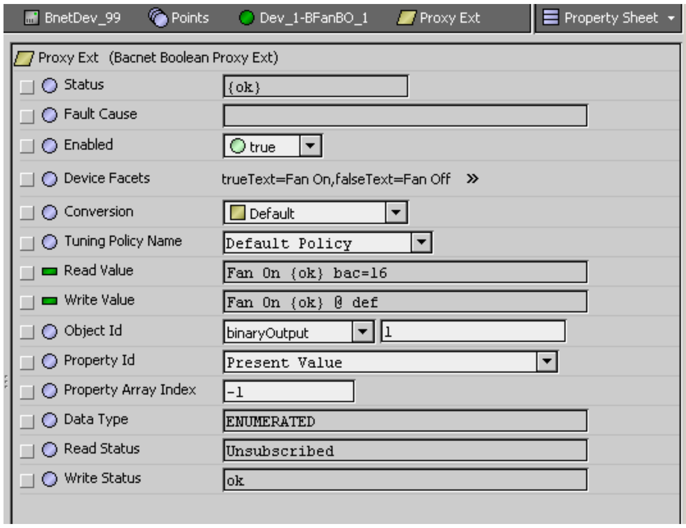
The following sections provide driver-specific details about Bacnet proxy points:

- Bacnet ProxyExt properties
- Bacnet ProxyExt actions
- Status merger for Bacnet proxy points
- Facets usage to poll additional properties (AX-3.2 and later)

Bacnet ProxyExt properties

Figure 3-39 shows the property sheet of an example Bacnet ProxyExt.

Figure 3-39 *Property sheet for a BacnetBooleanProxyExt*



BnetDev_99 | Points | Dev_1-BFanBO_1 | Proxy Ext | Property Sheet

Proxy Ext (Bacnet Boolean Proxy Ext)

Status	{ok}
Fault Cause	
Enabled	true
Device Facets	trueText=Fan On,falseText=Fan Off »
Conversion	Default
Tuning Policy Name	Default Policy
Read Value	Fan On {ok} bac=16
Write Value	Fan On {ok} @ def
Object Id	binaryOutput 1
Property Id	Present Value
Property Array Index	-1
Data Type	ENUMERATED
Read Status	Unsubscribed
Write Status	ok

Note: *To deal with proprietary object types and properties, you can refer to the numerical codes supplied by the device vendor. Then click to highlight the Object Id, or the Property Id, and type in the needed numeric code. For example, a proprietary property 1000 may be included on some company's analog input objects.*

Or, if you have added the relevant enumeration value into the BacnetDevice's Enumeration List property, you should be able to select it by name (normally) from the drop-down list.

In addition to typical ProxyExt properties (refer to "ProxyExt properties" in the *Drivers Guide*), the proxy extension in Bacnet proxy points include these additional properties:

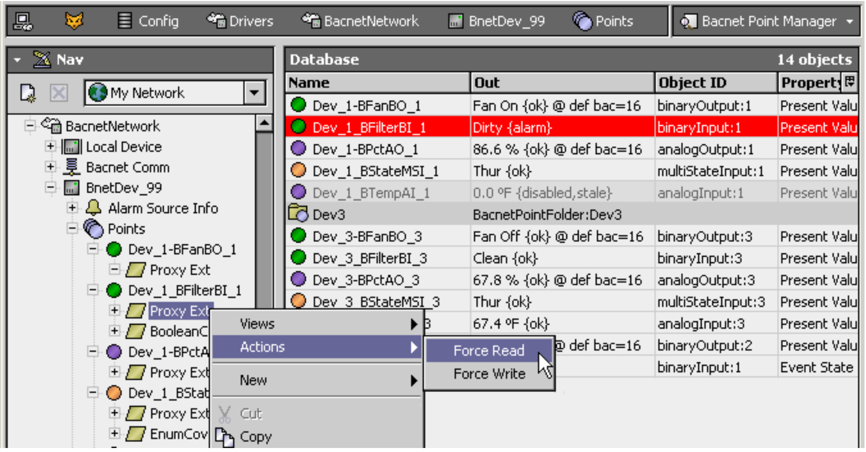
- Object Id
  Consists of two fields: BACnet object *type*, and object instance *number*. Within any one object type in that device, instance numbers must be unique. See "Object ID" on page 3-37, also Note above.
- Property Id
  BACnet property ID for the specific data item being proxied. Often, this is "Present Value." See "Property ID" on page 3-37. If a proprietary property, see the preceding Note.
  Note: Do not change *Property ID on an existing Bacnet proxy point. This can produce undesirable results, especially if a different control point type would apply. Instead, make a new proxy point.*
- Property Array Index
  Index to a particular element in an arrayed property, if applicable. For example, if the point proxies priority level 7 of a priority array input of a Binary_Output point, this value is "7". If not applicable to the property being proxied, this value is "-1".
- Data Type
  (read only) ASN primitive application data type, for example, "REAL" or "ENUMERATED."

- Read Status
  (read only) Indicates whether “Polled,” “COV,” or “Unsubscribed.” See “Read” on page 3-37.
- Write Status
  (read only) Indicates “readonly” or “writable,” or some other value following a Niagara write. See “Write” on page 3-37 for more details.

Bacnet ProxyExt actions

For any Bacnet proxy point, its ProxyExt provides two actions, as shown in Figure 3-40. Note that these actions are on the ProxyExt itself, and not the parent control point.

Figure 3-40 *Actions for any Bacnet ProxyExt*



Database 14 objects

Name	Out	Object ID	Property
Dev_1-BFanBO_1	Fan On {ok} @ def bac=16	binaryOutput:1	Present Valu
Dev_1_BFilterBI_1	Dirty {alarm}	binaryInput:1	Present Valu
Dev_1-BPctAO_1	86.6 % {ok} @ def bac=16	analogOutput:1	Present Valu
Dev_1_BStateMSI_1	Thur {ok}	multiStateInput:1	Present Valu
Dev_1_BTempAI_1	0.0 ºF {disabled,stale}	analogInput:1	Present Valu
Dev3	BacnetPointFolder:Dev3		
Dev_3-BFanBO_3	Fan Off {ok} @ def bac=16	binaryOutput:3	Present Valu
Dev_3_BFilterBI_3	Clean {ok}	binaryInput:3	Present Valu
Dev_3-BPctAO_3	67.8 % {ok} @ def bac=16	analogOutput:3	Present Valu
Dev_3_BStateMSI_3	Thur {ok}	multiStateInput:3	Present Valu
3	67.4 ºF {ok}	analogInput:3	Present Valu
	@ def bac=16	binaryOutput:2	Present Valu
		binaryInput:1	Event State

These actions are briefly described as follows:

- Force Read
  Results in an immediate poll of the source BACnet object’s property.
- Force Write
  Forces an attempt to write from Niagara to the BACnet object’s property. Note that if the property is read-only (or otherwise protected by the BACnet device), an error is seen in the Write Status property.

Status merger for Bacnet proxy points

Bacnet proxy points are unique from proxy points in most other field bus drivers, because the BACnet protocol provides for *native* “abnormal status” of data objects. Niagara can learn about this from the “Status_Flags” property of a BACnet object.

Possible abnormal BACnet “Status_Flags” include the following:

- IN_ALARM — appears as `{alarm}` in Niagara
- FAULT — appears as `{fault}` in Niagara
- OVERRIDDEN — appears as `{overridden}` in Niagara
- OUT_OF_SERVICE — appears as `{disabled}` in Niagara

To get this native status, you do not have to create a proxy point expressly for the “statusFlags” of a BACnet object—however, this is handled *differently* in the Bacnet driver by NiagaraAX release level:

- Starting in AX-3.2, *for polled points*, by default *only the selected property* is polled and reflected in the **`Read Value`** of the ProxyExt. However, you *can* include the polling of other data (including “statusFlags”), by adding to the *facets* of the point. See “Facets usage to poll additional properties” on page 3-40.
  - If “statusFlags” is added to facets, a “status merger” with Niagara point status occurs—just as it does “automatically” for pre-AX-3.2 Bacnet proxy points.
  - If “statusFlags” is *not* added to facets, no status merger occurs—only *Niagara status* is shown for the proxy point (e.g., if the BACnet object is “in_alarm”, it will likely show “ok” in Niagara).

  Prior to AX-3.2, if you create a proxy point for *any other* property (typically, “presentValue” or” eventState”), “statusFlags” is also *automatically* polled. Values from both properties reflect in the **`Read Value`** of the ProxyExt. A “status merger” with Niagara point status occurs.
- Starting in AX-3.7u1, default behavior of Bacnet proxy points *using COV* (not polling) *changed*, to now match polled point behavior for status. Before this change, by default, proxy points using COV included “Status_Flags” data, where a “status merger” was used. For related details, see “Changes in COV statusFlags reporting” on page 3-25.

Status merger operation

In the parent proxy point, any of the above “native” abnormal statuses are *OR’ed* with the equivalent Niagara-originated statuses, and *merged* with Niagara-only status flags, such as “stale,” “unackedAlarm,” etc. (see “About point status” and “How status flags are set” in the *User Guide*).

This is mentioned because it is possible to see a proxy point show a status (as one example) “disabled,” and yet it is enabled in Niagara (the source BACnet object is set to “Out of Service”). Or, you may encounter a variety of other combinations.

Note that the “Read Value” property of a point’s ProxyExt should *only* show the BACnet status, so it can be used to distinguish between Niagara and BACnet contributions to the status bit string. Also, consider the possibility of any independent alarm and fault parameters between the source BACnet object, and any possible NiagaraAX alarm extension on the proxy point.

Facets usage to poll additional properties

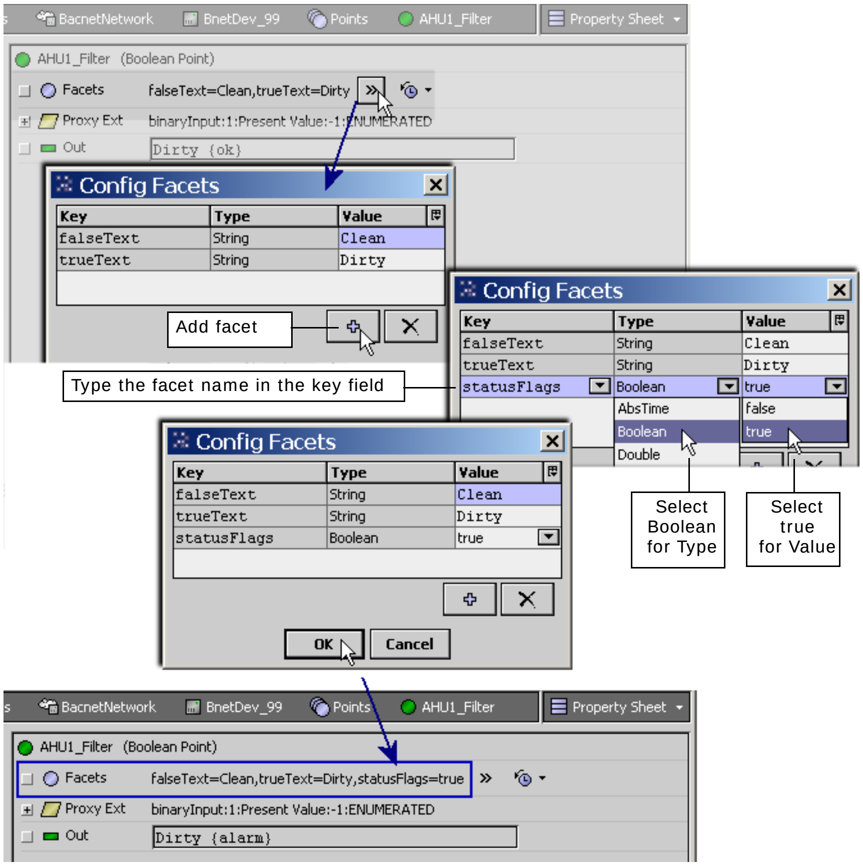
As previously mentioned, starting in AX-3.2 only the *single selected property* of the BACnet object *is polled by default*, for any Bacnet proxy point.For example, if you add a proxy point for a Binary Input object and select “presentValue” as its Property Id, by default that is the *only* value polled in the source object. If that BACnet object was to have a native “in_alarm” status, you would have no indication in Niagara—it would show only Niagara point status, such as “ok”.

Note: *Starting in AX-3.7u1, a similar change was made for Bacnet proxy points using COV (not polling). For related details, see “Changes in COV statusFlags reporting” on page 3-25.*

However (if needed), you can *edit* the *point facets* of any Bacnet proxy point *to include additional properties* (beyond the configured property) for polling—*one* of which could be “statusFlags.” Note this facets edit applies to the main *point’s* facets (and not “device facets” in its ProxyExt).

The facet edit sequence in Figure 3-41 shows facets editing (in AX-3.3) to include statusFlags.

Figure 3-41 *Adding “statusFlags” facet to point’s facets for polling*



AHU1_Filter (Boolean Point)
Facets falseText=Clean,trueText=Dirty
Proxy Ext binaryInput:1:Present Value:-1:ENUMERATED
Out Dirty {ok}

Config Facets

Key	Type	Value
falseText	String	Clean
trueText	String	Dirty

Add facet

Type the facet name in the key field

Config Facets

Key	Type	Value
falseText	String	Clean
trueText	String	Dirty
statusFlags	Boolean	true

AbsTime
Boolean
Double

false
true

Select Boolean for Type

Select true for Value

Config Facets

Key	Type	Value
falseText	String	Clean
trueText	String	Dirty
statusFlags	Boolean	true

AHU1_Filter (Boolean Point)
Facets falseText=Clean,trueText=Dirty,statusFlags=true
Proxy Ext binaryInput:1:Present Value:-1:ENUMERATED
Out Dirty {alarm}

To add to the point poll using this technique, add Boolean facet(s) with any of these (Key) names:

- *statusFlags* for the Status_Flags property;
- *priorityArray* for the Priority_Array property;

- *eventState* for the Event_State property;
- *reliability* for the Reliability property

After adding additional facet(s), resulting metadata from additional polled properties is reflected in the status of a proxy point in the following ways:

- *statusFlags* are included by *merging* with the Niagara proxy point status (bits of BStatus). See "Status merger for Bacnet proxy points" on page 3-39 for more details.
- *priorityArray* is included as it was in pre-AX-3.2 point status, showing "`bac=`*`X`*", where *`X`* is 1 to 16.
- *eventState* is included as a facet showing "`state=`*`stateName`*", where *`stateName`* is the name of a BacnetEventState enumeration, such as `normal`, `offnormal`, `highLimit`, and so on.
- *reliability* is included as a facet showing "`reliability=`*`reliabilityEnum`*", where reliabilityEnum is one of the BacnetReliability values, such as `noFaultDetected`, `overRange`, and so on.

Note: *To reproduce the "pre-AX-3.2" status display of a proxy point configured to poll for "presentValue" of a commandable BACnet object (with a priorityArray, such as an Analog_Output, Binary_Output, etc.), you need to use the technique described above to add two facets to the proxy point:*

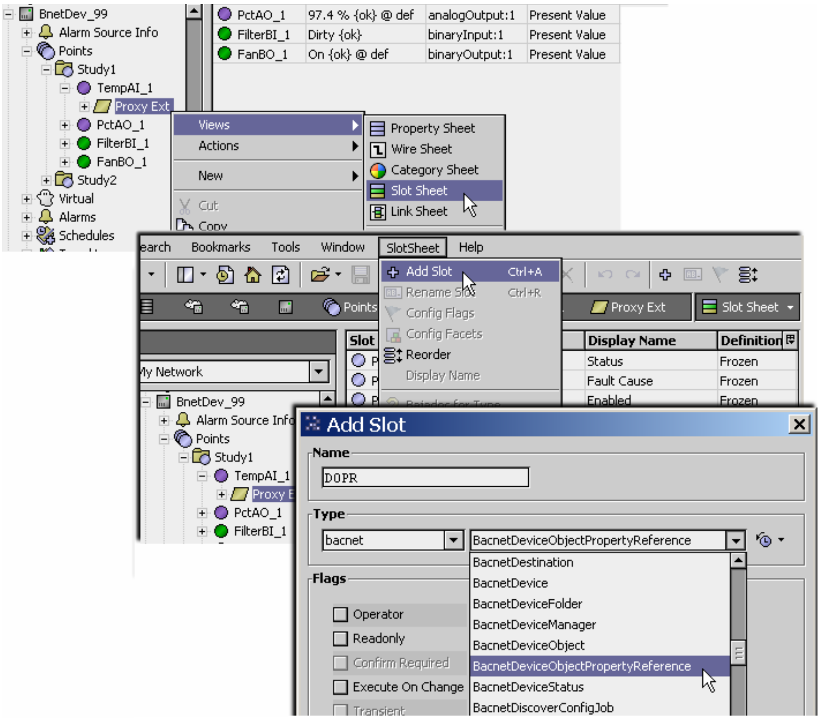
- statusFlags
- priorityArray

Advanced "add DOPR slot" to ProxyExt method

In AX-3.2 and later, in addition to editing the proxy point's facets to poll additional properties (see "Facets usage to poll additional properties" on page 3-40), the point's *ProxyExt* can have a slot added that points to a specific property for additional polling—even one in a different BACnet object and/or BACnet device. This technique is based on a "Bacnet*D*evice*O*bject*P*roperty*R*eference" format (or *DOPR*, for short), where numerical codes are required for processing.

Figure 3-42 shows a slot of the proper type being added to the ProxyExt of a proxy point.

Figure 3-42 *Adding slot in proxy point for DOPR poll method*



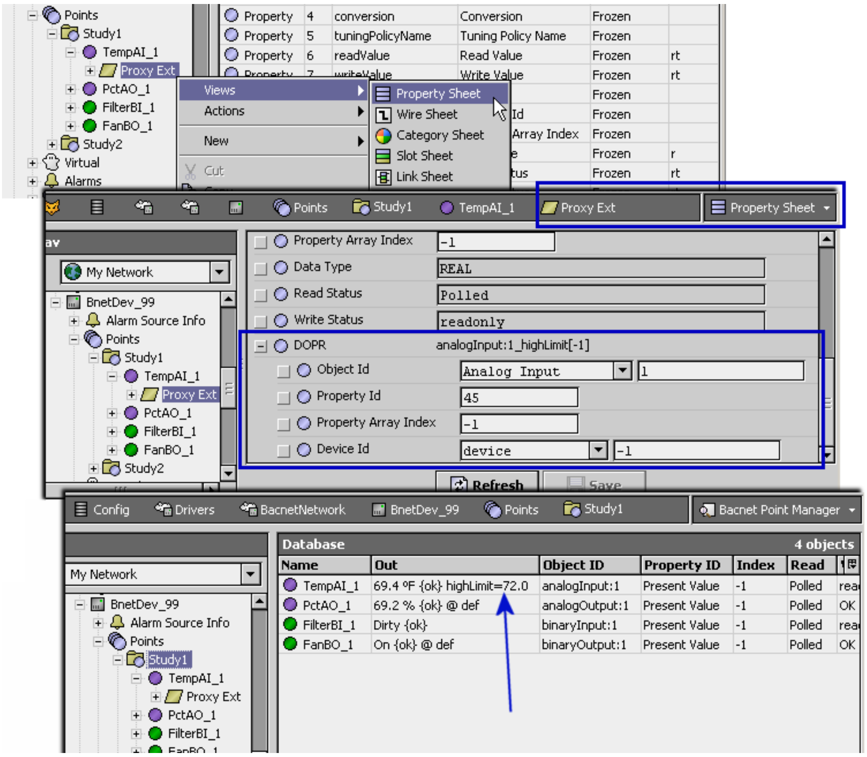
PctAO_1	97.4 % {ok} @ def	analogOutput:1	Present Value
FilterBI_1	Dirty {ok}	binaryInput:1	Present Value
FanBO_1	On {ok} @ def	binaryOutput:1	Present Value

Slot	Display Name	Definition
	Status	Frozen
	Fault Cause	Frozen
	Enabled	Frozen

Add Slot
Name: DOPR
Type: bacnet, BacnetDeviceObjectPropertyReference

After adding the slot, go to ProxyExt's property sheet and edit the new slot's properties to poll/display the property needed. Figure 3-43 shows a DOPR example for property "highLimit" (Property Id 45) of Analog Input 1 in the same device (device -1).

Figure 3-43 *Configuring properties in DOPR slot in ProxyExt for DOPR poll method*



Property	4	conversion	Conversion	Frozen	
Property	5	tuningPolicyName	Tuning Policy Name	Frozen	
Property	6	readValue	Read Value	Frozen	rt
Property	7	writeValue	Write Value	Frozen	rt

Property Array Index: -1
Data Type: REAL
Read Status: Polled
Write Status: readonly
DOPR: analogInput:1_highLimit[-1]
Object Id: Analog Input 1
Property Id: 45
Property Array Index: -1
Device Id: device -1

Name	Out	Object ID	Property ID	Index	Read	
TempAI_1	69.4 °F {ok} highLimit=72.0	analogInput:1	Present Value	-1	Polled	rea
PctAO_1	69.2 % {ok} @ def	analogOutput:1	Present Value	-1	Polled	OK
FilterBI_1	Dirty {ok}	binaryInput:1	Present Value	-1	Polled	rea
FanBO_1	On {ok} @ def	binaryOutput:1	Present Value	-1	Polled	OK

Due to the complexity of this technique, it is expected to be infrequently used. However, in certain applications, such as with Event Enrollment objects (a BACnet object that monitors other objects for the purpose of generating alarms based on its own algorithm), it may prove useful.

The resulting metadata is included with the name of the DOPR. If it is an Event Enrollment object, and the DOPR property is Event_State (the alarm state of the EE object), the facet appears like:

```
<EEinstanceNumber>=<EE event state>
```

if another property, the facet is:

```
<EEinstanceNumber>_<EEpropId>=<prop value>
```

About Bacnet Device's Schedules

The **Schedules** device extension of a BacnetDevice allows you to *import* one or more remote Schedule and/or Calendar objects in a BACnet device into the station as read-only Niagara schedule components. As needed, you can then use these BACnet schedules or calendars in the station.

Also, you can *export* any Niagara schedule component (residing anywhere in the station), into a device's existing BACnet Schedule or Calendar object. In this case, the Niagara schedule has "supervisory control" over the remote BACnet Schedule object, essentially "swapping in" its configuration.

Both of these are "client-side operations," in which you must select the particular BACnet device, then its specific Schedule or Calendar object that you wish to import (from) or export (to).

Do these operations using the two manager views on a BacnetDevice's Schedules extension:

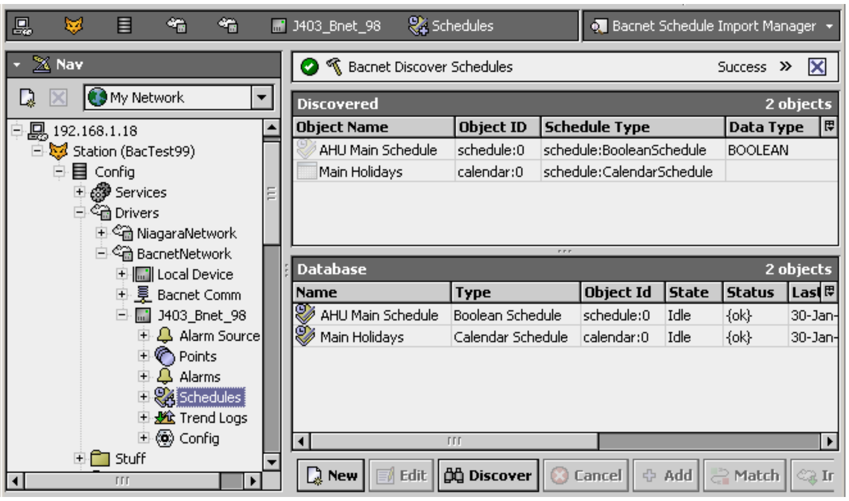
- Bacnet Schedule Import Manager (default view)
- Bacnet Schedule Export Manager

Note: *A BACnet device must contain a Schedule object or Calendar object to make use of these functions. Sometimes, a BACnet device may have neither type of object. A Discover command in either schedule manager view will determine this—if no Schedule or Calendar objects are found, that BacnetDevice's Schedules extension has no practical application.*

About the Bacnet Schedule Import Manager

Double-click **Schedules** under any BacnetDevice to access its Bacnet Schedule Import Manager. Figure 3-44 shows this view in "Learn mode" after a discover, with BACnet schedules added.

Figure 3-44 *Bacnet Schedule Import Manager (Learn mode shown)*



Discovered — 2 objects

Object Name	Object ID	Schedule Type	Data Type
AHU Main Schedule	schedule:0	schedule:BooleanSchedule	BOOLEAN
Main Holidays	calendar:0	schedule:CalendarSchedule	

Database — 2 objects

Name	Type	Object Id	State	Status	Last
AHU Main Schedule	Boolean Schedule	schedule:0	Idle	{ok}	30-Jan-
Main Holidays	Calendar Schedule	calendar:0	Idle	{ok}	30-Jan-

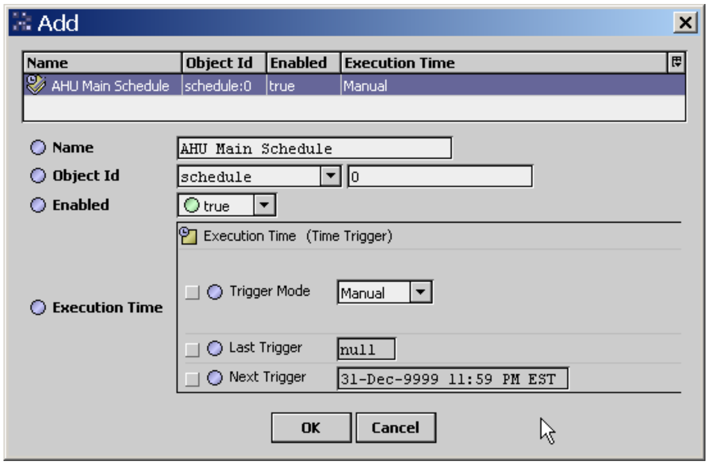
Imported schedules and calendars are Niagara schedule components with a BacnetScheduleImportExt. Events in an imported schedule (or calendar) are obtained from that BACnet device, and are *read-only*. By default, the Schedules device extension also contains a Retry Trigger, for automatic usage as needed. For details see "About the Retry Trigger" in the *Drivers Guide*.

Note that the Schedules device extension can also contain BacnetSchedule*Export* components. These correspond to *Niagara* schedules or calendars that are "pushed" to specific BACnet Schedule or Calendar objects in the remote device. See "About the Bacnet Schedule Export Manager" on page 3-44 for related details. Also, see the next section "Notes on Bacnet Schedule Imports".

Notes on Bacnet Schedule Imports

The Add (and Edit) dialog for importing BACnet Schedules and Calendars is shown in Figure 3-45.

Figure 3-45 *Add dialog in Bacnet Schedule Import Manager*



Name	Object Id	Enabled	Execution Time
AHU Main Schedule	schedule:0	true	Manual

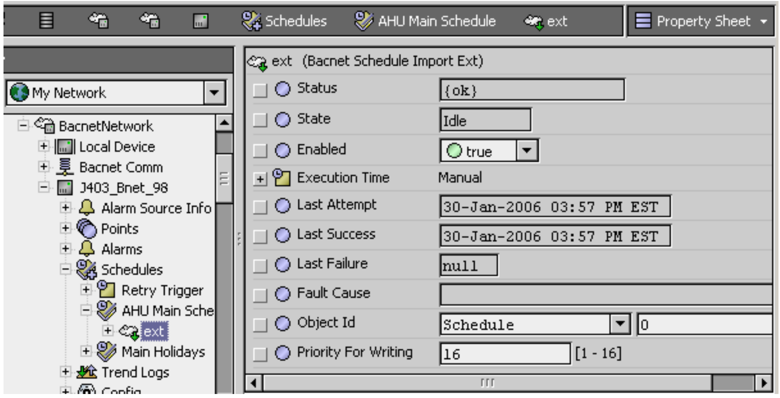
○ Name: AHU Main Schedule
○ Object Id: schedule 0
○ Enabled: ○ true
○ Execution Time: Execution Time (Time Trigger)
□ ○ Trigger Mode: Manual
□ ○ Last Trigger: null
□ ○ Next Trigger: 31-Dec-9999 11:59 PM EST

The default name is the BACnet object's name, and you typically leave this and other properties at defaults. Upon adding, the object's configuration is uploaded to the read-only Niagara schedule component. Note that the default import (read synchronization) from the device's object is "Manual", meaning that you must use the `Import` button in the Bacnet Schedule Import Manager to refresh the Niagara schedule. Alternatively, you can also set the trigger time of the BacnetScheduleImportExt to be either Daily or Interval, if you anticipate ongoing changes in the BACnet object's configuration.

Note: *When the remote BACnet schedule is imported, if it contains a reference to a Calendar object in the device, the Calendar object will also be brought in as a schedule import, in order to keep the schedule in Niagara self-complete.*

Figure 3-46 shows the property sheet of a BacnetScheduleImportExt for an imported schedule.

Figure 3-46 *Property sheet of BacnetScheduleImportExt*



ext (Bacnet Schedule Import Ext)

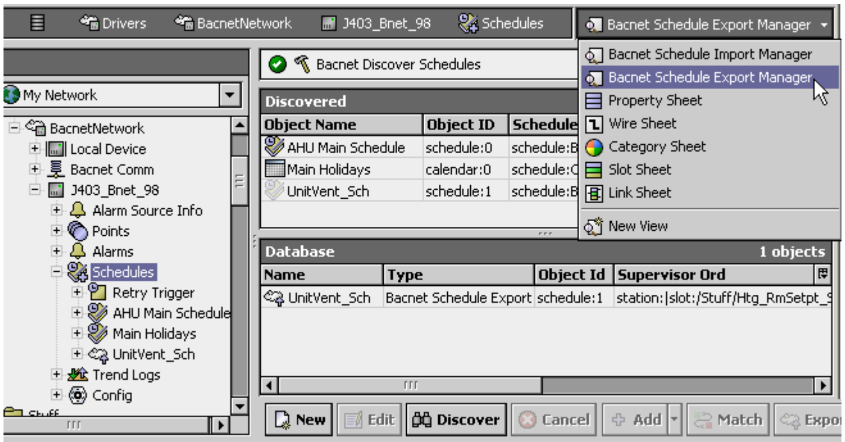
Status	{ok}
State	Idle
Enabled	true
Execution Time	Manual
Last Attempt	30-Jan-2006 03:57 PM EST
Last Success	30-Jan-2006 03:57 PM EST
Last Failure	null
Fault Cause	
Object Id	Schedule 0
Priority For Writing	16 [1 - 16]

Note that the BACnet property "Priority for Writing" is included—however, this means little in the Niagara (station) usage of this imported schedule.

About the Bacnet Schedule Export Manager

Right-click **`Schedules`** under any BacnetDevice and select **`Bacnet Schedule Export Manager`** for this view, or use its view selector (Figure 3-47). This figure shows this view in "Learn mode" after a discover, with one schedule already exported.

Figure 3-47 *Bacnet Schedule Export Manager (Learn mode shown)*



Discovered

Object Name	Object ID	Schedule
AHU Main Schedule	schedule:0	schedule:B
Main Holidays	calendar:0	schedule:C
UnitVent_Sch	schedule:1	schedule:B

Database — 1 objects

Name	Type	Object Id	Supervisor Ord
UnitVent_Sch	Bacnet Schedule Export	schedule:1	station:|slot:/Stuff/Htg_RmSetpt_S

Exported schedules and calendars are schedule export components, each "pointing" to a Niagara schedule component in the station. Events in an exported schedule (or calendar) are written by Niagara (as a *client-side operation*) to the identified BACnet Schedule object in that BACnet device.

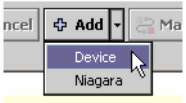
Note: *The BacnetNetwork's* **`Local Device`** *also has a child* **`Export Table`**, *with a Bacnet Export Manager view that permits "exposing" Niagara schedule components in the station as either BACnet Schedule objects or Calendar objects. However, this is a "server type" export, where exported components are made available to any networked BACnet device (and not written to specific objects in a BACnet device, to which we act as a client). See "Niagara Bacnet Server Operation" on page 4-59 for more details.*

The following sections provide more details about exporting to BACnet objects:

- Exporting to BACnet Schedule and Calendar objects
- BacnetScheduleExport properties (including Skip Writes)

Exporting to BACnet Schedule and Calendar objects

Starting in AX-3.1, there are *two* Add options when exporting to BACnet Schedule and Calendars, reflected in the drop-down control beside the **`Add`** button, as shown below:

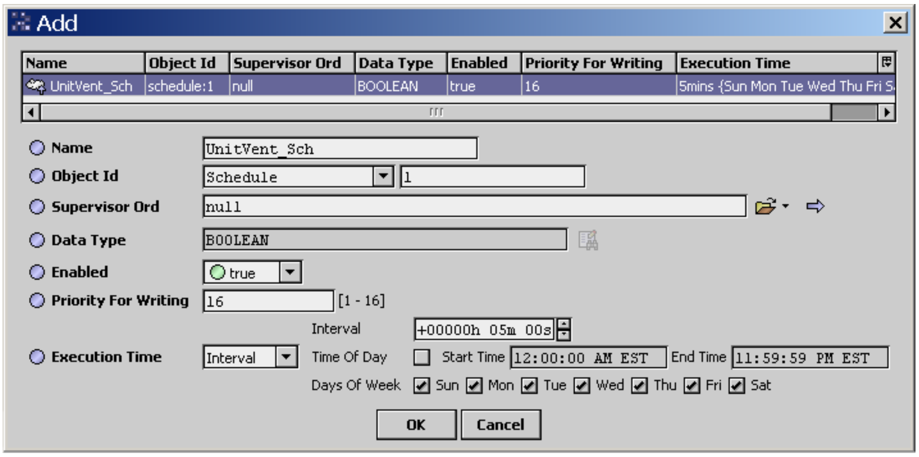


- **`Device`** — Data in the schedule that exists in the BACnet device is initially used, overwriting (initializing) the Niagara schedule. Essentially, this starts the same as if the schedule was imported.

- **`Niagara`** — Data in the Niagara schedule overwrites data in the existing BACnet schedule when exported (how it originally worked in AX-3.0).

It is expected that the "Device" option may often be useful. After choosing an option and clicking **`Add`**, the dialog appears. The **`Add`** (and Edit) dialog for exporting to a BACnet object is shown in Figure 3-48.

Figure 3-48 *Add dialog in Bacnet Schedule Export Manager*



Add

Name	Object Id	Supervisor Ord	Data Type	Enabled	Priority For Writing	Execution Time
UnitVent_Sch	schedule:1	null	BOOLEAN	true	16	5mins {Sun Mon Tue Wed Thu Fri S

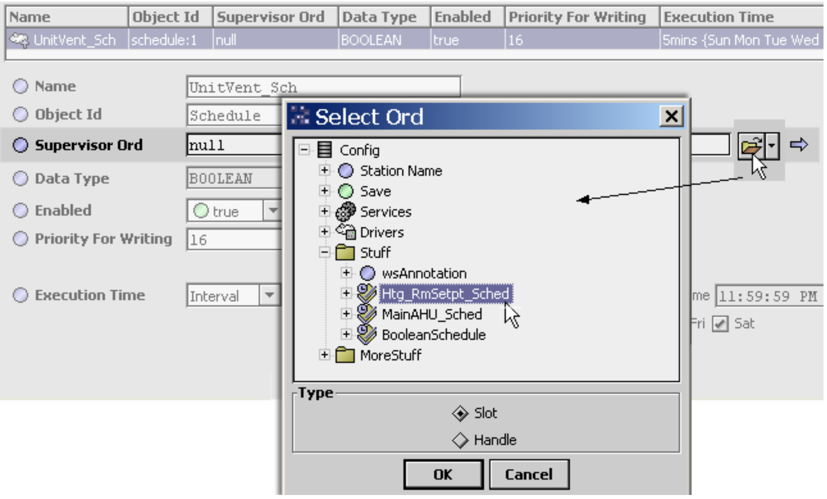
○ Name UnitVent_Sch
○ Object Id Schedule 1
○ Supervisor Ord null
○ Data Type BOOLEAN
○ Enabled ○ true
○ Priority For Writing 16 [1 - 16]
○ Execution Time Interval
Interval +00000h 05m 00s
Time Of Day □ Start Time 12:00:00 AM EST End Time 11:59:59 PM EST
Days Of Week ☑ Sun ☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat

OK Cancel

The default name is the *target* BACnet object's name, and you typically leave this at default. The most critical property is the Supervisor Ord (`null` by default).

In the Supervisor Ord property, click the open folder 📂 for a **`Select Ord`** popup dialog, in which you can navigate to find the *source* Niagara schedule or calendar in the station (Figure 3-49).

Figure 3-49 *Select Ord popup for Supervisor Ord property in BacnetExport*



Name	Object Id	Supervisor Ord	Data Type	Enabled	Priority For Writing	Execution Time
UnitVent_Sch	schedule:1	null	BOOLEAN	true	16	5mins {Sun Mon Tue Wed

○ Name UnitVent_Sch
○ Object Id Schedule
○ Supervisor Ord null
○ Data Type BOOLEAN
○ Enabled ○ true
○ Priority For Writing 16
○ Execution Time Interval

Select Ord

Config
Station Name
Save
Services
Drivers
Stuff
wsAnnotation
Htg_RmSetpt_Sched
MainAHU_Sched
BooleanSchedule
MoreStuff

Type
◈ Slot
◇ Handle

OK Cancel

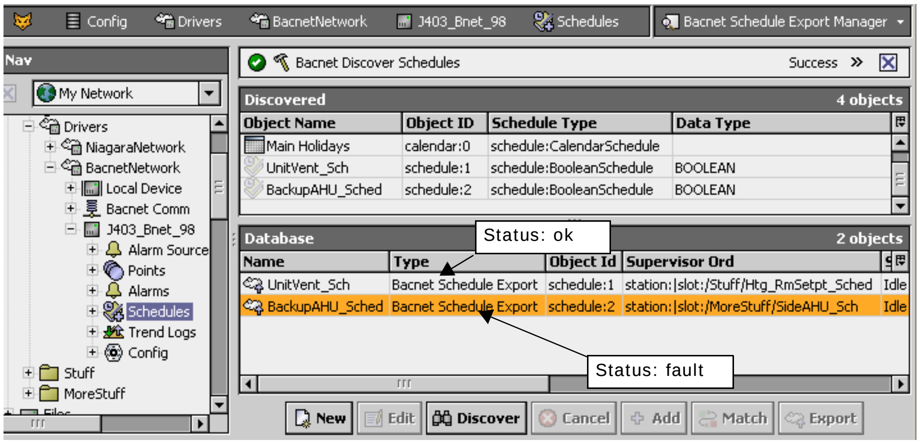
Another property of interest (to BACnet) is the "Priority For Writing" property, which defaults to priority 16. Set this as needed by the application in the BACnet device.

Finally, the default "Execution Time" (write synchronization) to the device's BACnet object is at a continuous 5-minute "Interval". If needed you can adjust this, or set to "Daily" or even "Manual" (whereby an **`Export`** from the Bacnet Schedule Export Manager is required).

When you click **`OK`** in the **`Add`** dialog to create the schedule export, the component is added to the database, and an attempt is made to write the (source) Niagara configuration into the (target) BACnet Schedule or Calendar object. If successful, the status of the BacnetScheduleExport component remains "`ok`," and its row in the Database table of the export manager remains uncolored (white).

However, if any portion of the BACnet write failed, the BacnetScheduleExport has a "fault" status, and its row in the export manager's Database table appears colored orange, as shown in Figure 3-50.

Figure 3-50 *Fault status (orange) indicates write problem to target BACnet schedule*



Discovered — 4 objects

Object Name	Object ID	Schedule Type	Data Type
Main Holidays	calendar:0	schedule:CalendarSchedule	
UnitVent_Sch	schedule:1	schedule:BooleanSchedule	BOOLEAN
BackupAHU_Sched	schedule:2	schedule:BooleanSchedule	BOOLEAN

Database — 2 objects

Name	Type	Object Id	Supervisor Ord	S
UnitVent_Sch	Bacnet Schedule Export	schedule:1	station:|slot:/Stuff/Htg_RmSetpt_Sched	Idle
BackupAHU_Sched	Bacnet Schedule Export	schedule:2	station:|slot:/MoreStuff/SideAHU_Sch	Idle

Status: ok

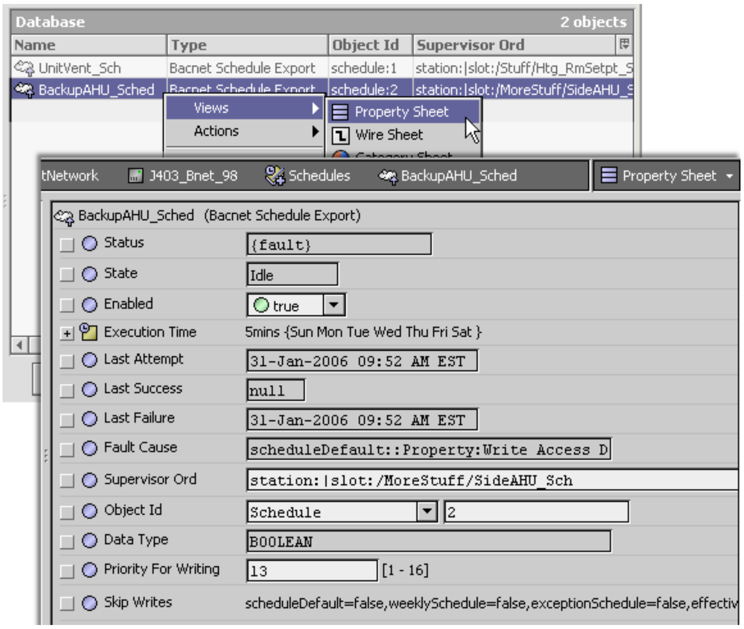
Status: fault

In the case of fault, access the BacnetScheduleExport properties and examine the Fault Cause, and if necessary adjust the Skip Writes property.

BacnetScheduleExport properties

To access the property sheet of a BacnetScheduleExport component, right-click it from either the Nav side bar, or within the Database table of the export manager (as shown in Figure 3-51).

Figure 3-51 *Property sheet of BacnetScheduleExport component*



Database — 2 objects

Name	Type	Object Id	Supervisor Ord
UnitVent_Sch	Bacnet Schedule Export	schedule:1	station:|slot:/Stuff/Htg_RmSetpt_S
BackupAHU_Sched	Bacnet Schedule Export	schedule:2	station:|slot:/MoreStuff/SideAHU_S

BackupAHU_Sched (Bacnet Schedule Export)

Status	{fault}
State	Idle
Enabled	true
Execution Time	5mins {Sun Mon Tue Wed Thu Fri Sat }
Last Attempt	31-Jan-2006 09:52 AM EST
Last Success	null
Last Failure	31-Jan-2006 09:52 AM EST
Fault Cause	scheduleDefault::Property:Write Access D
Supervisor Ord	station:|slot:/MoreStuff/SideAHU_Sch
Object Id	Schedule 2
Data Type	BOOLEAN
Priority For Writing	13 [1 - 16]
Skip Writes	scheduleDefault=false,weeklySchedule=false,exceptionSchedule=false,effectiv

Included is a "Fault Cause" property that provides a text string description if a write operation failed to the target BACnet Schedule or Calendar object. For example:

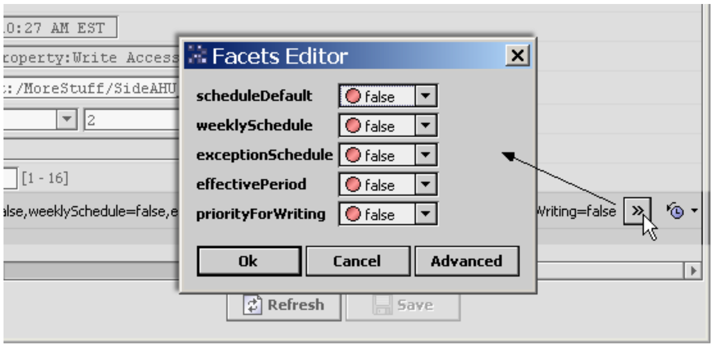
"`scheduleDefault::Property:Write Access Denied`".

In addition, a Skip Writes property lets you adjust which property types in a target BACnet Schedule object are written to, upon an export from a Niagara weekly schedule component.

About schedule export Skip Writes Depending on the BACnet device's implementation by vendor, some properties of its Schedule objects may be read-only. For example, a BACnet Schedule object may allow writes to its weekly schedule events, but not to its exception schedule ("Special Events" in Niagara), if they are read-only (or perhaps do not even exist). Or, the object's "Priority_For_Writing" property may be read-only, or the object may not even have a weekly schedule or exception schedule (a Schedule must only have one or the other, it may have both).

To allow for this, a BacnetScheduleExport component provides a "Skip Writes" property in which you can specify the properties to be written upon an export from Niagara. From the property sheet, click the far-right side control for a popup **`Facets Editor`**, as shown in Figure 3-52.

Figure 3-52 *Skip Writes property of BacnetScheduleExport component*



Facets Editor

- scheduleDefault: false
- weeklySchedule: false
- exceptionSchedule: false
- effectivePeriod: false
- priorityForWriting: false

As shown above, the default is to *not skip* writing any properties upon export (`false` for all), meaning that *all property areas* of the source Niagara schedule component *are written* to the target BACnet Schedule object. As needed, set any of these in the **`Facets Editor`** to `true`, such that Niagara does not attempt to write to them. This can allow an export without a fault (see Figure 3-50).

Note: *Skip Writes does not apply if exporting to a BACnet Calendar (from a CalendarSchedule).*

The Skip Write property areas (facets) are:

- scheduleDefault — Corresponds to “Default Output” of the source Niagara schedule.
- weeklySchedule — Corresponds to the regular day-of-week events of the source Niagara schedule, as defined in its Weekly Scheduler view.
- exceptionSchedule — Corresponds to all “Special Events” of the source Niagara schedule.
- effectivePeriod — As defined by the “Effective Period” in the source Niagara schedule.
- priorityForWriting — As defined by the “Priority For Writing” property in the BacnetScheduleExport component itself (instead of the source Niagara schedule).

About Bacnet Trend Logs (Histories)

The **`Trend Logs`** device extension under the BacnetDevice allows you to import BACnet **`Trend Log`** objects in the device into the station as Niagara histories. For general information on this device extension, see “About the Histories extension” in the *Drivers Guide*.

Note: *Starting in build 3.6.41 of the BACnet driver, the* **`Trend Logs`** *device extension also supports BACnet* **`Trend Log Multiple`** *objects, in addition to* **`Trend Log`** *objects.*

- A BACnet Trend Log Multiple object can monitor/record values of one or more properties of one or more referenced objects, either in the same device as the Trend Log Multiple object, or in an external device. A key feature is that all values are recorded using the *same timestamp*.
- Instead of a BacnetHistoryImport descriptor, Niagara uses a **`BacnetTrendLogMultipleImport`** descriptor to import a Trend Log Multiple object. One import descriptor typically results in *multiple* Niagara histories in the station—one history per object/property value in the record.
- Each BacnetTrendLogMultipleImport descriptor component has a special default view: the **`Bacnet Trend Multiple View`**. For more details, see “Bacnet Trend Multiple View” on page 3-51.

See the following sections about Trend Log imports:

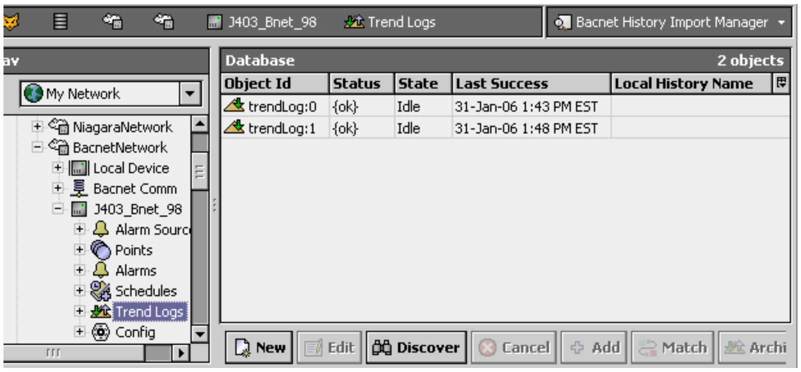
- About the Bacnet History Import Manager
- BACnet Trend Log import notes
- Bacnet Trend Multiple View

Note: *A BACnet device must contain Trend Log objects (and/or Trend Log Multiple objects) to make use of this feature. A Discover command will determine this—if no such objects are found, the* **`Trend Logs`** *extension has no practical application.*

About the Bacnet History Import Manager

The Bacnet History Import Manager is the default view of the **Trend Logs** (history) extension under any BacnetDevice. Simply double-click Trend Logs to open this manager view (Figure 3-53).

Figure 3-53 Bacnet History Import Manager is default view of a device's Trend Logs



Database 2 objects

Object Id	Status	State	Last Success	Local History Name
trendLog:0	{ok}	Idle	31-Jan-06 1:43 PM EST	
trendLog:1	{ok}	Idle	31-Jan-06 1:48 PM EST	

You discover **Trend Log** objects in the BACnet device, and add them as history import components, each of which displays as one row in this manager view. For each import component added, a corresponding Niagara history is made in the station, and is populated with data from the BACnet Trend Log object.

Starting in build 3.6.41, support was added for **Trend Log Multiple** objects, also discoverable in the Bacnet History Import Manager. These are also added as history import components occupying *one* row in the view. However, each one typically results in *multiple* Niagara histories in the station. These history import components also have a special *view*. See "Bacnet Trend Multiple View" on page 3-51.

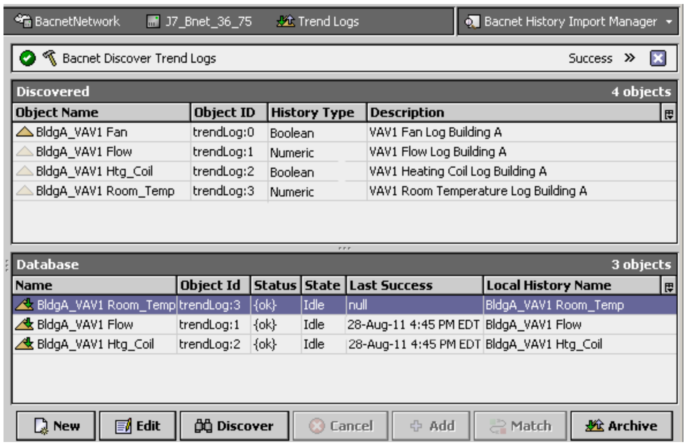
By default, the **Trend Logs** container also has a Retry Trigger, for automatic usage as needed. For details, refer to "About the Retry Trigger" in the *Drivers Guide*.

See "BACnet Trend Log import notes" for more details.

BACnet Trend Log import notes

As in other manager views, when you click **Discover** the Bacnet History Import Manager goes to "Learn mode," split into two panes, and a Trend Logs discover job occurs (Figure 3-54). The figure below shows three Trend Logs already imported in the station.

Figure 3-54 Learn mode in Bacnet History Import Manager



Bacnet Discover Trend Logs Success

Discovered 4 objects

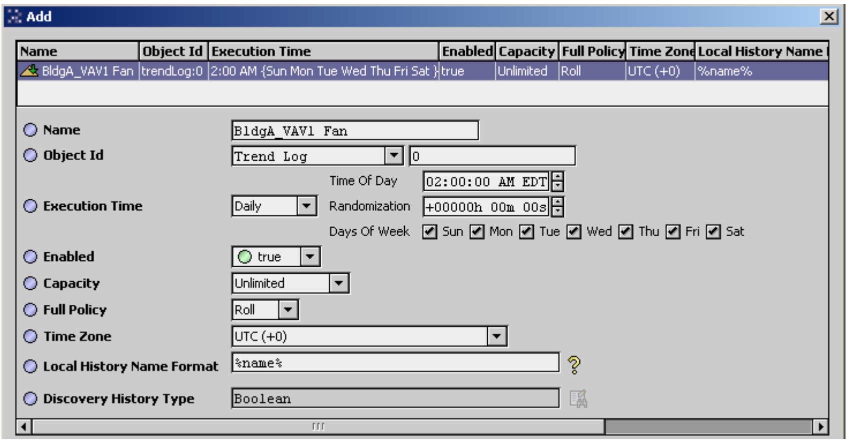
Object Name	Object ID	History Type	Description
BldgA_VAV1 Fan	trendLog:0	Boolean	VAV1 Fan Log Building A
BldgA_VAV1 Flow	trendLog:1	Numeric	VAV1 Flow Log Building A
BldgA_VAV1 Htg_Coil	trendLog:2	Boolean	VAV1 Heating Coil Log Building A
BldgA_VAV1 Room_Temp	trendLog:3	Numeric	VAV1 Room Temperature Log Building A

Database 3 objects

Name	Object Id	Status	State	Last Success	Local History Name
BldgA_VAV1 Room_Temp	trendLog:3	{ok}	Idle	null	BldgA_VAV1 Room_Temp
BldgA_VAV1 Flow	trendLog:1	{ok}	Idle	28-Aug-11 4:45 PM EDT	BldgA_VAV1 Flow
BldgA_VAV1 Htg_Coil	trendLog:2	{ok}	Idle	28-Aug-11 4:45 PM EDT	BldgA_VAV1 Htg_Coil

To import Trend Logs (or Trend Log Multiples), you select them in the top Discovered pane, and then click **Add**, producing the **Add** dialog (Figure 3-55). This dialog shows *some* of a BacnetHistoryImport component's properties.

Figure 3-55 *Add dialog in Bacnet History Import Manager*



Add

Name	Object Id	Execution Time	Enabled	Capacity	Full Policy	Time Zone	Local History Name
BldgA_VAV1 Fan	trendLog:0	2:00 AM {Sun Mon Tue Wed Thu Fri Sat }	true	Unlimited	Roll	UTC (+0)	%name%

○ Name	BldgA_VAV1 Fan
○ Object Id	Trend Log ▼ 0
○ Execution Time	Daily ▼ Time Of Day 02:00:00 AM EDT Randomization +00000h 00m 00s Days Of Week ☑ Sun ☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat
○ Enabled	○ true ▼
○ Capacity	Unlimited ▼
○ Full Policy	Roll ▼
○ Time Zone	UTC (+0) ▼
○ Local History Name Format	%name%
○ Discovery History Type	Boolean

Fields in the **Add** (and Edit) dialog for a history import descriptor in the Bacnet History Import Manager are as follows:

- Name
  The BACnet Trend Log (or Trend Log Multiple) object's name.
- Object Id
  The BACnet object type, either Trend Log or Trend Log Multiple, and its instance number for that type, unique within the device.
- Execution Time
  The frequency at which data is imported from the remote Trend Log (or Trend Log Multiple) into the Niagara history (or histories). Default is daily, at 2:00 AM.
- Enabled
  Must be true (default) to import Trend Log or Trend Log Multiple data into the Niagara history.
- Capacity
  Capacity of the imported Niagara history or histories, defaulting to unlimited.
- Full Policy
  Niagara policy when history or histories reaches capacity.
- Time Zone
  Specifies the "best guess" time zone offset, taking daylight savings time in consideration.
- Local History Name Format
  Allows "BFormat" (Baja Format) syntax to specify the name of the Niagara history initially created and populated by the Trend Log data. The default value is `%name%`, which recreates the BACnet name of the Trend Log object. You can modify this with other BFormat scripting, for example: `%parent.parent.displayName% %name%`, which typically would be the name of the Niagara BACnetDevice component and the BACnet Trend Log object name, e.g. `J7Bnet BldgA_VAV1_Fan`
  Or, you can enter static text for the history name—or, some combination of static text and valid BFormat scripting text.
  Note in the case of Trend Log Multiple import descriptor (BacnetTrendLogMultipleImport), the default `%name%` value results in histories named using the BACnet name of the Trend Log Multiple object, with underscore delimiters adding the BACnet names of each _object_property record. See Figure 3-60 on page 52 for an example.

Note that after adding, you can access additional BacnetHistoryImport properties from the property sheet of any BacnetHistoryImport or BacnetTrendLogMultipleImport. Also the following right-click actions are available:

- Execute
  Like an **Archive** from the manager, this updates the Niagara history from the Trend Log data.
- Clear Records in Device
  Produces a confirmation dialog. If answered "`Yes`," the attempts to clear all records in the source BACnet Trend Log object. However, note that records in the imported Niagara history remain.

Also see the next section, "About histories imported from Trend Logs".

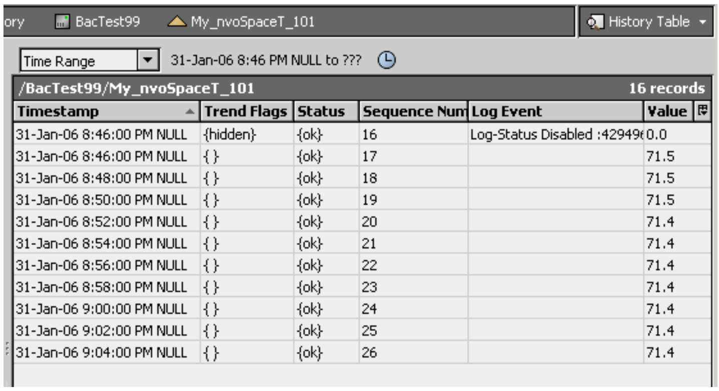
About histories imported from Trend Logs

Niagara histories imported from BACnet Trend Logs (and Trend Log Multiples) have additional fields in each record, namely "Sequence Number" and "Log Event," which reflect BACnet-required items. These fields do not affect "History Chart" views of these histories, but do require slightly more storage space.

Note: *The concept of sequence number was introduced with Addendum B to the 2001 edition of the specification. Trend Logs that were implemented against earlier editions (which includes logs from R2 Niagara stations, and Niagara AX histories that were collected with a non-BacnetTrendLogExt) will not have sequence number in their log data, so Niagara has to access the records by time only.*

Figure 3-56 shows a **`History Table`** view of an example history created by a BacnetHistoryImport.

Figure 3-56 *History Table for history imported from BACnet Trend Log*



ory | BacTest99 | My_nvoSpaceT_101 | History Table

Time Range 31-Jan-06 8:46 PM NULL to ???

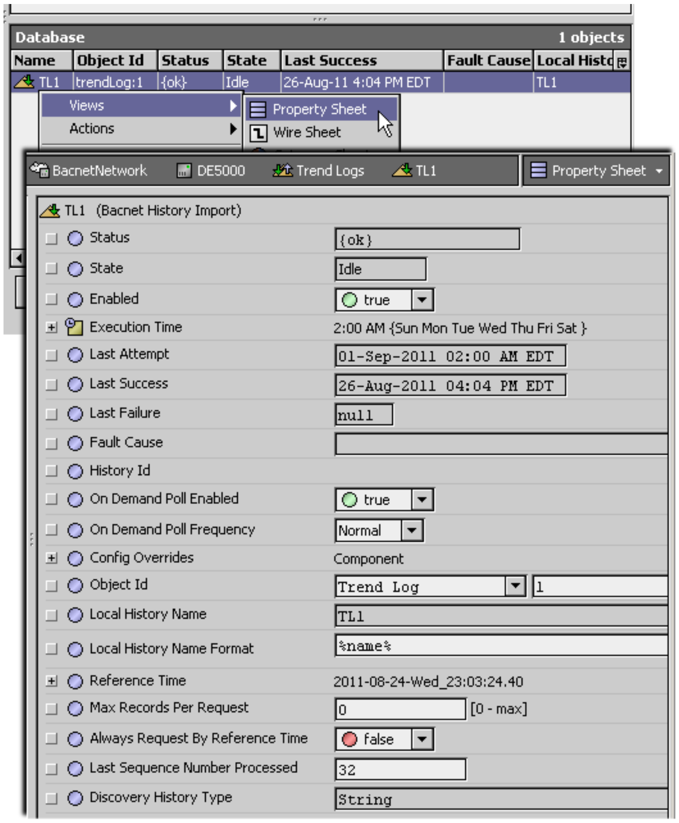
/BacTest99/My_nvoSpaceT_101 — 16 records

Timestamp	Trend Flags	Status	Sequence Num	Log Event	Value
31-Jan-06 8:46:00 PM NULL	{hidden}	{ok}	16	Log-Status Disabled :42949	0.0
31-Jan-06 8:46:00 PM NULL	{ }	{ok}	17		71.5
31-Jan-06 8:48:00 PM NULL	{ }	{ok}	18		71.5
31-Jan-06 8:50:00 PM NULL	{ }	{ok}	19		71.5
31-Jan-06 8:52:00 PM NULL	{ }	{ok}	20		71.4
31-Jan-06 8:54:00 PM NULL	{ }	{ok}	21		71.4
31-Jan-06 8:56:00 PM NULL	{ }	{ok}	22		71.4
31-Jan-06 8:58:00 PM NULL	{ }	{ok}	23		71.4
31-Jan-06 9:00:00 PM NULL	{ }	{ok}	24		71.4
31-Jan-06 9:02:00 PM NULL	{ }	{ok}	25		71.4
31-Jan-06 9:04:00 PM NULL	{ }	{ok}	26		71.4

BacnetHistoryImport properties

To access the property sheet of a BacnetHistoryImport component (or BacnetTrendLogMultipleImport component), right-click it from either the Nav side bar, or within the Database table of the Bacnet History Import Manager (as shown in Figure 3-57).

Figure 3-57 *Property sheet of BacnetHistoryImport component*



Database — 1 objects

Name	Object Id	Status	State	Last Success	Fault Cause	Local Histo
TL1	trendLog:1	{ok}	Idle	26-Aug-11 4:04 PM EDT		TL1

BacnetNetwork | DE5000 | Trend Logs | TL1 | Property Sheet

TL1 (Bacnet History Import)

Status	{ok}
State	Idle
Enabled	true
Execution Time	2:00 AM {Sun Mon Tue Wed Thu Fri Sat }
Last Attempt	01-Sep-2011 02:00 AM EDT
Last Success	26-Aug-2011 04:04 PM EDT
Last Failure	null
Fault Cause	
History Id	
On Demand Poll Enabled	true
On Demand Poll Frequency	Normal
Config Overrides	Component
Object Id	Trend Log 1
Local History Name	TL1
Local History Name Format	%name%
Reference Time	2011-08-24-Wed_23:03:24.40
Max Records Per Request	0 [0 - max]
Always Request By Reference Time	false
Last Sequence Number Processed	32
Discovery History Type	String

The “Config Overrides” properties (capacity, fullPolicy) work as they do in the “History Config” properties of a normal history extension. See the “Configure history extension” section in the *User Guide* for more details.

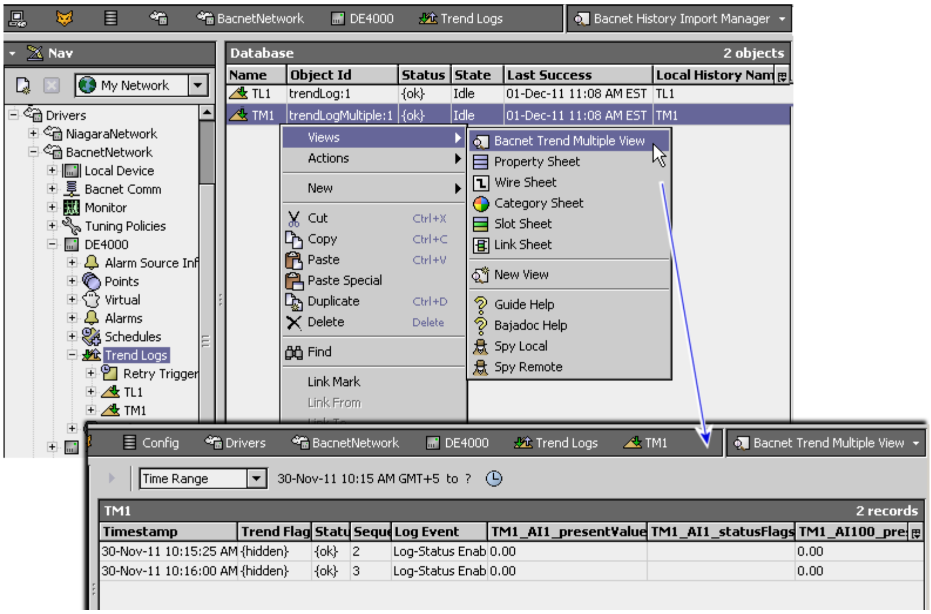
Four of the last five properties affect the special collection process from the imported Trend Log object, and are described as follows:

- Reference Time
  Provides a Date and Time for reference.
- Max Records By Request
  Defines the maximum Trend Log records that can be archived (0 to `max`), with default of 0. Note that with the 0 default, Niagara asks for 10 records per request.
- Always Request By Reference Time
  Whether Trend Log records should always request against the reference time. Default is `false`.
- Last Sequence Number Processed
  Shows the highest numerical BACnet sequence number processed. Note that sequence numbers are included in the imported Niagara history. See "About histories imported from Trend Logs" on page 3-49, including the relevant Note.

Bacnet Trend Multiple View

Starting in the 3.6.41 build of the BACnet driver, client support was add for BACnet *Trend Log Multiple* objects. Using the Bacnet History Import Manager, each Trend Log Multiple object imported into the station has an available **`Bacnet Trend Log Multiple View`** on its associated **`BacnetTrendLogMultipleImport`** descriptor.

Figure 3-58 *Bacnet Trend Multiple View is available on each BacnetTrendLogMultipleImport descriptor*



Database 2 objects

Name	Object Id	Status	State	Last Success	Local History Nam
TL1	trendLog:1	{ok}	Idle	01-Dec-11 11:08 AM EST	TL1
TM1	trendLogMultiple:1	{ok}	Idle	01-Dec-11 11:08 AM EST	TM1

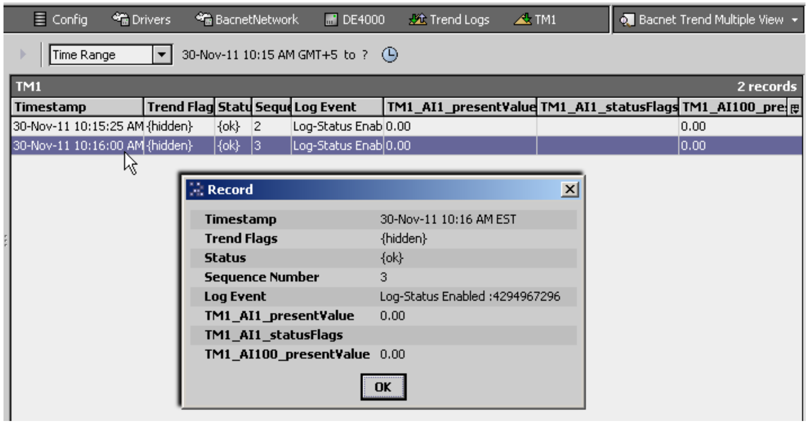
Time Range 30-Nov-11 10:15 AM GMT+5 to ?

TM1 2 records

Timestamp	Trend Flag	Statu	Seque	Log Event	TM1_AI1_presentValue	TM1_AI1_statusFlags	TM1_AI100_pre
30-Nov-11 10:15:25 AM	{hidden}	{ok}	2	Log-Status Enab	0.00		0.00
30-Nov-11 10:16:00 AM	{hidden}	{ok}	3	Log-Status Enab	0.00		0.00

As shown in Figure 3-58, this view provides tabular access to recorded values for all items referenced in the Trend Log Multiple object, along with status and sequence information.

Figure 3-59 *Double-click record in Bacnet Trend Multiple View for popup details*



Time Range 30-Nov-11 10:15 AM GMT+5 to ?

TM1 2 records

Timestamp	Trend Flag	Statu	Sequ	Log Event	TM1_AI1_presentValue	TM1_AI1_statusFlags	TM1_AI100_pre
30-Nov-11 10:15:25 AM	{hidden}	{ok}	2	Log-Status Enab	0.00		0.00
30-Nov-11 10:16:00 AM	{hidden}	{ok}	3	Log-Status Enab	0.00		0.00

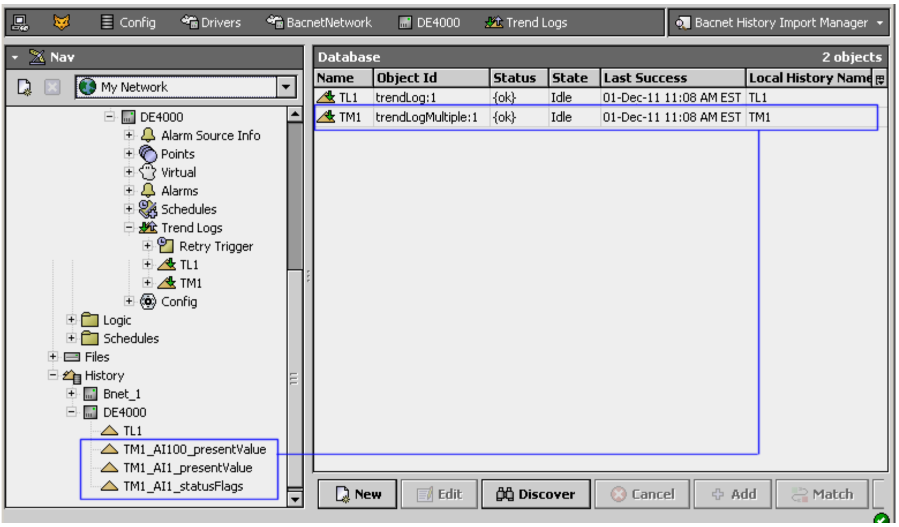
Record

Timestamp	30-Nov-11 10:16 AM EST
Trend Flags	{hidden}
Status	{ok}
Sequence Number	3
Log Event	Log-Status Enabled :4294967296
TM1_AI1_presentValue	0.00
TM1_AI1_statusFlags	
TM1_AI100_presentValue	0.00

You can double-click any record (row) in the **Bacnet Trend Multiple View** for a popup window showing this same information, as shown in Figure 3-59. This example Trend Log Multiple shown has three different items trended—two properties of object “AI1” (Present_Value and Status_Flags) and one property of object “AI100” (Present_Value).

The separate Niagara histories created by importing a Trend Log Multiple object are also recognizable in the station’s Nav tree, as shown in Figure 3-60 below.

Figure 3-60 *Separate Niagara histories created by importing a BACnet Trend Log Multiple object*



Name	Object Id	Status	State	Last Success	Local History Name
TL1	trendLog:1	{ok}	Idle	01-Dec-11 11:08 AM EST	TL1
TM1	trendLogMultiple:1	{ok}	Idle	01-Dec-11 11:08 AM EST	TM1

By default, the name of each history associated with a Trend Log Multiple import uses the format of `TrendLogMultipleName_objectName_propertyName`, for example `TM1_AI1_presentValue`, as shown above.

About Bacnet virtual points

Starting in AX-3.2, each Bacnet Device includes a child **Virtual** gateway child component that provides access to the “virtual component space” specific to that device. For general information, refer to the “Virtual gateway and components” section in the *Drivers Guide*. Virtual points are under this gateway.

The following main subsections provide more details on Bacnet virtual points:

- Usage of Bacnet virtual points
- Browsing Bacnet virtual points
- Bacnet virtual points in Px views
- Bacnet virtual ord syntax

Usage of Bacnet virtual points

There are two primary uses for Bacnet virtual points:

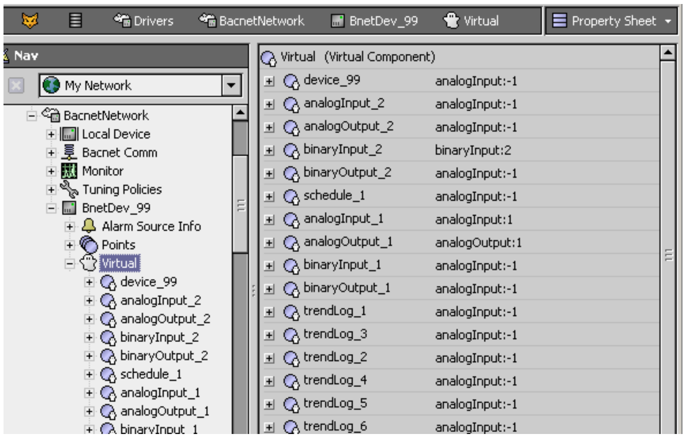
- Px view bindings for simple polling of values, without the station resource overhead of (persisted) proxy points. Upon being unsubscribed, virtual components are simply removed from the poll scheduler and also station memory. The only persisted parts are the *ords* (using virtual syntax) in the Px widget bindings. In many cases, particularly with replicated device applications, this allows a JACE station to graphically monitor *many* more devices than if using just proxy points.
- Quick review and (if needed) adjustments to one or more properties in BACnet objects, from the Workbench property sheets of virtual components. Otherwise, you would typically need to create persisted "Config" type objects, then delete them after reviewing and changing properties.

Virtuals are *transient* vs. persisted components—they are dynamically created (and subscribed) only when accessed, and are not permanently stored in the station database. This precludes any linking to or from virtuals—as links would be lost. Nor are point extensions (alarm, history) supported under virtual components. These things require use of Bacnet proxy points, which are persisted in the station database.

Browsing Bacnet virtual points

Unlike the **`Virtual`** device extension, there is no special view for a virtual gateway—you can simply double-click it to access its property sheet, or expand it in the Nav tree. When you do this for a device's *BacnetVirtualGateway*, a call is made to the device to discover its BACnet objects, each appearing as a virtual object (slot) under the gateway. This request returns the device's "object list" (Figure 3-61).

Figure 3-61 *BacnetVirtualGateway functions as BACnet object list*



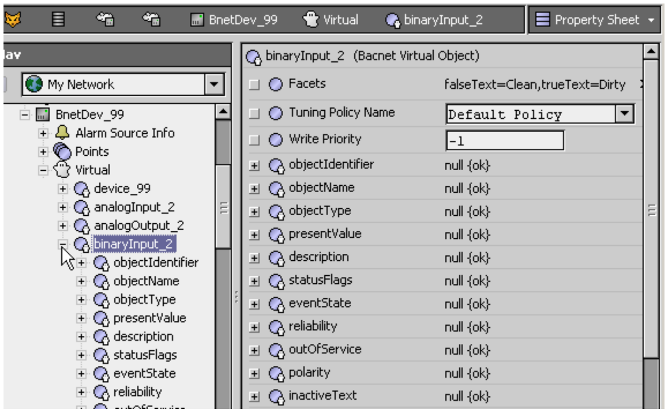
Virtual (Virtual Component)

device_99	analogInput:-1
analogInput_2	analogInput:-1
analogOutput_2	analogInput:-1
binaryInput_2	binaryInput:2
binaryOutput_2	analogInput:-1
schedule_1	analogInput:-1
analogInput_1	analogInput:1
analogOutput_1	analogOutput:1
binaryInput_1	analogInput:-1
binaryOutput_1	analogInput:-1
trendLog_1	analogInput:-1
trendLog_3	analogInput:-1
trendLog_2	analogInput:-1
trendLog_4	analogInput:-1
trendLog_5	analogInput:-1
trendLog_6	analogInput:-1

As shown above, virtual objects are listed by object ID, i.e. `<objectType>_<instanceNumber>`, for example `analogInput_2` or `trendLog_1`. You can expand any Bacnet virtual object to see that object's properties, where each property is a child virtual. See the next section "About BacnetVirtualProperties (AX-3.3 and later)".

Note: *Note if an AX-3.2 station, this varies slightly—see "Bacnet AX-3.2 virtual notes" on page 3-54.*

Figure 3-62 *Properties of a BacnetVirtualObject are BacnetVirtualProperty components (AX-3.3 and later)*



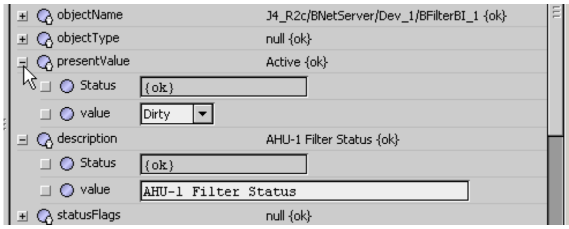
binaryInput_2 (Bacnet Virtual Object)

Facets	falseText=Clean,trueText=Dirty
Tuning Policy Name	Default Policy
Write Priority	-1
objectIdentifier	null {ok}
objectName	null {ok}
objectType	null {ok}
presentValue	null {ok}
description	null {ok}
statusFlags	null {ok}
eventState	null {ok}
reliability	null {ok}
outOfService	null {ok}
polarity	null {ok}
inactiveText	null {ok}

About BacnetVirtualProperties (AX-3.3 and later)

Starting in AX-3.3, the granularity of Bacnet virtual points increased to the “object, property level”, using BacnetVirtualProperty components. Each BacnetVirtualProperty has two properties: Status and value, as shown in Figure 3-63. By default, value automatically includes the object’s units abbreviation, state_text, and so on.

Figure 3-63 Each BacnetVirtualProperty has two properties: Status and value

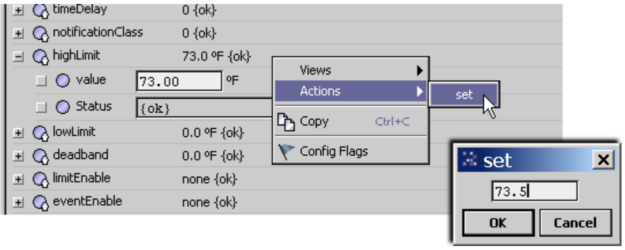


objectName J4_R2c/BNetServer/Dev_1/BFilterBI_1 {ok}
objectType null {ok}
presentValue Active {ok}
Status {ok}
value Dirty
description AHU-1 Filter Status {ok}
Status {ok}
value AHU-1 Filter Status
statusFlags null {ok}

Note that status reflects Niagara poll status, but does not reflect any “intrinsic” BACnet status (such as alarm). Intitially, status changes from “stale” to “ok” upon the first polled read, and typically remains “ok”. Status could possibly change to either “stale” or “down” in certain scenarios.

Each BacnetVirtualProperty also has a “set” action for right-click access, as shown in Figure 3-63.

Figure 3-64 Each BacnetVirtualProperty has a write action



timeDelay 0 {ok}
notificationClass 0 {ok}
highLimit 73.0 ºF {ok}
value 73.00 ºF
Status {ok}
lowLimit 0.0 ºF {ok}
deadband 0.0 ºF {ok}
limitEnable none {ok}
eventEnable none {ok}

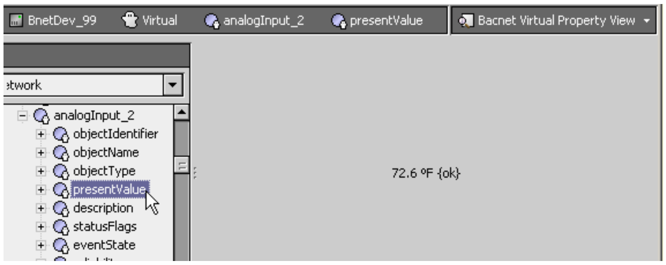
set
73.5

This allows you to easily do “one time” configuration tweaks from the Workbench property sheet of the virtual property—providing that the remote BACnet device permits writes, of course. Note that this write action exists for *every* property of the BACnet object, even read-only or status types. Furthermore, this action is also available on a Px widget bound to any BacnetVirtualProperty. see “Px usage of Bacnet virtuals (AX-3.3 and later)” on page 3-55 for related details.

Note: *You can globally rename the set action (and resultant popup dialog) to something else by editing the bacnet lexicon value for the key named* `BacnetVirtualProperty.set`.

In addition, each BacnetVirtualProperty has a default “Bacnet Virtual Property View” that shows both value and status, as shown in Figure 3-65.

Figure 3-65 Default “Bacnet Virtual Property View” shows both value and poll status



BnetDev_99 Virtual analogInput_2 presentValue Bacnet Virtual Property View

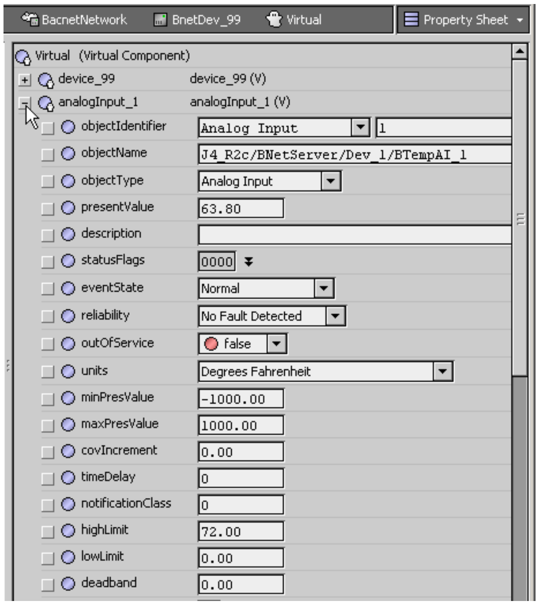
72.6 ºF {ok}

Although it is not an exciting view, it is an option when creating Px bindings and selecting “Workbench view,” and it also provides a quick preview outside of the Px editor.

Bacnet AX-3.2 virtual notes

In AX-3.2 only, Bacnet virtual points are at the “object level” ony, using a single BacnetVirtualComponent to model each BACnet object under the device’s gateway. See Figure 3-66.

Figure 3-66 *Property sheet of BacnetVirtualComponent (AX-3.2 only)*



Virtual (Virtual Component)

device_99	device_99 (V)
analogInput_1	analogInput_1 (V)
objectIdentifier	Analog Input 1
objectName	J4_R2c/BNetServer/Dev_1/BTempAI_1
objectType	Analog Input
presentValue	63.80
description	
statusFlags	0000
eventState	Normal
reliability	No Fault Detected
outOfService	false
units	Degrees Fahrenheit
minPresValue	-1000.00
maxPresValue	1000.00
covIncrement	0.00
timeDelay	0
notificationClass	0
highLimit	72.00
lowLimit	0.00
deadband	0.00

This “virtual view” property sheet reflects current property values. If needed, you can modify selectively and click **`Save`** with new values entered. Unlike with the later AX-3.3 virtual point design, there is no available action on BacnetVirtualComponents, nor is status (of polling) included with property values.

Note: *If making wide use of Bacnet virtual points, it is recommended to use AX-3.3 or later. In addition to providing more features, polling has proven to work better under certain scenarios.*

Bacnet virtual points in Px views

Bacnet virtual points can show real-time values in Px views. Starting in AX-3.3, this includes an available right-click “set” action, as well as the Niagara poll status (ok, stale, down) of the polled property.

Note: *A BacnetVirtualGateway cannot have its “own” Px view, but you can use its child virtual components in Px views for other components, for example on the Bacnet Device itself, or its Points extension, and so on.*

The only persisted (permanent) record any Bacnet virtual point is its *ord* in the Px binding to it, which uses a “Bacnet virtual ord syntax” that includes the BacnetVirtualGateway within the ord. This ord is automatically resolved when you drag a Bacnet virtual component onto the Px page, and make your selection in the popup Px “**`Make Widget`**” dialog.

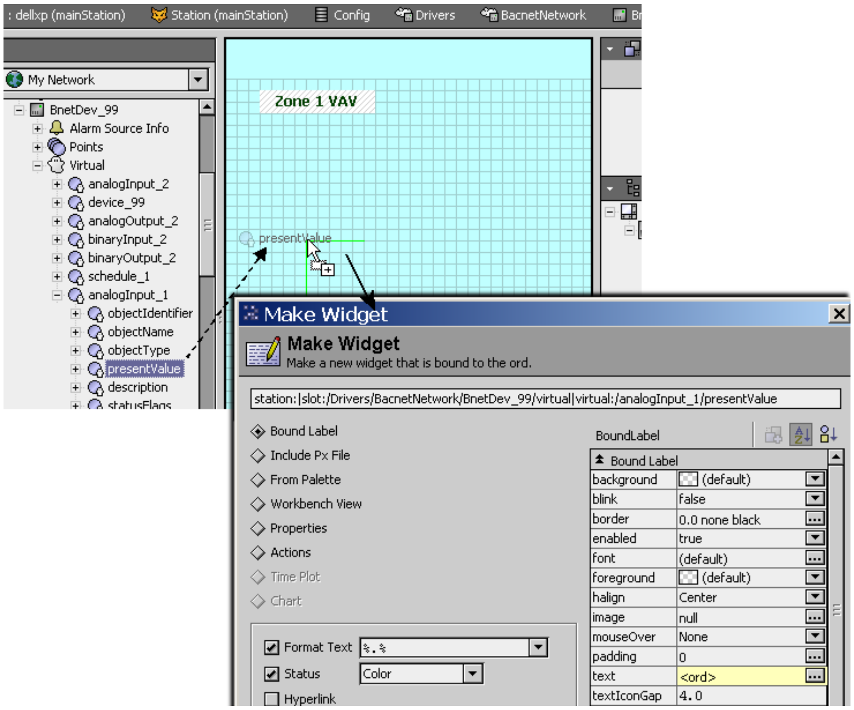
Note: *For complete details on Px view and widget editing, see “About Px Editor” in the* User Guide.

The next two sections explain “Px usage of Bacnet virtuals (AX-3.3 and later)” and “Px usage of Bacnet virtuals in AX-3.2”.

Px usage of Bacnet virtuals (AX-3.3 and later)

Starting in AX-3.3, you can drag a BacnetVirtualProperty into a Px view for a “Make Widget” dialog, as shown in the example in Figure 3-67.

Figure 3-67 *Dragging BacnetVirtualProperty into Px editor view, with resulting Make Widget popup*



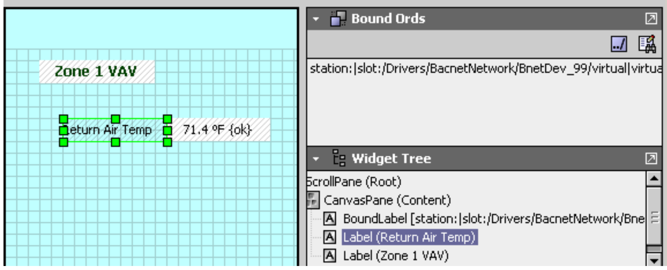
station:|slot:/Drivers/BacnetNetwork/BnetDev_99/virtual|virtual:/analogInput_1/presentValue

background	(default)
blink	false
border	0.0 none black
enabled	true
font	(default)
foreground	(default)
halign	Center
image	null
mouseOver	None
padding	0
text	<ord>
textIconGap	4.0

☑ Format Text %.%
☑ Status Color
□ Hyperlink

Often you may wish to include the "presentValue" property, as shown here. In the **Make Widget** dialog, you can select either "Bound label" (as shown) or "Workbench view" for the "Bacnet Virtual Property View." If using a bound label, the accompanying display name label (if that option was selected) can be edited to another text string, as shown in Figure 3-68.

Figure 3-68 *Binding to BacnetVirtualProperty in Px view*

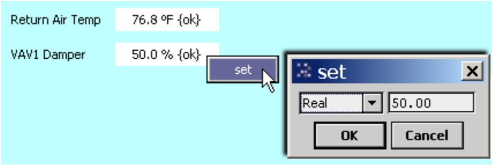


station:|slot:/Drivers/BacnetNetwork/BnetDev_99/virtual|virtua

ScrollPane (Root)
CanvasPane (Content)
BoundLabel [station:|slot:/Drivers/BacnetNetwork/Bne
Label (Return Air Temp)
Label (Zone 1 VAV)

At the time of this document, every BacnetVirtualProperty provides an action to write ("set" is default name), which may be useful in some scenarios. Figure 3-69 shows an example action being invoked.

Figure 3-69 *Default "set" action and resulting popup for write to a present_value property*

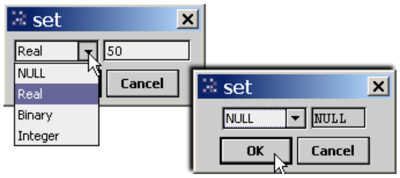


Return Air Temp 76.8 ºF {ok}
VAV1 Damper 50.0 % {ok}
Real 50.00

By default, in the case of a write to the "presentValue" property of a BACnet object with a priority_array (such as an Analog Output or Binary Output), the Niagara write is issued at priority level 16. You can specify another priority level by editing the ord used in the Px binding (to the Bacnet virtual component). See "Bacnet virtual ord syntax" on page 3-58 for related details.

Note: *Currently, the "set" dialog for a BacnetVirtualProperty for present_value provides two fields (Figure 3-70).*

Figure 3-70 *Set dialog type field (drop-down list), value field in example binding to present_value*



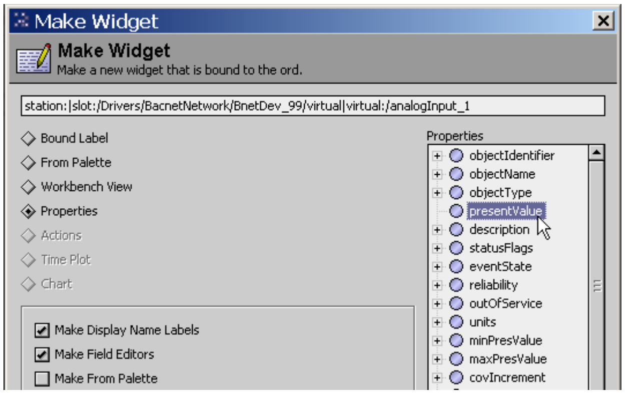
The two fields in the set action dialog apply as follows:

- Data *type* drop-down list (Real, Binary, Integer, NULL), with the applicable type (among the first three) *preselected*. The only valid change is to NULL, to clear (auto) the priority_array at that level.
- Value to write (numeric real, binary state drop-down, integer), with the current value initially shown. Currently if a binary object, the drop-down values are expressed as either “active” or “inactive”.

Px usage of Bacnet virtuals in AX-3.2

In an AX-3.2 station, you can drag a BacnetVirtualComponent in the Px view, and then select “Properties” in the popup **`Make Widget`** dialog to select which property, as shown in the example in Figure 3-71.

Figure 3-71 *Make Widget popup from dragging BacnetVirtualComponent (AX-3.2) into Px view*



Make Widget

Make Widget

Make a new widget that is bound to the ord.

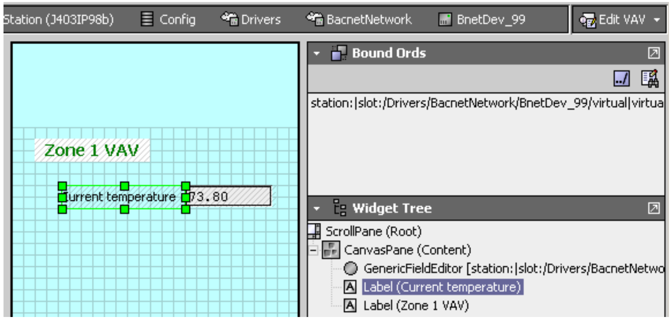
station:|slot:/Drivers/BacnetNetwork/BnetDev_99/virtual|virtual:/analogInput_1

◇ Bound Label
◇ From Palette
◇ Workbench View
◈ Properties
◇ Actions
◇ Time Plot
◇ Chart

☑ Make Display Name Labels
☑ Make Field Editors
☐ Make From Palette

Often the “presentValue” property will be selected, as shown above. The accompanying display name label (if that option was selected) can be edited to another text string, if desired, as shown in Figure 3-72.

Figure 3-72 *BacnetVirtualComponent in Px view*



station:|slot:/Drivers/BacnetNetwork/BnetDev_99/virtual|virtua

ScrollPane (Root)
CanvasPane (Content)
GenericFieldEditor [station:|slot:/Drivers/BacnetNetwo
Label (Current temperature)
Label (Zone 1 VAV)

Unlike with Bacnet virtual points in AX-3.3 and later, there is no included polling status nor any available right-click actions, such as “set”.

Bacnet virtual ord syntax

Specific to Bacnet usage, the syntax for a "virtual ord" uses a specialized form as follows:

`<ord to VirtualGateway>|virtual:/objectType_Instance[A]/propertyName[B]`

where `[A]` may be a semicolon-separated list of modifiers. Valid options include:

- priority=*X* (to indicate the priority at which the point shall be written)
- policy=*X* (to indicate the name of a tuning policy. Note that virtual points do not use COV.

and where `[B]` may be a semicolon-separated list of modifiers. Valid options include:

- status=*DOPR* (the device object property reference encoded all numerically, ignore line wrap): `objectType_instanceNumber_propertyId_[propertyArrayIndex]_[deviceObjectType_deviceInstance]`, where `[optionals in brackets]`
  - `status=0_0_85_8_10` (AI0, PresentValue in Dev10)
  - `status=0_0_111` (AI0, StatusFlags in local device)
  - `status=1_3_87_10` (AO3, PriorityArray[10] in local device)

  Properties that are BACnetStatusFlags will be merged with the bits in the status portion of the BStatusValue, other properties will add their value to the facets of the status portion.

Or in the special case of an arrayed property:

`<ord to VirtualGateway>|virtual:/objectType_Instance/propertyName/elementN`

where *N* is the property array index.

Priority example in Bacnet virtual ord

Any "set" write to the polled present_value property of a Binary Output object occurs at priority level 10:

```
station:|slot:/Drivers/BacnetNetwork/BnetDev_99/virtual|virtual:/binaryOutput_2;priority=10/presentValue
```

Tuning Policy example in Bacnet virtual ord

Polling of this Analog Input object's present_value occurs using a Bacnet tuning policy named "FastP" :

```
station:|slot:/Drivers/BacnetNetwork/BnetDev_99/virtual|virtual:/analogInput_1;policy=FastP/presentValue
```

CHAPTER 4

Niagara Bacnet Server Operation

This section describes the “server side” operation of the NiagaraAX Bacnet driver. If the host platform is licensed as a BACnet server, you can “export” any number of selected types of objects in the station (regardless of location) to appear as BACnet objects. As such, these objects can service client requests from any networked BACnet devices. Exporting of Niagara schedules and calendars is included. Histories can be exported also, where they appear externally as BACnet Trend Log objects.

Note: *For related licensing details, see “Bacnet server (export) ability” on page 1-1. Note that at the time of this document, the AX-3.6 and later* **`BACnet AWS Supervisor`** *(BacnetAwsNetwork) and* **`BACnet OWS Supervisor`** *(BacnetOwsNetwork) are not typically licensed for “server side” (export) operation. Therefore, the remainder of this section has little practical application for those two network types, unless some special license exception has been made.*

Server operation of the Bacnet driver is concurrent with client configuration and operation—often, you may choose to do both. In special cases in a station running in a JACE, you may configure it for BACnet server operation only.

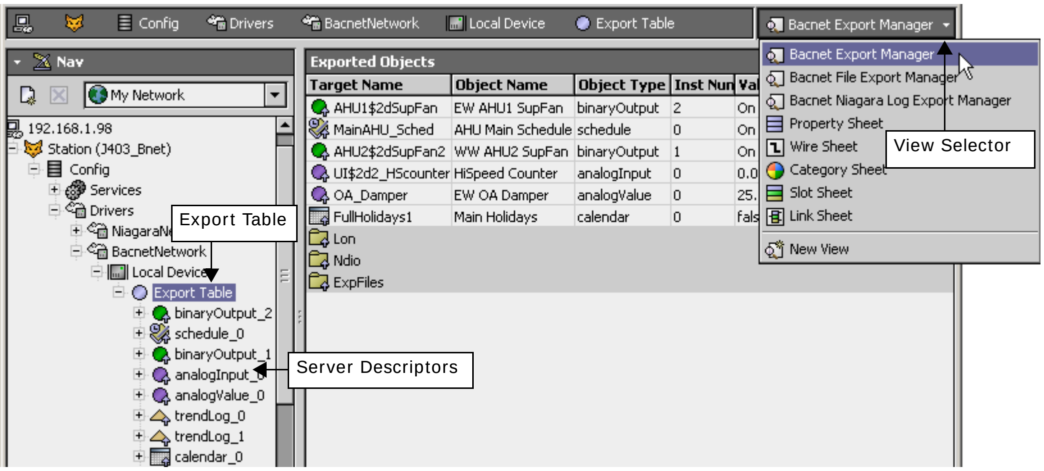
These are the main subsections:

- “Bacnet server configuration overview” on page 4-60
  - “About the Export Table” on page 4-60
  - “About server descriptors” on page 4-60
  - “Types of objects for Bacnet export” on page 4-62
  - “About BACnet server access” on page 4-62
- “Bacnet Export Manager” on page 4-63
  - “Using export folders” on page 4-64 (optional)
  - “About Discover in Bacnet Export Manager” on page 4-67
  - “Add (and Edit) in Bacnet Export Manager” on page 4-68
  - “Properties of Bacnet export descriptors” on page 4-69
  - “Bacnet Export Manager application notes” on page 4-70
- “Bacnet File Export Manager” on page 4-71
  - “About Discover in Bacnet File Export Manager” on page 4-71
  - “Add (and Edit) in Bacnet File Export Manager” on page 4-72
  - “Properties of Bacnet file descriptors” on page 4-72
- Bacnet Niagara Log Export Manager
  - “About Trend Log exports” on page 4-73
  - “About Discover in Bacnet Niagara Log Export Manager” on page 4-75
  - “Add (and Edit) in Bacnet Niagara Log Export Manager” on page 4-76
  - “Properties of Bacnet trend log descriptors” on page 4-77
- Local Device notes
  - “Local Device properties” on page 4-78
  - “Local Device container slots” on page 4-79
  - “Local Device backup and restore properties” on page 4-80

Bacnet server configuration overview

You configure almost[1] all BACnet server operation in and under the Bacnet **Local Device** in the station's BacnetNetwork. The local device's **Export Table** provides special *manager views* to centrally manage server functions, via a right-click on the Export Table node, or in its view selector (Figure 4-1).

Figure 4-1 *Export Table has 3 special views*



Export Table

Server Descriptors

View Selector

Exported Objects

Target Name	Object Name	Object Type	Inst Nun	Val
AHU1$2dSupFan	EW AHU1 SupFan	binaryOutput	2	On
MainAHU_Sched	AHU Main Schedule	schedule	0	On
AHU2$2dSupFan2	WW AHU2 SupFan	binaryOutput	1	On
UI$2d2_HScounter	HiSpeed Counter	analogInput	0	0.0
OA_Damper	EW OA Damper	analogValue	0	25.
FullHolidays1	Main Holidays	calendar	0	fals
Lon				
Ndio				
ExpFiles				

The following sections provide more details:

- About the Export Table
- About server descriptors
- Types of objects for Bacnet export
- About BACnet server access

About the Export Table

The Export Table (Figure 4-1) is a frozen slot under the BacnetNetwork **Local Device** that acts as the container for all the various Bacnet server descriptors and Bacnet export folders. Currently, it has no other configuration properties. However, the Export Table has three important views.

- Bacnet Export Manager
- Bacnet File Export Manager
- Bacnet Niagara Log Export Manager

About server descriptors

Bacnet server descriptors are the Niagara components responsible for exporting station objects as BACnet objects. Each descriptor resides under the Export Table, and corresponds to a particular component, file, or history in the station.

You add, edit, and manage server descriptors using the different manager views of the Export Table. Each manager view simplifies selection and enforces object ID instance rules. Server descriptors are automatically given Niagara names (upon creation) using an `<ObjectType>_<InstanceNumber>` convention, for example: `analogValue_1`, `trendLog_0`, `schedule_2`, and so on (Figure 4-1).

Note: *Generally, it is recommended that you do not change the Niagara name of the export descriptor. If you wish to change the name by which the object is known to BACnet clients, do this in the objectName property, which is initialized with the name of the exposed component.*

There are three categories of Bacnet server descriptors:

- export descriptor (for components, reflecting the Niagara station "object space")
- file descriptor (for files)
- log descriptor (for histories)

In addition, as needed, you can add Bacnet export folders to organize Bacnet server descriptors.

[1] In some cases if operating as a BACnet server, a few configuration changes under the BacnetNetwork's BacnetComm, Server component may be needed. See "Bacnet Comm: Server configuration" on page 3-17.

About export descriptors

Bacnet export descriptors include 15 different types to export different Niagara *components*, including Boolean, Enum, and Numeric points (read-only and writable), and various Schedule types. You add them using the Bacnet Export Manager view.

Externally, to another BACnet device, the exported BACnet object appears with properties that source from two different areas of the station. For example, each exported component (point or schedule) exports as a BACnet object with properties from the component itself, plus additional properties in its export descriptor.

Note: *You do not see export (component) descriptors in the other two export manager views (Bacnet File Export Manager, Bacnet Niagara Log Export Manager). Those views show other "non-component" server descriptors.*

The different types of export descriptors include:

- BacnetAnalogInputDescriptor
- BacnetAnalogOutputDescriptor
- BacnetAnalogValueDescriptor
- BacnetAnalogValuePrioritizedDescriptor
- BacnetBinaryInputDescriptor
- BacnetBinaryOutputDescriptor
- BacnetBinaryValueDescriptor
- BacnetBinaryValuePrioritizedDescriptor
- BacnetBooleanScheduleDescriptor
- BacnetCalendarDescriptor
- BacnetEnumScheduleDescriptor
- BacnetLoopDescriptor
- BacnetMultiStateInputDescriptor
- BacnetMultiStateOutputDescriptor
- BacnetMultiStateValueDescriptor
- BacnetMultiStateValuePrioritizedDescriptor

For additional details, see "About Discover in Bacnet Export Manager" on page 4-67, "Add (and Edit) in Bacnet Export Manager" on page 4-68 and "Properties of Bacnet export descriptors" on page 4-69.

About file descriptors

A Bacnet file descriptor exports a file under the station directory as a BACnet File object. You add file descriptors using the Bacnet File Export Manager view of the Export Table.

Note: *You do not see file descriptors in the other two export manager views (Bacnet Export Manager, Bacnet Niagara Log Export Manager). Those views show other server descriptors.*

There is only one type: BacnetFileDescriptor. For more details, see "About Discover in Bacnet File Export Manager" on page 4-71 and "Properties of Bacnet file descriptors" on page 4-72.

About log descriptors

A Bacnet log descriptor exports a Niagara history as a BACnet Trend Log object. There are two different types of log descriptors:

- BacnetNiagaraHistoryDescriptor — Created when you select and add a "standard" Niagara history using the Bacnet Niagara Log Export Manager view of the Export Table. Note that the resulting exported Trend Log object is compliant only with the original "broken" specification for BACnet Trend Logs, which was superseded and fixed in Addendum B to the 2001 version of the spec,
- BacnetTrendLogDescriptor — Automatically created when you add a specialized BacnetTrendLogExt extension to a point. The resulting exported Trend Log object is fully-BACnet compliant with the 2004 version of the BACnet spec.

For more details, see "About Trend Log exports" on page 4-73.

Note: *You do not see log descriptors in the other two export manager views (Bacnet Export Manager, Bacnet File Export Manager). Those views show other server descriptors.*

About export folders

Use Bacnet export folders to organize any collection of Bacnet server descriptors. Add export folders using the `New Folder` button in any of the Export Table manager views.

There is only *one type* of Bacnet export folder—you see export folders in all views (Bacnet Export Manager, Bacnet File Export Manager, Bacnet Niagara Log Export Manager). Note that the *default* (double-click) *view* for any export folder is the Bacnet Export Manager.

Note: *Starting in AX-3.5, Bacnet export folders are exposed as BACnet* `Structured View Objects` *(SVOs). Additional folder properties and a related new view on export folders are now available. For more details, see the section "Using export folders" on page 4-64.*

Types of objects for Bacnet export

Table 4-1 lists the types of objects in the station (components, files, histories) that you can export as BACnet objects. Export any *component* using the Export Table's Bacnet Export Manager view. To export files and histories, you use the Bacnet File Export Manager and Bacnet Niagara Log Export Manager views, respectively.

Table 4-1 *Niagara objects for export as BACnet objects*

Station component, file, or history type	BACnet object choices	Notes
Alarm Class	Notification Class	See "Sending Alarms to BACnet" on page 5-85.
BooleanPoint, BooleanWritable, various kitControl components (i.e. Logic)	Binary Input	Proxy points under any driver are supported.
	Binary Value	
BooleanWritable (additional choices)	Binary Output	
	Binary Value Prioritized	
EnumPoint, EnumWritable	Multi State Input	Proxy points under any driver are supported. Note that the Facets, range, ordinal (integer) in the exported EnumPoint or EnumWritable must begin with 1, and not 0. The range must also be contiguous up to the maximum value (gaps in the range are not allowed).Otherwise, the object does not export to BACnet.
	Multi State Value	
EnumWritable (additional choices)	Multi State Output	
	Multi State Value Prioritized	
NumericPoint, NumericWritable, various kitControl components (i.e. Math)	Analog Input	Proxy points under any driver are supported.
	Analog Value	
NumericWritables (additional choices)	Analog Output	
	Analog Value Prioritized	
LoopPoint	Loop	Loop object export is default.
	Analog Input	
	Analog Value	
BooleanSchedule, EnumSchedule, NumericSchedule, StringSchedule	Schedule	In addition, you can selectively export the configuration of a standard Niagara schedule or calendar into an existing Schedule object or Calendar object in a client BACnet device, using the Bacnet Schedule Export Manager view of that BacnetDevice's Schedules extension.
CalendarSchedule	Calendar	
<*file*> (any)	File	Files under the station folder can be exported.
<*history*> (any)	Trend Log	Two different methods exist for exporting Niagara histories. See "About Trend Log exports" on page 4-73.

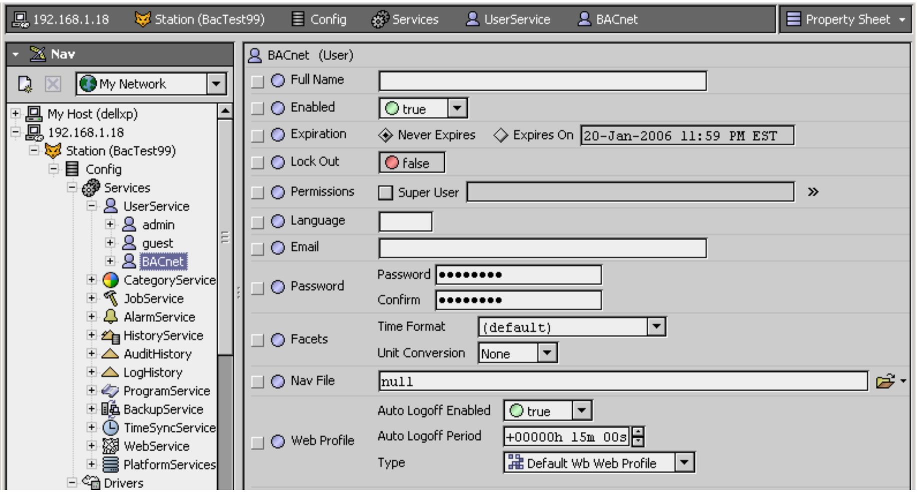
About BACnet server access

By default, the Bacnet driver provides external BACnet (client) devices "read access" to *all* exposed (exported to BACnet) objects in the station. Access depends on a station user named "`BACnet`".

If this user does not already exist in the station, the Bacnet driver *automatically creates it*, upon startup. The `BACnet` user is intially created without any permissions, as shown in Figure 4-1.

If you want BACnet clients to have *write access* to objects, you must assign this `BACnet` user the necessary permissions. See "Allowing write access from BACnet" on page 4-63.

Figure 4-2 *BACnet station user automatically created, but without write permissions*



BACnet (User)

Full Name	
Enabled	true
Expiration	◈ Never Expires ◇ Expires On 20-Jan-2006 11:59 PM EST
Lock Out	false
Permissions	□ Super User
Language	
Email	
Password	Password •••••••• Confirm ••••••••
Facets	Time Format (default) Unit Conversion None
Nav File	null
Web Profile	Auto Logoff Enabled true Auto Logoff Period +00000h 15m 00s Type Default Wb Web Profile

Allowing write access from BACnet

Assigning "read permissions" to the station user BACnet is not necessary—BACnet server access is automatic to external BACnet client requests. However, to allow any external writes (from BACnet) to properties of exported components, including invoking commands (actions), you must assign the BACnet user the necessary permissions to those components.

For example, to allow an invoked "Active" action from BACnet to an exported BooleanWritable, in addition to making it "BACnet Writable" at priority level 8 when exporting (see "Add (and Edit) in Bacnet Export Manager" on page 4-68), you must configure the station's BACnet user to have *operator write* permissions on that BooleanWritable, at a minimum. Or, if an exported NumericWritable has an alarm extension, and you want to permit external BACnet writes to its "alarm limit" values, configure the BACnet user to have *admin write* permissions on the exported NumericWritable.

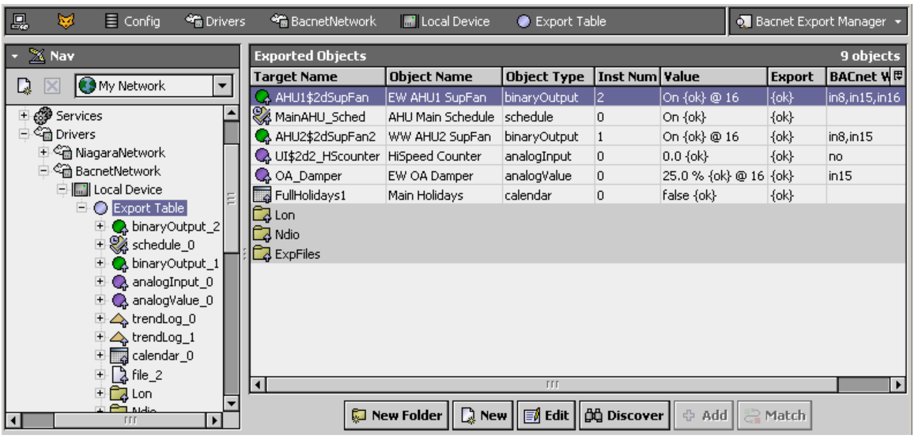
In either example, to allow an external BACnet write to a property like "Out Of Service" or "Notify Type," you must give the BACnet user admin write permissions on the Bacnet export descriptors. For details about station user security, see "About Security" in the *User Guide*.

Note: BACnet *user permissions also apply to writes of any exported files and histories. Also, note that while a password for the* BACnet *user is technically not needed (for external BACnet access), you should assign one anyway, because of the write permissions typically assigned. Make it a "non-blank" password, and guard this password carefully!*

Bacnet Export Manager

The Bacnet Export Manager (Figure 4-3) is the default view for the **Export Table**, as well as any child **Bacnet Export Folder**, and provides features like many other manager views.

Figure 4-3 *Bacnet Export Manager view*



Exported Objects 9 objects

Target Name	Object Name	Object Type	Inst Num	Value	Export	BACnet W
AHU1$2dSupFan	EW AHU1 SupFan	binaryOutput	2	On {ok} @ 16	{ok}	in8,in15,in16
MainAHU_Sched	AHU Main Schedule	schedule	0	On {ok}	{ok}	
AHU2$2dSupFan2	WW AHU2 SupFan	binaryOutput	1	On {ok} @ 16	{ok}	in8,in15
UI$2d2_HScounter	HiSpeed Counter	analogInput	0	0.0 {ok}	{ok}	no
OA_Damper	EW OA Damper	analogValue	0	25.0 % {ok} @ 16	{ok}	in15
FullHolidays1	Main Holidays	calendar	0	false {ok}	{ok}	
Lon						
Ndio						
ExpFiles						

This view shows only Bacnet export descriptors and any Bacnet export folders. The following sections provide more details:

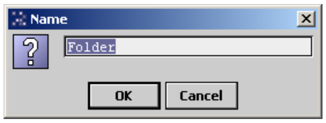
- Using export folders (optional)
- About Discover in Bacnet Export Manager
- Add (and Edit) in Bacnet Export Manager
- Properties of Bacnet export descriptors
- Bacnet Export Manager application notes

Using export folders

Use Bacnet export folders to organize any collection of Bacnet server descriptors. Add export folders using the **New Folder** button in any of the Export Table manager views.

Prior to AX-3.5, when adding a **New Folder** in any manager view of the Export Table, a simple **Name** dialog appears, as shown in Figure 4-4.

Figure 4-4 *New folder dialog prior to AX-3.5*



Name

Folder

In this case you simply enter the Niagara (station) name for the Bacnet export folder.

Note: *Starting in AX-3.5, Bacnet export folders are exposed as BACnet* **Structured View Objects** *(SVOs). When adding an export folder, the* **New** *dialog has many more fields. For more details, see the next section "Structured View Object server support".*

Structured View Object server support

Starting in AX-3.5, Bacnet export folders made are exposed as BACnet **Structured View Objects** (SVOs). Server descriptors that are children of the export folder are automatically included as entries in the SVO's Subordinate_List property. On any export folder, there is also now an **Svo Subordinate Manager** view which you can use to manually configure additional subordinates, if needed.

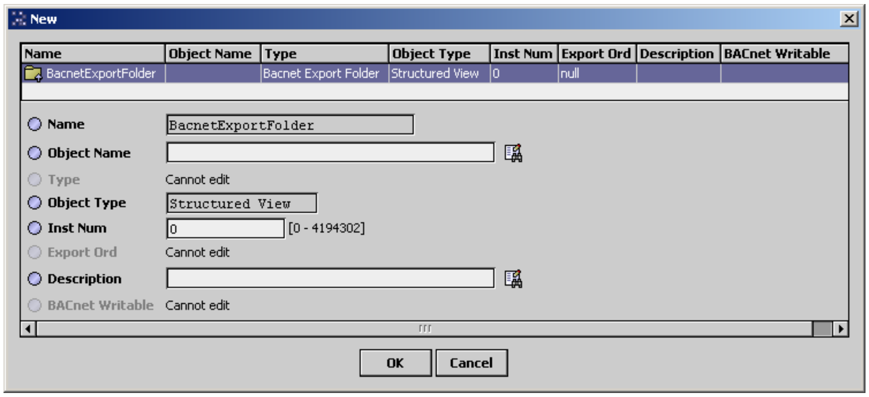
BACnet SVOs allow an organizational interface, where each holds references to subordinate objects. These objects may include other SVOs, thus allowing multi-level hierarchies. The intent is to convey an organization beyond what is available in the (otherwise) "flat" BACnet object list. An SVO organization could be network (physical), logical, geographical, or some other structure.

See the following sections for more details:

- "Adding new Bacnet export folders"
- "Svo Subordinate Manager view"

Adding new Bacnet export folders When adding new export folders in AX-3.5 and later from the Bacnet Export Manager view, the **New** dialog has several fields, as shown in Figure 4-5.

Figure 4-5 *New export folder dialog (AX-3.5 and later)*



New

Name	Object Name	Type	Object Type	Inst Num	Export Ord	Description	BACnet Writable
BacnetExportFolder		Bacnet Export Folder	Structured View	0	null		

○ Name	BacnetExportFolder
○ Object Name	
○ Type	Cannot edit
○ Object Type	Structured View
○ Inst Num	0 [0 - 4194302]
○ Export Ord	Cannot edit
○ Description	
○ BACnet Writable	Cannot edit

Important fields in the New export folder dialog are as follows:

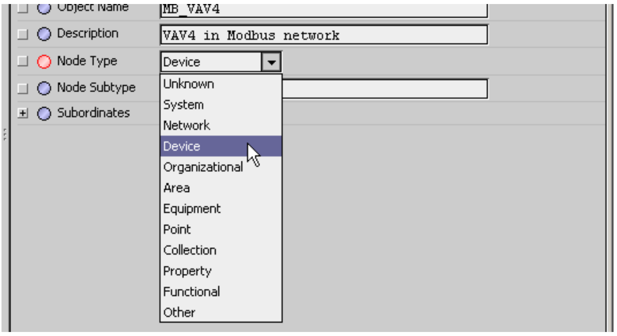
- **Name**
  NiagaraAX name of the export folder component. Currently this is read-only, and initially reflects the default "`BacnetExportFolder`" (or "`BacnetExportFolder`*N*") name.

Note: *After adding an export folder, you can rename it (click to select it, then Ctrl + R for the rename dialog). For clarity, rename to match its* **`Object Name`** *(BACnet) property, described next.*

- **Object Name**
  Enter the BACnet name for the Structured View Object (SVO) this export folder will represent. This name must be unique at this level in the structured view hierarchy.
- **Object Type**
  BACnet Object Type (read-only) is **`Structured View`**.
- **Inst Num**
  The numerical BACnet instance number for this Structured View Object (SVO), with a range from 0 to 4194302. Must be unique among all SVOs in the station.
  Note: *Workbench automatically increments this value when adding multiple Bacnet export folders.*
- **Description**
  Enter the BACnet Description property for this Structured View Object.

After adding an export folder, in addition to renaming it from "`BacnetExportFolderN`", you may wish to edit one or two other properties from its *property sheet*, as shown in Figure 4-6.

Figure 4-6 *Node Type and Node Subtype properties in Bacnet export folder property sheet*



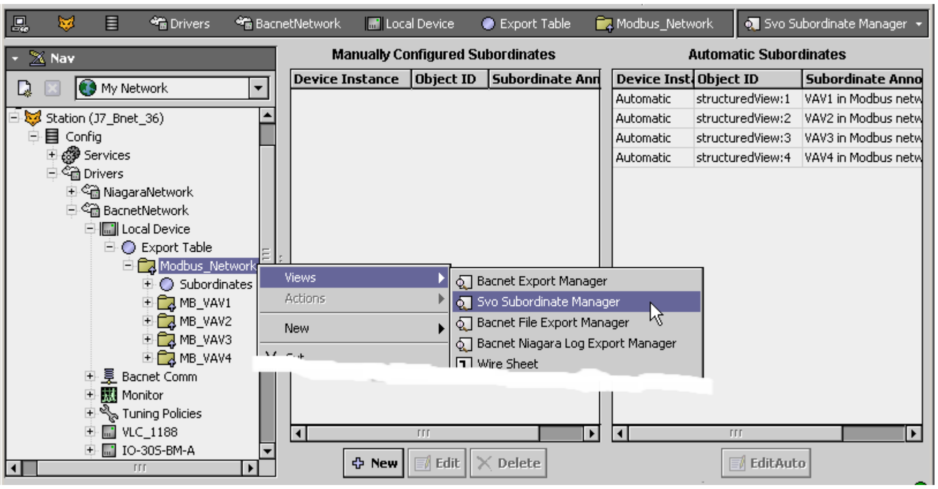
Object Name: MB_VAV4
Description: VAV4 in Modbus network
Node Type: Device
Node Subtype:
Subordinates

These two properties on the BacnetExportFolder's property sheet are:

- **Node Type**
  The BACnet Node_Type property is an enumerated value to generally classify the object. A drop-down list allows selection as either `unknown` (the default), or else `system`, `network`, `device`, `organizational`, `area`, `equipment`, `point`, `collection`, `property`, `functional`, or `other`.
- **Node Subtype**
  The BACnet Node_Subtype property allows a more specific classification, using any text descriptor (default is blank).

Svo Subordinate Manager view Starting in AX-3.5, each BacnetExportFolder has an available **`Svo Subordinate Manager`** view, as shown in Figure 4-7.

Figure 4-7 *Svo Subordinate Manager view on BacnetExportFolder (AX-3.5 and later)*



Manually Configured Subordinates

Device Instance	Object ID	Subordinate Ann

Automatic Subordinates

Device Inst	Object ID	Subordinate Anno
Automatic	structuredView:1	VAV1 in Modbus netw
Automatic	structuredView:2	VAV2 in Modbus netw
Automatic	structuredView:3	VAV3 in Modbus netw
Automatic	structuredView:4	VAV4 in Modbus netw

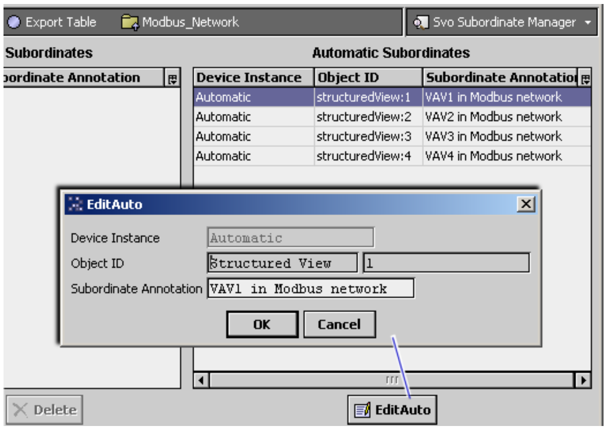
This view is to manage references to *subordinate* BACnet objects. In the exposed Structured View Object (from a Bacnet export folder), note that subordinates appear in two property lists:

- Subordinate_List: List of object identifiers for all BACnet objects the SVO organizes.
- Subordinate_Annotations: List of descriptions of the same objects—the same size as the Subordinate_List, where entries in one correspond to entries in the other.

Typically, most entries are "automatic" subordinates, shown on the *right side*. These result from either child Bacnet server descriptors or export folders.

You can select one, and click **Edit Auto** for an **EditAuto** dialog.

Figure 4-8 *EditAuto dialog for selected "Automatic" subordinate object*



Device Instance	Object ID	Subordinate Annotation
Automatic	structuredView:1	VAV1 in Modbus network
Automatic	structuredView:2	VAV2 in Modbus network
Automatic	structuredView:3	VAV3 in Modbus network
Automatic	structuredView:4	VAV4 in Modbus network

EditAuto

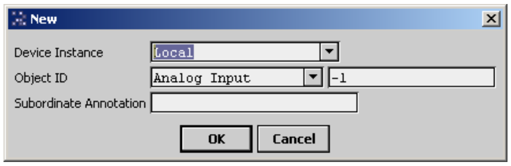
Device Instance: Automatic
Object ID: Structured View 1
Subordinate Annotation: VAV1 in Modbus network

OK Cancel

As shown above in Figure 4-8, you can edit the "Subordinate Annotation" text—note this is actually the Description property in the referenced Bacnet server descriptor or export folder.

The *left side* of the Svo Subordinate Manager allows you to *manually* add (or edit or delete) new subordinate references, using the buttons below that view half. When you click **New**, a **New** dialog appears.

Figure 4-9 *New dialog for manually-added subordinate object*



New

Device Instance: Local
Object ID: Analog Input -1
Subordinate Annotation:

OK Cancel

As shown in Figure 4-9, there are three fields available:

- **Device Instance**
  Either accept the default "Local" (for the BACnet device represented by this Niagara station), or type in the numerical Device ID for another remote BACnet device.
  Note: *BACnet allows a Structured View Object to reference subordinate objects in other BACnet devices. If you elect to use this feature, this is how it is done.*
- **Object Id**
  Using the drop-down control, select the BACnet object type (on the left side), and on the right-side enter a valid instance number (from 0 on up), overwriting the default "-1" instance number.
- **Subordinate Annotation**
  Enter a text string for the Description property of this BACnet object.

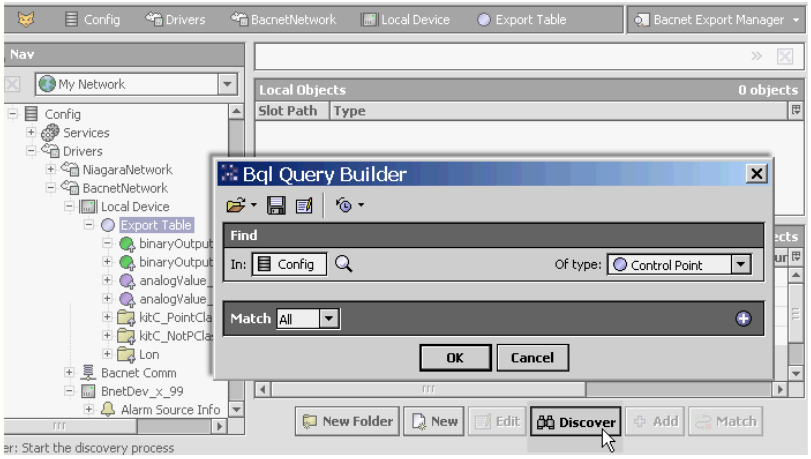
Other buttons below the left side of the Svo Subordinate Manager view allow you to **Edit** or **Delete** manually-added entries for subordinate objects.

Note: *The same BACnet object can be a subordinate in more than one Structured View Object. For example, an Analog Input object that represents a temperature can be a subordinate in both a "Temperatures" SVO tree as well as a "Admin Building" SVO tree.*

About Discover in Bacnet Export Manager

You typically use **`Discover`** in the Bacnet Export Manager, vs. **`New`** and/or **`Match`**. When you click **`Discover`**, the **`Bql Query Builder`** dialog appears, as shown in Figure 4-10.

Figure 4-10 *Discover yields Bql Query Builder*



Bql Query Builder

Find

In: Config

Of type: Control Point

Match All

See “About the Bql Query Builder” in the *Drivers Guide* for general information. Relative to usage in the **`Bacnet Export Manager`**, this dialog lets you find and select *components* in the station to export (expose) as BACnet objects.

You can export these component types (Also see “Types of objects for Bacnet export” on page 4-62):

- Any components sub-classed from BooleanPoint, EnumPoint or NumericPoint, (including Writables) meaning the following types:
  - Any proxy point that is not a StringPoint or StringWritable, under any driver.
  - Most kitControl components that have a null ProxyExt—this includes all Math and Logic components, for example. Typically, you export these as a “value type” object (Analog Value, Binary Value, Multistate Value). Note that you can export a LoopPoint as a BACnet Loop object. Again, kitControl objects subclassed from StringPoint (string Output) cannot be exported.
- Any Niagara Schedule component *except* a TriggerSchedule (no BACnet object equivalent).
- Alarm Class components (under the station’s AlarmService), which export as BACnet Notification Class objects.

Note: *Therefore in the* **`Bql Query Builder`***, type selections other than Control Point, Boolean Point, Enum Point, Numeric Point, Schedule, and Alarm Class have no practical application.*

As with other manager views featuring online discovery, “Learn Mode” in Bacnet Export Manager has two panes:

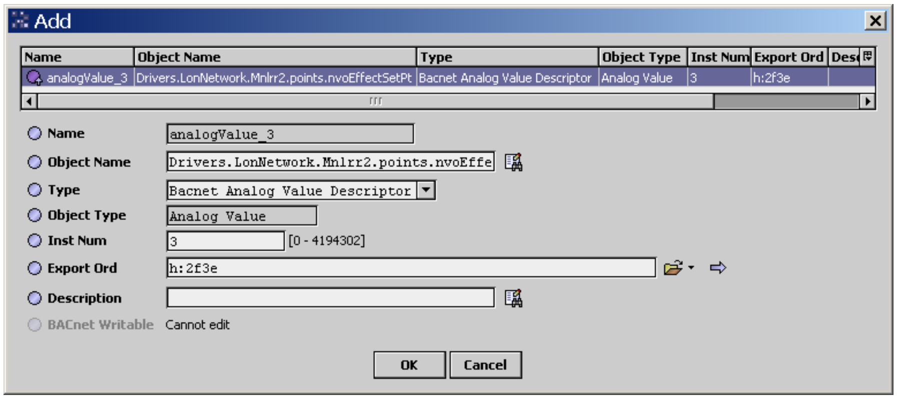
- Top (discovered) **`Local Objects`** pane—listing components found from your last Bql query.
- Bottom (database) **`Export Objects`** pane—listing components currently exported to BACnet. These are special “export descriptor” components—each essentially a pointer to the exposed Niagara component, with additional slots that determine how BACnet access/writes are handled. See the next section, “Add (and Edit) in Bacnet Export Manager” for more details.

When exporting, the manager automatically performs “object ID maintenance” on exported components, ensuring that no duplicate combinations of object *type* and *instance number* are created.

Add (and Edit) in Bacnet Export Manager

When you click **Add** with a component highlighted in the Bacnet Export Manager, the **Add** dialog (Figure 4-11) contains a number of fields with either default or empty values, described below.

Figure 4-11 *Add dialog in Bacnet Export Manager*



Add

Name	Object Name	Type	Object Type	Inst Num	Export Ord	Desc
analogValue_3	Drivers.LonNetwork.Mnlrr2.points.nvoEffectSetPt	Bacnet Analog Value Descriptor	Analog Value	3	h:2f3e	

- Name: analogValue_3
- Object Name: Drivers.LonNetwork.Mnlrr2.points.nvoEffe
- Type: Bacnet Analog Value Descriptor
- Object Type: Analog Value
- Inst Num: 3 [0 - 4194302]
- Export Ord: h:2f3e
- Description:
- BACnet Writable: Cannot edit

All fields in this dialog apply separately to each highlighted (for export) component, as follows:

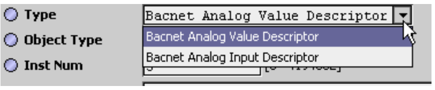
Note: *Select* `Type` *first, before editing other fields such as Object Name and Description. Otherwise, those entries become cleared and you will need to re-enter.*

In addition, Type is the only slot you cannot change after adding (say, in the **Edit** *dialog).*

- Name
  Niagara read-only component name for the Bacnet export descriptor, which defaults to the combination of <*object type*>_<*instance number*>. Reflects the "Type" chosen (below it in dialog).
- Object Name
  The "exposed to BACnet" name for this object. By default, the *entire component path* under the station's Config is included, using period (".") delimiters between *parent.child* levels. This enforces (externally) the BACnet requirement for unique names for all objects in a device.
  Note: *You can shorten or edit object name, either now or later. However, please note that each Object Name should be unique among all server descriptors under the Export Table.*

  The Batch Search and Replace feature is useful when adding/editing multiple exports.
- Type
  Initially (in **Add** dialog) you can select the specific Bacnet export descriptor (*object type*)—in most cases, from two or more choices. Click the drop down control (Figure 4-12) to select type.

Figure 4-12 *Bacnet export descriptor (object) type choice is typically two or more*



- Type: Bacnet Analog Value Descriptor
- Object Type
- Inst Num

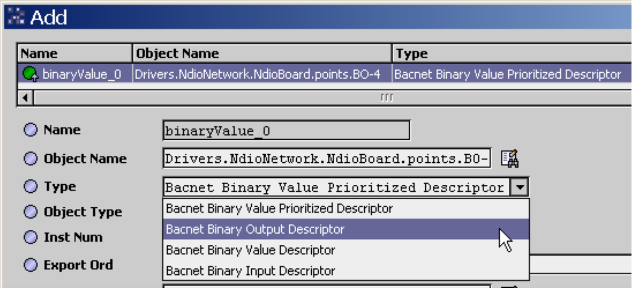
Bacnet Analog Value Descriptor
Bacnet Analog Input Descriptor

Writable Niagara components typically offer more type selections than read-only points, such as:

- Two read-only: "input" or "value" type, e.g. Binary Input or Binary Value object.
- Two writable: "output" or "prioritized value" type, e.g. Binary Output or Binary Value Prioritized object.

Figure 4-13 shows an example of type selections when exporting an NdioBooleanWritable point.

Figure 4-13 *Writable points provide more export type selections*



Add

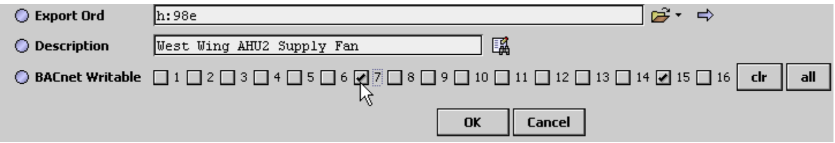
Name	Object Name	Type
binaryValue_0	Drivers.NdioNetwork.NdioBoard.points.BO-4	Bacnet Binary Value Prioritized Descriptor

- Name: binaryValue_0
- Object Name: Drivers.NdioNetwork.NdioBoard.points.BO-
- Type: Bacnet Binary Value Prioritized Descriptor
- Object Type
- Inst Num
- Export Ord

Bacnet Binary Value Prioritized Descriptor
Bacnet Binary Output Descriptor
Bacnet Binary Value Descriptor
Bacnet Binary Input Descriptor

- Object Type
  Reflects the "BACnet Object type" exposed, dependent on Type selection.
- Inst Num
  The instance number portion of this object's "Object ID," which must be unique within the station for this (exported) BACnet Object type. By default, the export manager enforces this.
  Note: *If you are exporting a group of points at once, you can select them all in the Add/Edit dialog, and type the instance number for the first one, and each descriptor will be assigned successive instance numbers as available.*
- Export Ord
  Station's Ord location of the source component. By default, format used is the numeric "handle" instead of slot (better if source object gets renamed). In an **Edit** (dialog) scenario, you can access a more meaningful ord by clicking the right-side Folder control. This produces a popup **Select Ord** dialog, showing the component's location in the station's component tree hierarchy.
- Description
  Optional text string; this appears as the Description property value in the exposed BACnet object.
- BACnet Writable
  ("Dimmed" if the export Type is read-only, for example Analog Input or Binary Value)
  For writable types, an array of checkboxes lets you select the *specific* priority levels externally exposed to BACnet to accept writes. Included are "all controls" to clear or select, see Figure 4-14. (For "full BACnet compliance," all levels must be selected—however, see the Note: below).

Figure 4-14 *BACnet Writable settings for exposed writable point includes all priority levels*



○ Export Ord: h:98e
○ Description: West Wing AHU2 Supply Fan
○ BACnet Writable: □ 1 □ 2 □ 3 □ 4 □ 5 □ 6 ☑ 7 □ 8 □ 9 □ 10 □ 11 □ 12 □ 13 □ 14 ☑ 15 □ 16 clr all

Note: *Each priority level (1—16) that you enable for BACnet writes results in that "InN" input on the source component to be linked to this "Bacnet export descriptor" component. These links appear as "nubs" when viewing the source writable point in its wire sheet view.*

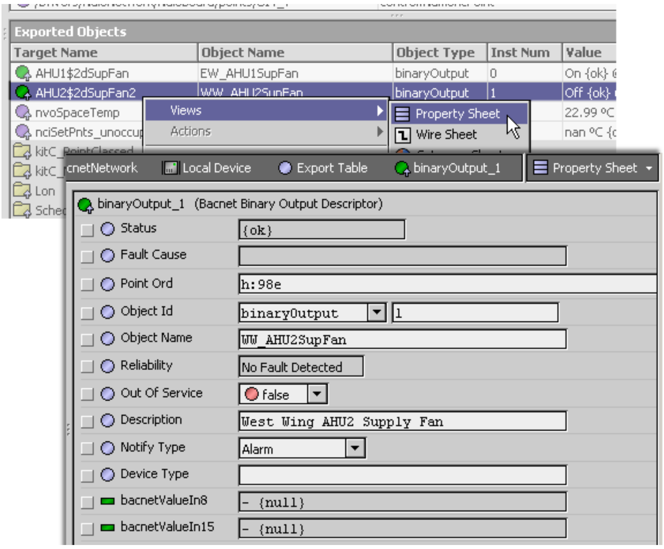
Do not select any priority level already linked or in use by Niagara, otherwise control contention will occur. Also see "Allowing write access from BACnet" on page 4-63.

Note: *After creation (adding) Bacnet export descriptors, note that each descriptor also has additional properties accessible in its property sheet, along with those seen in the* **Add** *and* **Edit** *dialog. See the next section, "Properties of Bacnet export descriptors" for more details.*

Properties of Bacnet export descriptors

Each Bacnet export descriptor has various properties that affect its BACnet representation. Access these properties in the property sheet view of any export descriptor, as shown in Figure 4-15.

Figure 4-15 *Additional properties of Bacnet export descriptors on property sheet*



Exported Objects

Target Name	Object Name	Object Type	Inst Num	Value
AHU1$2dSupFan	EW_AHU1SupFan	binaryOutput	0	On {ok}
AHU2$2dSupFan2	WW_AHU2SupFan	binaryOutput	1	Off {ok}
nvoSpaceTemp				22.99 °C
nciSetPnts_unoccup				nan °C {c

binaryOutput_1 (Bacnet Binary Output Descriptor)

Status	{ok}
Fault Cause	
Point Ord	h:98e
Object Id	binaryOutput 1
Object Name	WW_AHU2SupFan
Reliability	No Fault Detected
Out Of Service	false
Description	West Wing AHU2 Supply Fan
Notify Type	Alarm
Device Type	
bacnetValueIn8	- {null}
bacnetValueIn15	- {null}

Some of these properties are BACnet optional properties, such as Notify_Type and Device Type. Others are required, such as Out_Of_Service. The following list describes additional properties in Bacnet export descriptors, not seen in the **Add** or **Edit** dialog for the component:

- Fault Cause
  Niagara-only, read-only property showing an explanatory text string if the export descriptor is in fault. For example, if its Object Id is manually edited to be the same as another existing export descriptor—a fault occurs and the Fault Cause would be "Duplicate Object ID."
- Reliability
  BACnet read-only property that reads "No Fault Detected" when things are OK, and typically "Unreliable Other" if the exported BACnet object appears in fault. This might happen, for example, if the source (exported) component is a proxy point under some other driver, and device communications are down, or if the point has been disabled.
  In this case, note that the Bacnet *export descriptor* retains a status of "ok."
- Out Of Service
  Writable BACnet property, which you can set to false if needed—this affects the BACnet exposure and access of the source component only. In other words, if the source (exported) component is a proxy point, the equivalent "Enabled" property in its ProxyExt is *not* affected.
- Notify Type
  Writable BACnet property that can be set to either Alarm (default) or Event. Applies if the source Niagara component has an alarm extension.
- Cov Increment
  Writable BACnet property included only for analog object types (Analog Input, Analog Output, Analog Value, Analog Value Priority, Loop), specifying the minimum COV required before a COVNotification is issued to subscriber BACnet COV-clients. Default value is typically 1.00.
- Device Type
  Writable BACnet property included only for some object types (Analog Input, Analog Output, Binary Input, Binary Output, Multi-state Input, Multi-state Output), typically associated with the type of physical input or output. Use is optional. In NiagaraAX, this might apply mostly to exported Ndio points—where each point corresponds to a physical device.

Bacnet Export Manager application notes

If you are exporting large numbers of components to BACnet, it is recommended that you use the **New Folder** button in the Export Table's manager view to make *multiple* Bacnet export folder containers, for logical organization of export descriptors. For example, you may wish to make such one or more folders for a certain driver network, and another for Niagara schedules. Each export folder provides the same set of export manager views as the Export Table.

Note that starting in AX-3.5, export folders are exposed to BACnet as **Structured View Objects**, allowing a hierarchical organization of BACnet objects. For related details, see "Structured View Object server support" on page 4-64.

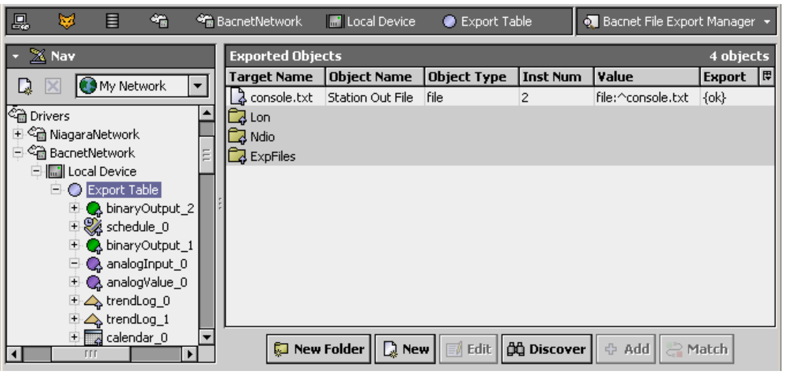
Note: *Before deleting any source (exported) component in the station, first delete its corresponding Bacnet export descriptor, using the Bacnet Export Manager. Otherwise, the export descriptor may remain "orphaned" in the Export Table, showing "*Invalid Ord!*" in the Target Name and Value columns. If this occurs, you can delete the orphaned export descriptors manually.*

Any exported points or components that contain an *alarm extension* are automatically exposed to BACnet with properties related to alarming available. For example, if you exported a NumericPoint with an OutOfRangeAlarmExt, its exposed BACnet object (say, Analog Value) will have properties "High Limit," "Low Limit," "Deadband," and so on.

Bacnet File Export Manager

Use the Bacnet File Export Manager (Figure 4-16) to export files under the station's folder as BACnet File objects. This view shows only Bacnet file descriptors and any Bacnet export folders.

Figure 4-16 *Bacnet File Export manager view*



Exported Objects 4 objects

Target Name	Object Name	Object Type	Inst Num	Value	Export
console.txt	Station Out File	file	2	file:^console.txt	{ok}
Lon					
Ndio					
ExpFiles					

Files can be accessed as read-only or as writable, depending on the local file system. Currently, the BACnet "File_Access_Method" is Stream Access only.

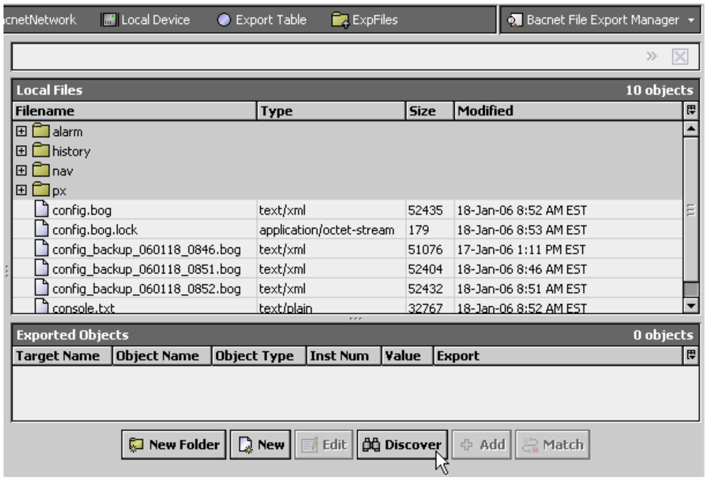
The following sections provide more details:

- About Discover in Bacnet File Export Manager
- Add (and Edit) in Bacnet File Export Manager
- Properties of Bacnet file descriptors

About Discover in Bacnet File Export Manager

You typically use **`Discover`** in the Bacnet File Export Manager, vs. **`New`** and/or **`Match`**. When you click **`Discover`**, the top pane shows files under the station's folder, as shown in Figure 4-17.

Figure 4-17 *Discover in Bacnet File Export Manager*



Local Files 10 objects

Filename	Type	Size	Modified
alarm			
history			
nav			
px			
config.bog	text/xml	52435	18-Jan-06 8:52 AM EST
config.bog.lock	application/octet-stream	179	18-Jan-06 8:53 AM EST
config_backup_060118_0846.bog	text/xml	51076	17-Jan-06 1:11 PM EST
config_backup_060118_0851.bog	text/xml	52404	18-Jan-06 8:46 AM EST
config_backup_060118_0852.bog	text/xml	52432	18-Jan-06 8:51 AM EST
console.txt	text/plain	32767	18-Jan-06 8:52 AM EST

Exported Objects 0 objects

Target Name	Object Name	Object Type	Inst Num	Value	Export

As with other manager views featuring online discovery, "Learn Mode" in Bacnet File Export Manager has two panes:

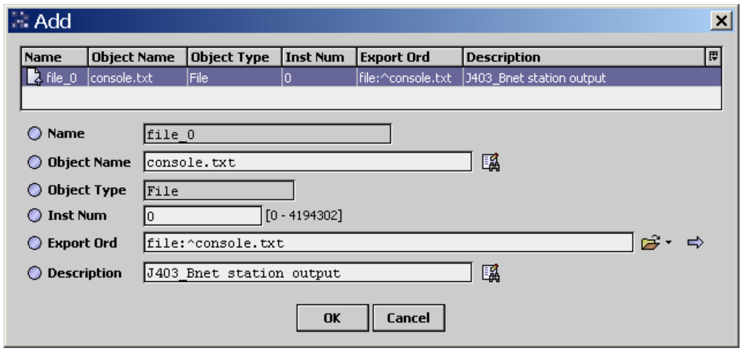
- Top (discovered) **`Local Files`** pane—listing all files and subfolders under the station folder.
- Bottom (database) **`Exported Objects`** pane—listing files currently exported to BACnet. These are special "file descriptor" components—each essentially a pointer to the exposed station file, with additional slots that determine how BACnet access/writes are handled. See the next section, "Add (and Edit) in Bacnet File Export Manager" for more details.

When exporting, the manager automatically performs "object ID maintenance" on exported files, ensuring that no duplicate combinations of object *type* and *instance number* are created.

Add (and Edit) in Bacnet File Export Manager

When you click **Add** with a file highlighted to export, the **Add** dialog (Figure 4-18) contains a number of fields with either default or empty values, described below.

Figure 4-18 Add dialog in Bacnet File Export Manager



Name	Object Name	Object Type	Inst Num	Export Ord	Description
file_0	console.txt	File	0	file:^console.txt	J403_Bnet station output

○ Name	file_0
○ Object Name	console.txt
○ Object Type	File
○ Inst Num	0 [0 - 4194302]
○ Export Ord	file:^console.txt
○ Description	J403_Bnet station output

All fields in this dialog apply separately to each highlighted (for export) file, as follows:

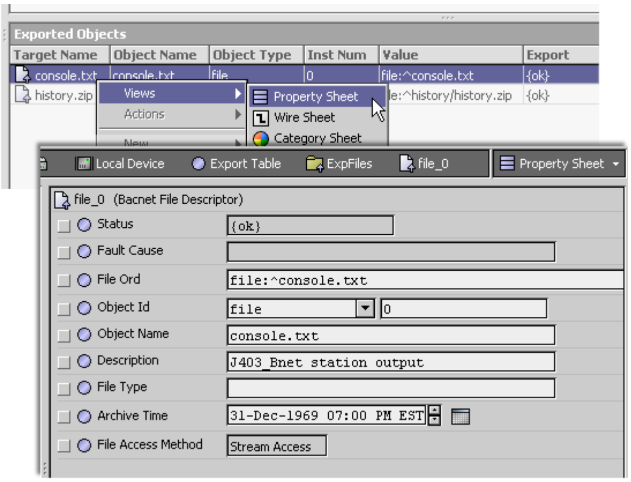
- Name
  Read-only Niagara component name for the Bacnet file descriptor, which defaults to the combination of: `file_<instance number>`.
- Object Name
  The "exposed to BACnet" name for this file. By default, only the filename and extension are used.
- Object Type
  Read-only reflection of the "BACnet Object type" exposed, in this case always: `File`.
- Inst Num
  The instance number portion of this object's "Object ID," which must be unique within the station for this (exported) BACnet File object type. By default, the export manager enforces this.
- Export Ord
  Station's Ord location of the source file, using standard file Ord notation.
- Description
  Optional text string; this appears as the Description property value in the exposed BACnet object.

Note: *After creation (adding) Bacnet file descriptors, note that each descriptor also has additional properties accessible in its property sheet, along with those seen in the* **Add** *and* **Edit** *dialog. See the next section, "Properties of Bacnet file descriptors" for more details.*

Properties of Bacnet file descriptors

Each Bacnet file descriptor has various properties that affect its BACnet representation. Access these properties in the property sheet view of any export descriptor, as shown in Figure 4-19.

Figure 4-19 Additional properties of Bacnet file descriptors on property sheet



Target Name	Object Name	Object Type	Inst Num	Value	Export
console.txt	console.txt	File	0	file:^console.txt	{ok}
history.zip				e:^history/history.zip	{ok}

file_0 (Bacnet File Descriptor)

○ Status	{ok}
○ Fault Cause	
○ File Ord	file:^console.txt
○ Object Id	file 0
○ Object Name	console.txt
○ Description	J403_Bnet station output
○ File Type	
○ Archive Time	31-Dec-1969 07:00 PM EST
○ File Access Method	Stream Access

The following list describes additional properties in Bacnet file descriptors, not seen in the **Add** or **Edit** dialog for the component:

- Status
  Niagara-only, read-only status flag for the file descriptor component.
- Fault Cause
  Niagara-only, read-only property showing an explanatory text string if the file descriptor is in fault. For example, if its Object Id is manually edited to be the same as another existing file descriptor—a fault occurs and the Fault Cause would be "`Duplicate Object ID`."
- File Type
  BACnet property, as a text string intended to identify the use of this file. Default value is blank, however, the actual exported BACnet File object contains some default value, such as "`text/plain`" or "`application/zip`".
- Archive Time
  Timestamp compared against the actual file timestamp, in order to set the BACnet property "Archive" (True if Archive Time < file timestamp, or False if Archive Time > file timestamp).
- File Access Method
  Read-only BACnet property that indicates the types of file access supported. Currently, this is Stream Access only.

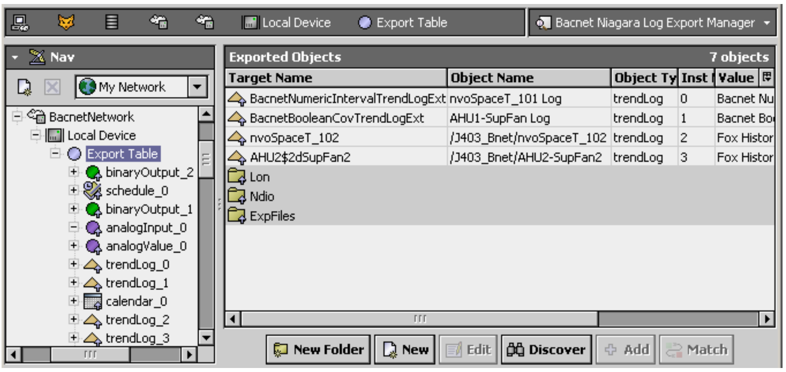
Note: *Additional properties in the exported File object exist, even though not visible in the Bacnet File Descriptor property sheet. These properties include:*

- File Size (integer, in bytes)
- Modification Date (actual file timestamp)
- Archive (True or False, see "Archive Time" property above)
- Read Only (True or False, as determined by the local Java file system)

Bacnet Niagara Log Export Manager

Use the Bacnet Niagara Log Export Manager (Figure 4-20) to export standard Niagara station histories as BACnet Trend Log objects, as well as manage any BacnetTrendLogDescriptors.

Figure 4-20 *Bacnet Niagara Log Export Manager view*



Exported Objects — 7 objects

Target Name	Object Name	Object Ty	Inst	Value
BacnetNumericIntervalTrendLogExt	nvoSpaceT_101 Log	trendLog	0	Bacnet Nu
BacnetBooleanCovTrendLogExt	AHU1-SupFan Log	trendLog	1	Bacnet Bo
nvoSpaceT_102	/J403_Bnet/nvoSpaceT_102	trendLog	2	Fox Histor
AHU2$2dSupFan2	/J403_Bnet/AHU2-SupFan2	trendLog	3	Fox Histor
Lon				
Ndio				
ExpFiles				

Note: *If fully BACnet-compliant Trend Log objects are required, you must configure the source Niagara points with one of the BacnetTrendLogExt extensions (from the* `bacnet` *palette). For details, see the next section "About Trend Log exports".*

The following sections provide more details:

- About Trend Log exports
- About Discover in Bacnet Niagara Log Export Manager
- Add (and Edit) in Bacnet Niagara Log Export Manager
- Properties of Bacnet trend log descriptors

About Trend Log exports

BACnet Trend Log objects closely resemble standard NiagaraAX histories in many ways. However, a fully-compliant BACnet Trend Log includes *additional* data such as sequence numbers, as well as record entries for log events such as log enable, log disable, and buffer purge within the log data.

Therefore, depending on requirements, there are two methods to export a NiagaraAX history:

- Whenever a fully BACnet-compliant Trend Log object is needed, you must paste a special BacnetTrendLogExt extension (from the `bacnet` palette) into the source point in the station, *instead* of the typical history extension. In this case, the resulting history will provide the additional BACnet-compliant data such as sequence numbers and log events. For more details, see the next section "About BacnetTrendLogExt extensions".

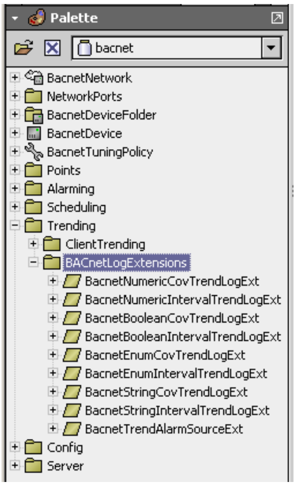
- If retroactively exporting to BACnet, and a Trend Log object that provides "by time" and "by index" access *only* is sufficient, export by simply selecting any "standard" Niagara history (created by a standard history extension). Do this in the Bacnet Niagara Log Export Manager, using a "Discover." In this case, the exposed Trend Log object does not provide sequence numbers or "by sequence number" access, because this information was not stored within the original history.

See "Differences in BACnet-exposed histories" on page 4-75 for an example of how the histories differ.

About BacnetTrendLogExt extensions

Find BacnetTrendLogExts in the `bacnet` palette, under the `Trending` folder (Figure 4-21).

Figure 4-21 *BacnetTrendLogExts in the bacnet palette*



Palette

bacnet

- BacnetNetwork
- NetworkPorts
- BacnetDeviceFolder
- BacnetDevice
- BacnetTuningPolicy
- Points
- Alarming
- Scheduling
- Trending
  - ClientTrending
  - BACnetLogExtensions
    - BacnetNumericCovTrendLogExt
    - BacnetNumericIntervalTrendLogExt
    - BacnetBooleanCovTrendLogExt
    - BacnetBooleanIntervalTrendLogExt
    - BacnetEnumCovTrendLogExt
    - BacnetEnumIntervalTrendLogExt
    - BacnetStringCovTrendLogExt
    - BacnetStringIntervalTrendLogExt
    - BacnetTrendAlarmSourceExt
- Config
- Server

You can use one of these *instead* of a "standard history extension," pasting it in any point in which you need to export its history as a fully BACnet-compliant Trend Log object. For example, if you have a Lon proxy point of type NumericPoint, for "nvoSpaceTemp" representing room temperature, you could copy a BacnetNumericIntervalTrendLogExt into that NumericPoint. Its resulting history (created only after you *enable* it, as with any history extension) will now be fully BACnet-compliant.

By default, when you add one of the BacnetTrendLogExt extension types to a component, note that a corresponding Bacnet log descriptor of type BacnetTrendLogDescriptor is *automatically created* in the *root* of the Export Table. You still define the normal collection parameters in the BacnetTrendLogExt history extension (of the source component), however, there are additional properties in its associated Bacnet Trend Log descriptor.

BacnetTrendLogExt extension types As found in the `bacnet` palette under the `Trending`, `Bacnet-LogExtensions` folder (Figure 4-21), the different types of BacnetTrendLogExt include:

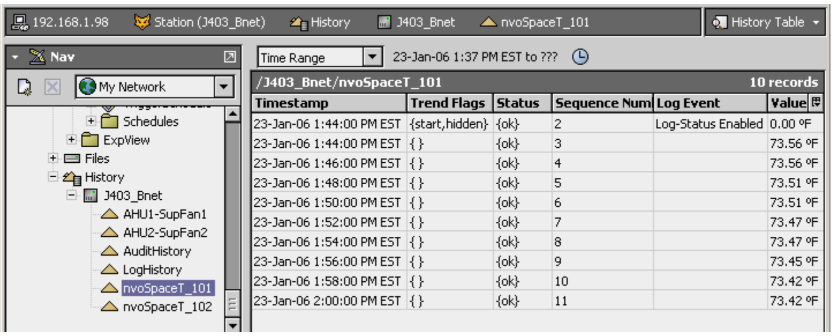
- BacnetBooleanCovTrendLogExt
- BacnetBooeanIntervalTrendLogExt
- BacnetEnumCovTrendLogExt
- BacnetEnumIntervalTrendLogExt
- BacnetNumericCovTrendLogExt
- BacnetNumericIntervalTrendLogExt
- BacnetStringCovTrendLogExt
- BacnetStringIntervalTrendLogExt

In addition, there is a BacnetTrendLogAlarmSourceExt that you can paste under any of the extensions above. This provides intrinsic BACnet alarming/notification for buffer near-full events.

Differences in BACnet-exposed histories

A Niagara history created by a BacnetTrendLogExt has extra fields for sequence numbers, and also "Log Events," as shown in Figure 4-22.

Figure 4-22 *History table for Niagara history created by a BacnetTrendLogExt*

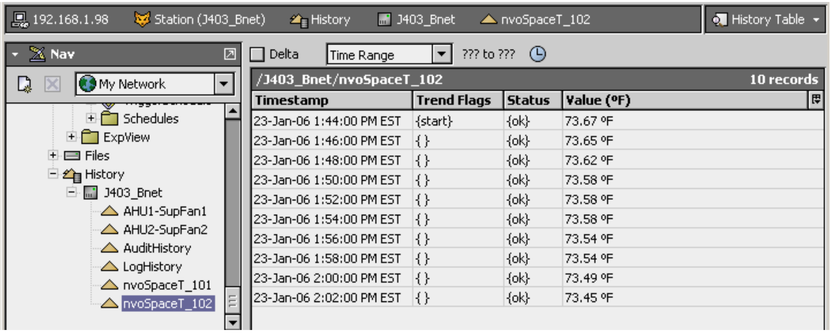


/J403_Bnet/nvoSpaceT_101 10 records

Timestamp	Trend Flags	Status	Sequence Num	Log Event	Value
23-Jan-06 1:44:00 PM EST	{start,hidden}	{ok}	2	Log-Status Enabled	0.00 °F
23-Jan-06 1:44:00 PM EST	{ }	{ok}	3		73.56 °F
23-Jan-06 1:46:00 PM EST	{ }	{ok}	4		73.56 °F
23-Jan-06 1:48:00 PM EST	{ }	{ok}	5		73.51 °F
23-Jan-06 1:50:00 PM EST	{ }	{ok}	6		73.51 °F
23-Jan-06 1:52:00 PM EST	{ }	{ok}	7		73.47 °F
23-Jan-06 1:54:00 PM EST	{ }	{ok}	8		73.47 °F
23-Jan-06 1:56:00 PM EST	{ }	{ok}	9		73.45 °F
23-Jan-06 1:58:00 PM EST	{ }	{ok}	10		73.42 °F
23-Jan-06 2:00:00 PM EST	{ }	{ok}	11		73.42 °F

Compare that with a "standard" Niagara history, as shown in Figure 4-23.

Figure 4-23 *History table for standard Niagara history*



/J403_Bnet/nvoSpaceT_102 10 records

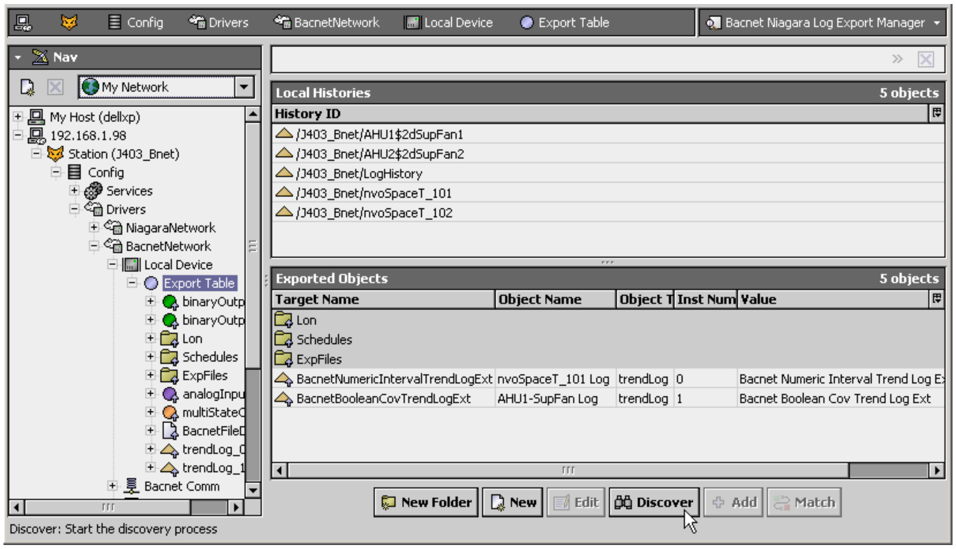
Timestamp	Trend Flags	Status	Value (°F)
23-Jan-06 1:44:00 PM EST	{start}	{ok}	73.67 °F
23-Jan-06 1:46:00 PM EST	{ }	{ok}	73.65 °F
23-Jan-06 1:48:00 PM EST	{ }	{ok}	73.62 °F
23-Jan-06 1:50:00 PM EST	{ }	{ok}	73.58 °F
23-Jan-06 1:52:00 PM EST	{ }	{ok}	73.58 °F
23-Jan-06 1:54:00 PM EST	{ }	{ok}	73.58 °F
23-Jan-06 1:56:00 PM EST	{ }	{ok}	73.54 °F
23-Jan-06 1:58:00 PM EST	{ }	{ok}	73.54 °F
23-Jan-06 2:00:00 PM EST	{ }	{ok}	73.49 °F
23-Jan-06 2:02:00 PM EST	{ }	{ok}	73.45 °F

About Discover in Bacnet Niagara Log Export Manager

You typically use **`Discover`** in the Bacnet Niagara Log Export Manager for any "standard history" you wish to export to BACnet as Trend Log object—that is, without the ability for sequence number access from BACnet. Using Discover is not necessary for the history of any component in which you have already added one of the BacnetTrendLogExts—as the Bacnet driver automatically added a BacnetTrendLogDescriptor under the root of the Export Table.

When you click **`Discover`**, the top pane shows the histories under the station's folder, as shown in Figure 4-24.

Figure 4-24 *Discover in Bacnet Niagara Log Export Manager*



Local Histories 5 objects

History ID
/J403_Bnet/AHU1$2dSupFan1
/J403_Bnet/AHU2$2dSupFan2
/J403_Bnet/LogHistory
/J403_Bnet/nvoSpaceT_101
/J403_Bnet/nvoSpaceT_102

Exported Objects 5 objects

Target Name	Object Name	Object T	Inst Num	Value
Lon				
Schedules				
ExpFiles				
BacnetNumericIntervalTrendLogExt	nvoSpaceT_101 Log	trendLog	0	Bacnet Numeric Interval Trend Log Ex
BacnetBooleanCovTrendLogExt	AHU1-SupFan Log	trendLog	1	Bacnet Boolean Cov Trend Log Ext

Discover: Start the discovery process

As with other manager views featuring online discovery, “Learn Mode” in Bacnet Niagara Log Export Manager has two panes:

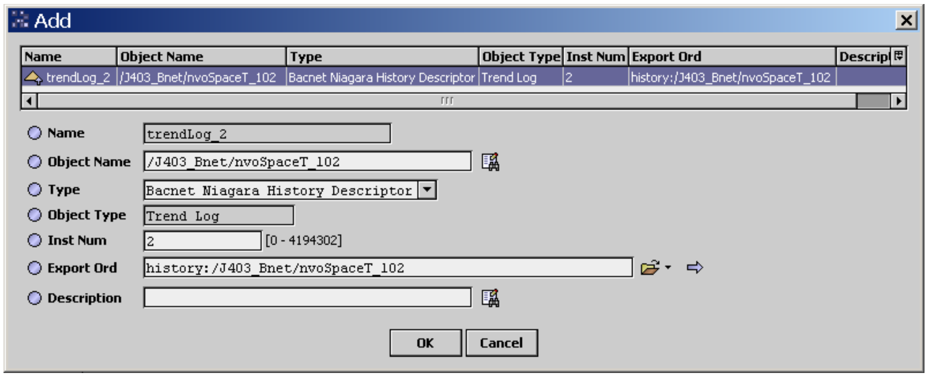
- Top (discovered) **Local Histories** pane—listing all histories in the local station.
- Bottom (database) **Exported Objects** pane—listing histories currently exported to BACnet. These include both types of log descriptor components—each essentially a pointer to the exposed history, with additional slots that determine how BACnet access/writes are handled. See the next section, “Add (and Edit) in Bacnet Niagara Log Export Manager” for more details.

When exporting, the manager automatically performs “object ID maintenance” on exported files, ensuring that no duplicate combinations of object *type* and *instance number* are created.

Add (and Edit) in Bacnet Niagara Log Export Manager

When you click **Add** with a history highlighted to export, the **Add** dialog (Figure 4-25) contains a number of fields with either default or empty values, described below.

Figure 4-25 *Add dialog in Bacnet Niagara Log Export Manager*



Name	Object Name	Type	Object Type	Inst Num	Export Ord	Descrip
trendLog_2	/J403_Bnet/nvoSpaceT_102	Bacnet Niagara History Descriptor	Trend Log	2	history:/J403_Bnet/nvoSpaceT_102	

○ Name: trendLog_2
○ Object Name: /J403_Bnet/nvoSpaceT_102
○ Type: Bacnet Niagara History Descriptor
○ Object Type: Trend Log
○ Inst Num: 2 [0 - 4194302]
○ Export Ord: history:/J403_Bnet/nvoSpaceT_102
○ Description:

All fields in this dialog apply separately to each highlighted (for export) history, as follows:

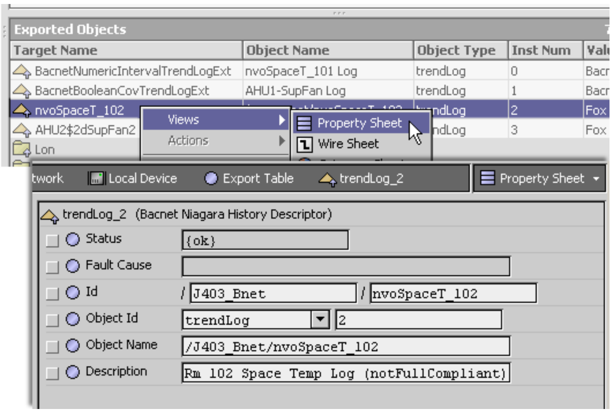
- Name
  Read-only Niagara component name for the Bacnet log descriptor, which defaults to the combination of: `trendLog_<instance number>`.
- Object Name
  The “exposed to BACnet” name for this history. By default, the *entire component path* under the station’s History is included, using period (“/”) delimiters between *parent.child* levels. This enforces (externally) the BACnet requirement for unique names for all objects in a device.
  Note: *You can shorten or edit object name, either now or later. However, please note that each Object Name should be unique among all server descriptors under the Export Table.*
- Type
  Type of descriptor, in the **Add** dialog it is: `Bacnet Niagara History Descriptor`.
  In any subsequent Edit dialog, this cannot be changed.
- Object Type
  Read-only reflection of the “BACnet Object type” exposed, in this case always: `Trend Log`.
- Inst Num
  The instance number portion of this object’s “Object ID,” which must be unique within the station for this (exported) BACnet Trend Log object type. By default, the export manager enforces this.
- Export Ord
  Station’s Ord location of the source history, using standard history Ord notation.
- Description
  Optional text string; this appears as the Description property value in the exposed BACnet object.

Note: *After creation (adding) Bacnet log descriptors, note that each descriptor also has additional properties accessible in its property sheet, along with those seen in the* **Add** *and* **Edit** *dialog. See the next section, “Properties of Bacnet file descriptors” for more details.*

Properties of Bacnet trend log descriptors

Each Bacnet trend log descriptor has various properties that affect its BACnet representation. Access these properties in the property sheet view of any log descriptor, as shown in Figure 4-26.

Figure 4-26 *Additional properties of BacnetNiagaraHistoryDescriptor on property sheet*



Exported Objects

Target Name	Object Name	Object Type	Inst Num	Valı
BacnetNumericIntervalTrendLogExt	nvoSpaceT_101 Log	trendLog	0	Bacr
BacnetBooleanCovTrendLogExt	AHU1-SupFan Log	trendLog	1	Bacr
nvoSpaceT_102		ndLog	2	Fox
AHU2$2dSupFan2		ndLog	3	Fox
Lon				

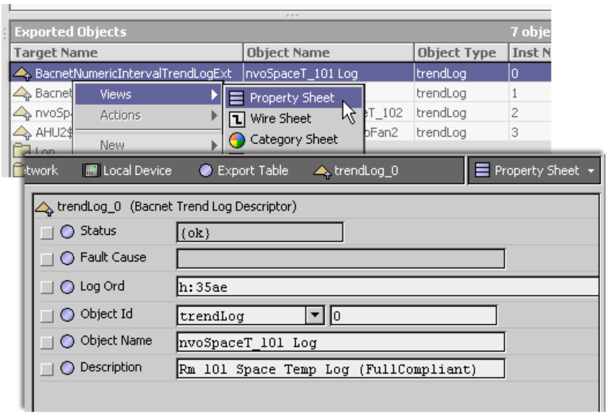
trendLog_2 (Bacnet Niagara History Descriptor)

□ ○ Status: {ok}
□ ○ Fault Cause:
□ ○ Id: / J403_Bnet / nvoSpaceT_102
□ ○ Object Id: trendLog ▾ 2
□ ○ Object Name: /J403_Bnet/nvoSpaceT_102
□ ○ Description: Rm 102 Space Temp Log (notFullCompliant)

The following list describes additional properties in Bacnet log descriptors, not seen in the **Add** or **Edit** dialog for the component:

- Status
  Niagara-only, read-only status flag for the file descriptor component.
- Fault Cause
  Niagara-only, read-only property showing an explanatory text string if the log descriptor is in fault. For example, if its Object Id is manually edited to be the same as another existing log descriptor—a fault occurs and the Fault Cause would be "`Duplicate Object ID`."
- Id
  Appears only if a BacnetNiagaraHistoryDescriptor (as shown Figure 4-26), and reflects the same Niagara history "Export Ord" in the station.
- Log Ord
  Appears only if a BacnetTrendLogDescriptor (as shown Figure 4-27), and reflects the handle of the the specific BacnetTrendLogExt responsible for creating the exported history.

Figure 4-27 *Properties of BacnetTrendLogDescriptor*



Exported Objects 7 obje

Target Name	Object Name	Object Type	Inst N
BacnetNumericIntervalTrendLogExt	nvoSpaceT_101 Log	trendLog	0
Bacnet		trendLog	1
nvoSp	T_102	trendLog	2
AHU2$	bFan2	trendLog	3
Lon			

trendLog_0 (Bacnet Trend Log Descriptor)

□ ○ Status: {ok}
□ ○ Fault Cause:
□ ○ Log Ord: h:35ae
□ ○ Object Id: trendLog ▾ 0
□ ○ Object Name: nvoSpaceT_101 Log
□ ○ Description: Rm 101 Space Temp Log (FullCompliant)

Local Device notes

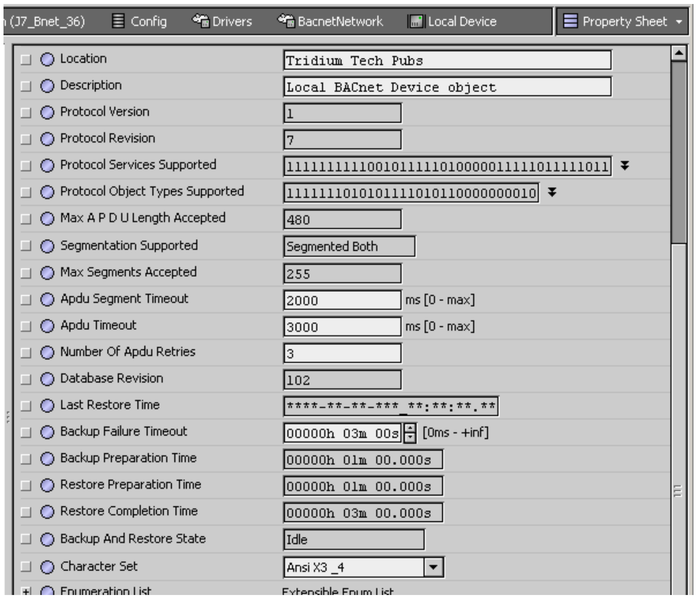
Relative to BACnet server operation, the following sections explain items about the Local Device, such as found on its property sheet:

- Local Device properties
- Local Device container slots
- Local Device backup and restore properties

Local Device properties

Various properties of the BacnetNetwork's Local Device affect server-side visibility or operation, in addition to the critical Object Id property (see Figure 3-14 on page 21). Figure 4-28 shows a portion of the property sheet of a Local Device, with some of these properties.

Figure 4-28 *Some properties of Bacnet Local Device affect server operation*



Location	Tridium Tech Pubs	
Description	Local BACnet Device object	
Protocol Version	1	
Protocol Revision	7	
Protocol Services Supported	1111111111001011111010000011111011111011	
Protocol Object Types Supported	111111010101111010110000000010	
Max A P D U Length Accepted	480	
Segmentation Supported	Segmented Both	
Max Segments Accepted	255	
Apdu Segment Timeout	2000	ms [0 - max]
Apdu Timeout	3000	ms [0 - max]
Number Of Apdu Retries	3	
Database Revision	102	
Last Restore Time	****-**-**-*** **:**:**.**	
Backup Failure Timeout	00000h 03m 00s	[0ms - +inf]
Backup Preparation Time	00000h 01m 00.000s	
Restore Preparation Time	00000h 01m 00.000s	
Restore Completion Time	00000h 03m 00.000s	
Backup And Restore State	Idle	
Character Set	Ansi X3 _4	
Enumeration List	Extensible Enum List	

The following descriptions apply to some of these writable properties:

Note: *The three APDU settings below are used in both client and server operation. Many sites will require different settings of these parameters.*

For instance, for a small IP-based network, optimal settings may be more like:

- Apdu Timeout = 5000 ms
- Number of Apdu retries =1

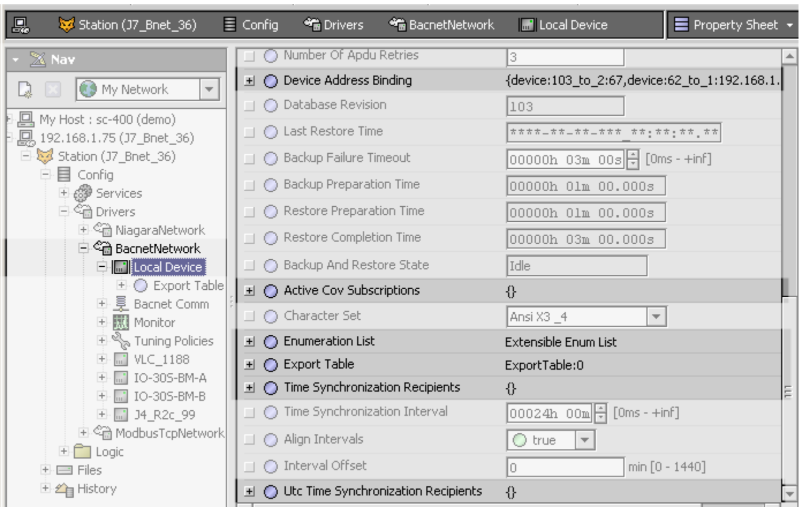
Large, routed networks may require higher Apdu Timeouts.

- Location — Optional text string to describe location of the Niagara host (default is "unknown.")
- Description — Optional text string to describe the BACnet Device object.
- Apdu Segment Timeout — (default is 2000 ms) Defines the time waited before retransmission of an APDU segment (only relevant if devices are doing segmentation).
- Apdu Timeout — (default is 3000 ms) Defines the time waited before retransmission of an APDU requiring acknowledgment, for which no acknowledgement has been received.
- Number of Apdu retries — (default is 3) Defines the maximum number of an APDU will be retransmitted.
- Backup Failure Timeout — (default is 3 minutes) See "Local Device backup and restore properties" on page 4-80 for details on this property and other read-only backup related properties.
- Character Set — (default ANSI X3.4) Defines the character set supported, with other selections being IBM/Microsoft DBCS, JIS C 6226, ISO 10646 (UCS-4), ISO 10646 (UCS-2), ISO 8859-1, and "Unknown."

Local Device container slots

The BacnetNetwork's Local Device provides a few container slots, found near the bottom of the Local Device's property sheet (Figure 4-29).

Figure 4-29 Container slots near bottom of Local Device property sheet



Station (J7_Bnet_36) | Config | Drivers | BacnetNetwork | Local Device | Property Sheet

Nav — My Network

- My Host : sc-400 (demo)
- 192.168.1.75 (J7_Bnet_36)
  - Station (J7_Bnet_36)
    - Config
      - Services
      - Drivers
        - NiagaraNetwork
        - BacnetNetwork
          - Local Device
            - Export Table
          - Bacnet Comm
          - Monitor
          - Tuning Policies
          - VLC_1188
          - IO-30S-BM-A
          - IO-30S-BM-B
          - J4_R2c_99
        - ModbusTcpNetwork
      - Logic
    - Files
    - History

Number Of Apdu Retries	3
Device Address Binding	{device:103_to_2:67,device:62_to_1:192.168.1.
Database Revision	103
Last Restore Time	****-**-**-*** **:**:**.**
Backup Failure Timeout	00000h 03m 00s [0ms - +inf]
Backup Preparation Time	00000h 01m 00.000s
Restore Preparation Time	00000h 01m 00.000s
Restore Completion Time	00000h 03m 00.000s
Backup And Restore State	Idle
Active Cov Subscriptions	{}
Character Set	Ansi X3 _4
Enumeration List	Extensible Enum List
Export Table	ExportTable:0
Time Synchronization Recipients	{}
Time Synchronization Interval	00024h 00m [0ms - +inf]
Align Intervals	○ true
Interval Offset	0 min [0 - 1440]
Utc Time Synchronization Recipients	{}

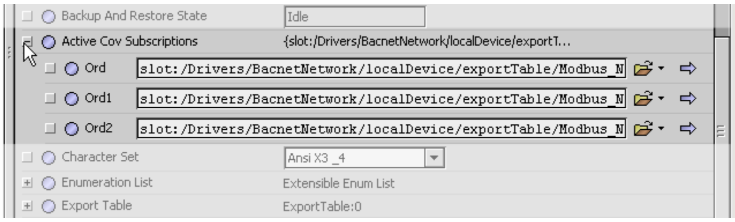
Apart from the Export Table (see "About the Export Table" on page 4-60), these container slots are as follows (note some may be *hidden* by default, by design):

- Device Address Binding — (may be *hidden*) has entries reflecting client-side operation, essentially a list of Device objects (by object ID) and BACnet device address. These identify actual device parameters used when making client BACnet service requests to these devices. For informational use only.
- Active Cov Subscriptions — (may be *hidden*) applies to server operation. See About Active Cov Subscriptions.
- Enumeration List — a list of the manufacturer-specific extensions to extensible BACnet enumerations that are defined and applicable for this specific device. In the case of the Local Device, there is no need for editing. In client BacnetDevice components, however, editing items in the Enumeration List may provide utility for proprietary items, perhaps property IDs.
- Time Synchronization Recipients — (AX-3.4 and later). This and other time synchronization related containers and properties were moved from the `bacnetws` module into the core `bacnet` module, with a full collection of properties starting in AX-3.5. They are in the property sheet of any BacnetNetwork's LocalDevice component. For more details, see "Time synchronization properties" on page 3-22.
- Utc Time Synchronization Recipients — (AX-3.4 and later). See above description.

About Active Cov Subscriptions

The NiagaraAX Bacnet driver provides "server-side Subscribe_COV" to remote BACnet devices (on station objects exposed as BACnet objects). The Active Cov Subscriptions slot in the Local Device contains a dynamic list of current Subscribe_COV subscriptions. Each appears as an Ord to the export descriptor. Figure 4-30 shows an example with 3 active COV subscriptions.

Figure 4-30 Active COV Subscriptions in Bacnet Local Device



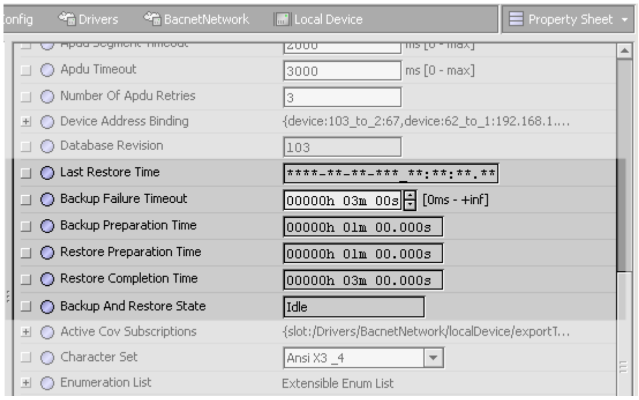
Backup And Restore State	Idle
Active Cov Subscriptions	{slot:/Drivers/BacnetNetwork/localDevice/exportT...
Ord	slot:/Drivers/BacnetNetwork/localDevice/exportTable/Modbus_N
Ord1	slot:/Drivers/BacnetNetwork/localDevice/exportTable/Modbus_N
Ord2	slot:/Drivers/BacnetNetwork/localDevice/exportTable/Modbus_N
Character Set	Ansi X3 _4
Enumeration List	Extensible Enum List
Export Table	ExportTable:0

Depending on the actual implementation, there may be *many* active COV subscriptions.

Local Device backup and restore properties

Starting in AX-3.2, server-side support was added for the BACnet DM-BR-B BIBB, such that a remote BACnet Workstation client can backup and restore the configuration of a station running on a JACE host. Several related properties were added to the LocalBacnetDevice, as shown in Figure 4-31.

Figure 4-31 *Backup and restore related properties in LocalBacnetDevice*



Apdu Segment Timeout	2000	ms [0 - max]
Apdu Timeout	3000	ms [0 - max]
Number Of Apdu Retries	3	
Device Address Binding	{device:103_to_2:67,device:62_to_1:192.168.1....	
Database Revision	103	
Last Restore Time	****-**-**-*** **:**:**.**	
Backup Failure Timeout	00000h 03m 00s	[0ms - +inf]
Backup Preparation Time	00000h 01m 00.000s	
Restore Preparation Time	00000h 01m 00.000s	
Restore Completion Time	00000h 03m 00.000s	
Backup And Restore State	Idle	
Active Cov Subscriptions	{slot:/Drivers/BacnetNetwork/localDevice/exportT...	
Character Set	Ansi X3 _4	
Enumeration List	Extensible Enum List	

The following descriptions apply to these properties:

- Last Restore Time — Timestamp of when a backup .dist file was last restored by a remote BACnet Workstation client. If never, all timestamp fields show asterisks (*).
- Backup Failure Timeout —(Default is 3 minutes). Specifies the time the server device must wait before deciding that the client has given up, and can leave backup/restore mode. The client typically writes this to the server prior to beginning the backup/restore procedure.
- Backup Preparation Time — (Default is 1 minute). Specifies the time allotted for the station, after receiving the backup request, to locally save the station database. May need to be increased in the case of a large station.
- Restore Preparation Time — (Default is 1 minute). Specifies the time allotted for the station, after receiving a restore request, to gracefully shut down the station and close files before beginning the dist file installation. May need to be increased in the case of a large station
- Restore Completion Time — (Default is 3 minutes). Specifies the time allotted for the station, after completing a dist file install from a restore, to restart and become responsive. May need to be increased in the case of a large station.
- Backup and Restore State — (read-only) Shows the current backup and restore state, where “Idle” is shown when no operation is active. This can change to “Preparing for backup” and so forth.

Note: *Properties above were defined in a recently accepted proposal to the SSPC to fix known issues with the backup and restore feature, as currently defined in the spec. The proposal will be the subject of a future public review document, but it will likely be accepted, and so it has been implemented for now. When the document becomes an official part of the standard, there will likely be an update patch to adjust the property identifier indices.*

See the next section, “About backup and restore operations”, for additional details.

About backup and restore operations

Any backup or restore initiated from a remote BACnet Workstation client requires several things before it is acted upon by the station, as follows:

- The station user “BACnet” requires *admin-level write* permissions on the BacnetNetwork and BacnetLocalDevice, at a minimum. The *password* for the BACnet user must be a non-blank (non-default) value, and this password must be known to the operator of the remote BACnet Workstation.
- The normally-hidden “Reinitialize Allowed” property of the `Server` layer component (child of the BacnetComm container under the BacnetNetwork) must be unhidden and set to a value of “true,” as it defaults to “false.” Once changed, it can be rehidden if desired. See “About Bacnet Comm: Client, Server, and Transport” on page 3-16 for related details.

If the conditions above are met, and if an AX-3.2 or later station receives a backup or restore request, these operations occur as described below:

- Backup operation
- Restore operation

Note: *Backup and restore procedures used adhere to Clause 19 of the BACnet Specification.*

Backup operation Backup operation proceeds as follows:

- The LocalBacnetDevice's "Backup and Restore State" property changes from "Idle" to "Preparing for backup."
- The station's BackupService begins preparing a backup .dist, containing the following:
  - Station database - config.bog
  - Alarm history database
  - History database
  - Module metadata - information about which version of each module is loaded.
    Note: *Modules themselves are not backed up.*
- The assembled backup .dist file is then exported as a File object and is part of the configurationFiles property of the device object. The backup client then reads the File object using BACnet file access services.
- The LocalBacnetDevice's "Backup and Restore State" property changes back to "Idle," and normal station operation resumes.

Restore operation Restore operation proceeds as follows:

- The LocalBacnetDevice's "Backup and Restore State" property changes from "Idle" to "Preparing for restore."
- The station begins a graceful shutdown, saving opened files.
- The backup .dist file is sent from the remote BACnet Workstation client to the JACE platform.
- The backup .dist file is installed, and then the JACE host rebooted.
- Upon station startup, the LocalBacnetDevice's "Backup and Restore State" property changes back to "Idle," the "Last Restore Time" reflects the time of the backup .dist installation, and normal station operation resumes.

Chapter 5

Niagara Bacnet Alarming

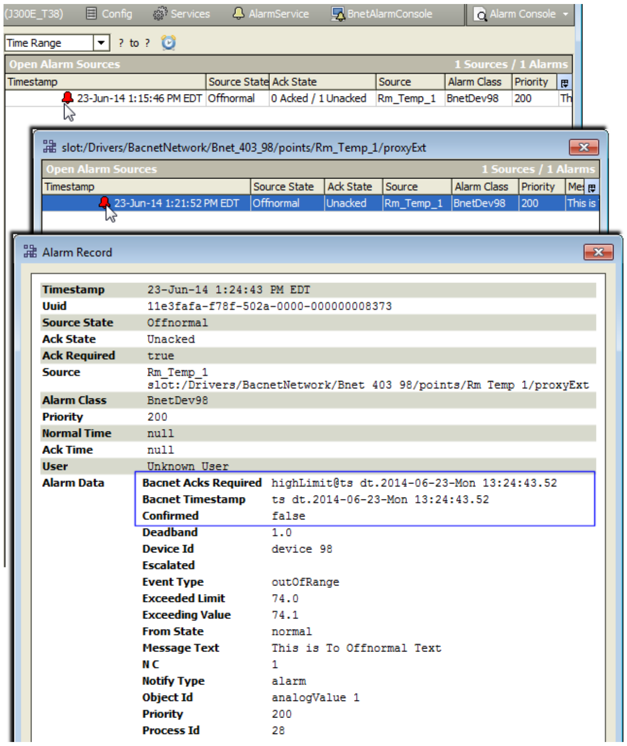
NiagaraAX supports BACnet “intrinsic event reporting”, more commonly called alarming. Support is available for both client and server operations, meaning the station can both receive (and reroute) BACnet alarms from Bacnet proxy points, and also send BACnet alarms from components that have been exported as BACnet objects. The following sections provide more details:

- “Receiving BACnet alarms” on page 5-83
    - “Prerequisites and restrictions to receive BACnet alarms” on page 5-84
    - “Configuring to receive BACnet alarms” on page 5-84
    - “BacnetAlarmDeviceExt (reference)” on page 5-85
- “Sending Alarms to BACnet” on page 5-85
    - “Prerequisites and restrictions on exporting alarms to BACnet” on page 5-85
    - “Configuring to send alarms to BACnet” on page 5-86
    - “BacnetDestination (reference)” on page 5-87

Receiving BACnet alarms

Receiving BACnet alarms is available for most stations running the Bacnet driver—meaning as a BACnet client. Incoming alarms from remote BACnet devices are converted to the Niagara alarm format, and can then be routed through the Niagara alarm subsystem to different recipient types (console, station, printer, email, and so on). A console user can acknowledge a BACnet alarm just like a native Niagara alarm.

Figure 5-1 *Example BACnet alarm received in NiagaraAX Alarm Console*



Open Alarm Sources — 1 Sources / 1 Alarms

Timestamp	Source State	Ack State	Source	Alarm Class	Priority	
23-Jun-14 1:15:46 PM EDT	Offnormal	0 Acked / 1 Unacked	Rm_Temp_1	BnetDev98	200	Th

slot:/Drivers/BacnetNetwork/Bnet_403_98/points/Rm_Temp_1/proxyExt

Open Alarm Sources — 1 Sources / 1 Alarms

Timestamp	Source State	Ack State	Source	Alarm Class	Priority	Me
23-Jun-14 1:21:52 PM EDT	Offnormal	Unacked	Rm_Temp_1	BnetDev98	200	This is

Alarm Record

Timestamp	23-Jun-14 1:24:43 PM EDT	
Uuid	11e3fafa-f78f-502a-0000-000000008373	
Source State	Offnormal	
Ack State	Unacked	
Ack Required	true	
Source	Rm_Temp_1 slot:/Drivers/BacnetNetwork/Bnet 403 98/points/Rm Temp 1/proxyExt	
Alarm Class	BnetDev98	
Priority	200	
Normal Time	null	
Ack Time	null	
User	Unknown User	
Alarm Data	**Bacnet Acks Required**	highLimit@ts dt.2014-06-23-Mon 13:24:43.52
	Bacnet Timestamp	ts dt.2014-06-23-Mon 13:24:43.52
	Confirmed	false
	Deadband	1.0
	Device Id	device 98
	Escalated	
	Event Type	outOfRange
	Exceeded Limit	74.0
	Exceeding Value	74.1
	From State	normal
	Message Text	This is To Offnormal Text
	N C	1
	Notify Type	alarm
	Object Id	analogValue 1
	Priority	200
	Process Id	28

Prerequisites and restrictions to receive BACnet alarms

The following prerequisites and restrictions apply on receiving BACnet alarms:

- To be received, a BACnet alarm must be generated from a BACnet device represented in the station. Otherwise, the alarm is discarded.
  - Ideally, the alarming BACnet object should be represented by a Bacnet proxy point, in which case the Niagara alarm record source provides the Niagara proxy point name and ord.
  - If the alarming BACnet object is *not* proxied in the station, the Niagara alarm record shows the source as the **Alarms** device extension (ord) of the corresponding BacnetDevice.

  In either case, alarm data details provide the source Object Id, for example “analogInput 3”, current “toState” and “statusFlags”, as well as any alarm message text.
- AlarmClass components in the station’s AlarmService receive BACnet alarms. Each device uses only *one* AlarmClass, as referenced in each BacnetDevice’s Alarms device extension. You can specify the same one (e.g. the DefaultAlarmClass), or create and use separate AlarmClasses, as needed.
- In the Notification Class object used by the sending BACnet device, any receiving station must be added in the “Recipient_List” property, using its numerical device ID (Local Device, Object Id). You can do this using the BACnet device’s native configuration tool, or if the device supports this, directly from Niagara using corresponding “Config” components for objects in that device.

Configuring to receive BACnet alarms

Note: *See “Prerequisites and restrictions to receive BACnet alarms” on page 5-84*

The following procedures explain how to get started receiving BACnet alarms.

- “To configure to receive BACnet alarms”
- “To configure a BACnet device to send alarms to Niagara”

To configure to receive BACnet alarms

Step 1 In the station, create BacnetDevice components for all devices from which you wish to receive alarms. It is also recommended to create Bacnet proxy points for all BACnet objects that will send alarms to the station. See “Create BacnetDevices” on page 2-6 and “Create Bacnet proxy points” on page 2-8.

Step 2 Under the station’s AlarmService, add AlarmClass components as needed, setting ack requirements and priorities. Each BacnetDevice can use only *one* AlarmClass (either used by other devices, or unique to that device.).

Step 3 Link the AlarmClass components into whatever alarm recipient objects are needed (AlarmConsole, etc).

Step 4 For each BacnetDevice in the station, in its **Alarms** device extension:

- Specify the AlarmClass component to use.
- Set the “Niagara Process Id” to match the BACnet process identifier (integer value) used in sending alarms to the station.

See “BacnetAlarmDeviceExt (reference)” on page 5-85 for related details.

Step 5 Using third-party tools for each remote BACnet device, configure the Notification Class object that routes alarms to include (add) the Device object ID of the station to its “Recipient_List” property. This is the numerical value seen in the station’s BacnetNetwork, LocalDevice, “Object Id” property.

You may be able to use Niagara for this. See “To configure a BACnet device to send alarms to Niagara”

To configure a BACnet device to send alarms to Niagara

Note: *Sometimes a remote BACnet device may not allow this configuration. In this case, you must use third-party tools to configure Notification Class objects in a BACnet device as described below.*

With the station opened, and a **BacnetDevice** already added and communicating in the station:

Step 1 Go to the **Config** device extension’s **Config Manager** view.

Discover the objects in the device.

Step 2 Add the **Notification Class** config object for the alarms you wish to receive.

Note: *To determine which one to add, add a config object for an object that will be alarming, and inspect its Notification Class property. This specifies which Notification Class object will route those alarms.*

Delete unused config objects after determining referenced notification classes.

Step 3 Right-click the **Notification Class** config object, and select **Actions > Add Destination**.

In the **Add Destination** dialog, expand the **Recipient** field and enter the station’s device ID, replacing the “`-1`” (with value of the station’s BacnetNetwork, LocalDevice, “Object Id” property).

Specify the (integer) **`Process Identifier`** to use in sending alarms to the station (default is `0`). You may configure other properties as desired for when you wish alarms to be sent.

Click **`OK`**.

Step 4 **`Save`** changes.

Alarms should now be sent to the (client) station.

Note: *For BACnet alarms from a device to be successfully received in a client station, the Process Identifier in the Recipient List entry for the station (in the BACnet Notification Class object of the sending device) must be matched when configuring the BacnetDevice's Alarms device extension, "Niagara Process Id" property.*

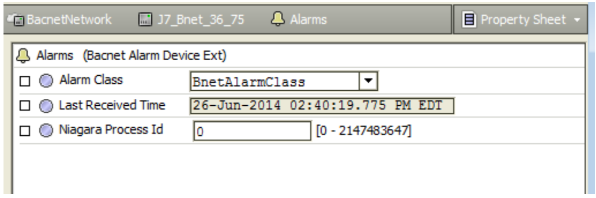
See "BacnetAlarmDeviceExt (reference)" for related details.

BacnetAlarmDeviceExt (reference)

Each BacnetDevice has an 🔔 **`Alarms`** device extension. This extension applies to BACnet event notifications (alarms) sent to the station by that device.

BacnetAlarmDeviceExt properties

Figure 5-2 *BacnetAlarmDeviceExt (Alarms) properties*



Alarms (Bacnet Alarm Device Ext)

□ ○ Alarm Class BnetAlarmClass

□ ○ Last Received Time 26-Jun-2014 02:40:19.775 PM EDT

□ ○ Niagara Process Id 0 [0 - 2147483647]

BacnetAlarmDeviceExt properties are as follows:

- **Alarm Class**
  Specifies the Niagara AlarmClass component used by all BACnet event notifications received from this device.
- **Last Received Time**
  Date-timestamp of the last BACnet event notification received from this device.
- **Niagara Process Id**
  (Integer) Specifies a process ID, which needs to match the BACnet process identifier used to send notifications to this client station. The default value is `0`.

For related details, see "Receiving BACnet alarms" on page 5-83, including "Prerequisites and restrictions to receive BACnet alarms" on page 5-84 and "Configuring to receive BACnet alarms" on page 5-84.

Sending Alarms to BACnet

Any station that is exporting control points to BACnet can also route BACnet alarms from those points to BACnet devices. A "BacnetDestination" component in the "Alarming" folder of the `bacnet` palette is used to specify a receiving BACnet client device.

Prerequisites and restrictions on exporting alarms to BACnet

The following prerequisites and restrictions apply on exporting alarms to BACnet:

- Only Niagara components in the station that are exported as BACnet objects can generate BACnet alarms, where alarming capability is "intrinsic" to that object. For example, a Niagara control point exposed as an Analog Input object must use an OutOfRange algorithm (in Niagara, an OutOfRangeAlarmExt)—another alarm extension, say the StatusAlarmExt, is not supported.
- In general, only components exported as BACnet object types Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Multistate Input, Multistate Output, or Multistate Value are good candidates. (Exported Calendar and Schedule components are not alarmable). In the station these are typically control points (often proxy points), but may include various components from the `kitControl` module—at least those with a "NullProxyExt".
- Only *one* alarm extension is supported on a Niagara point configured for BACnet alarming (referencing an AlarmClasss linked to a BacnetDestination).
- A separate BacnetDestination component must be added under the station's AlarmService for each remote BACnet client to which alarms need to be sent (one-to-one BacnetDestination to device). In this component, you specify the unique BACnet device identifier for this device, and possibly make

other non-default configuration changes.

- Each AlarmClass component that you link to a BACnetDestination component must be exported to BACnet, as a *Notification Class* object. Linked BacnetDestination components automatically appear as entries in the “Recipient List” of these Notification Class objects.
- Unless “escalation levels” are defined in AlarmClass components that are used, alarms are not regenerated. If a BACnet device is down when Niagara sends an alarm, the alarm is not saved for retransmission. Of course the station maintains a record of alarms in the alarm database.

Configuring to send alarms to BACnet

Note: *See “Prerequisites and restrictions on exporting alarms to BACnet” on page 5-85*

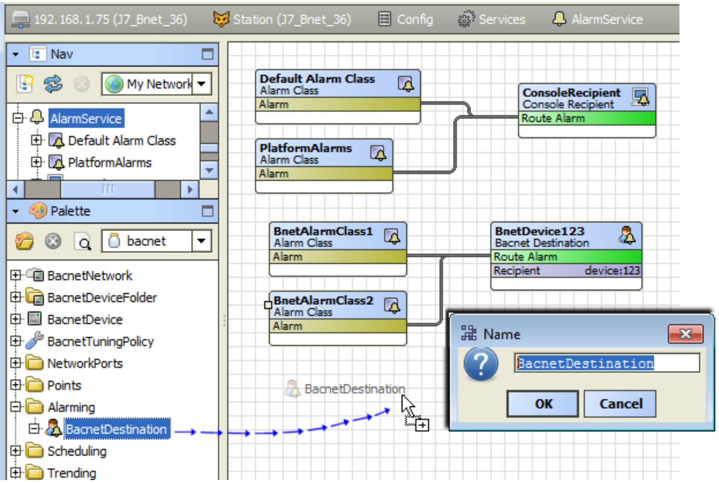
The following procedures explain how to get started sending Niagara alarms to BACnet client devices

- “To configure to send BACnet alarms”
- “To configure a BACnet device to send alarms to Niagara”

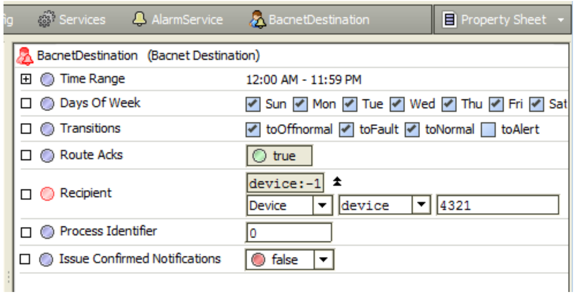
To configure to send BACnet alarms

Step 1 Under the station’s AlarmService, add any AlarmClass components that you want to use with alarmable control points exported to BACnet. Set ack requirements and priorities as needed, but do not export yet.

Step 2 In the station’s AlarmService, add (drag and drop) a **BacnetDestination** recipient from the bacnet palette (“Alarming” folder).



Set the “Recipient” property to the BACnet device ID for a BACnet client to receive alarms.



BacnetDestination (Bacnet Destination)

Time Range 12:00 AM - 11:59 PM

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

Transitions ☑ toOffnormal ☑ toFault ☑ toNormal □ toAlert

Route Acks true

Recipient device:-1 Device device 4321

Process Identifier 0

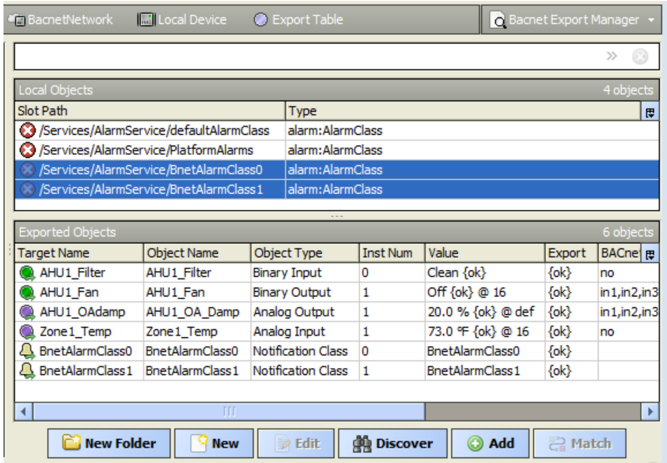
Issue Confirmed Notifications false

For details on other properties, see “BacnetDestination (reference)” on page 5-87.

Step 3 Repeat this for each BACnet client to receive alarms from Niagara (separate BacnetDestination for each remote BACnet client).

Step 4 Link the AlarmClasses used to the BacnetDestination recipients (and and other recipients) as needed.

Step 5 Working from the **`Bacnet Export Manager`**, export all AlarmClass components that are linked to BacnetDestination recipients. These export as only one object type: Notification Class



Local Objects — 4 objects

Slot Path	Type
/Services/AlarmService/defaultAlarmClass	alarm:AlarmClass
/Services/AlarmService/PlatformAlarms	alarm:AlarmClass
/Services/AlarmService/BnetAlarmClass0	alarm:AlarmClass
/Services/AlarmService/BnetAlarmClass1	alarm:AlarmClass

Exported Objects — 6 objects

Target Name	Object Name	Object Type	Inst Num	Value	Export	BACne
AHU1_Filter	AHU1_Filter	Binary Input	0	Clean {ok}	{ok}	no
AHU1_Fan	AHU1_Fan	Binary Output	1	Off {ok} @ 16	{ok}	in1,in2,in3
AHU1_OAdamp	AHU1_OA_Damp	Analog Output	1	20.0 % {ok} @ def	{ok}	in1,in2,in3
Zone1_Temp	Zone1_Temp	Analog Input	1	73.0 °F {ok} @ 16	{ok}	no
BnetAlarmClass0	BnetAlarmClass0	Notification Class	0	BnetAlarmClass0	{ok}	
BnetAlarmClass1	BnetAlarmClass1	Notification Class	1	BnetAlarmClass1	{ok}	

Note: *Find them quickly in a Discover by choosing "`Of Type: Alarm Class`"*

Note the BACnet Notification Class object "instance number" automatically starts at 0 for the first exported AlarmClass, then increments upwards by one.

Step 6 If not already done, add the appropriate Niagara alarm extension to control points you want to alarm to BACnet, choosing one of the AlarmClass components (from Step 1) in their "Alarm Class" property.

Export these control points to BACnet from the **`Bacnet Export Manager`** view of the BacnetNetwork's **`LocalDevice`**, **`ExportTable`** component. See "About the Export Table" on page 4-60 and "Bacnet Export Manager" on page 4-63 for related details.

The station is now configured to generate BACnet alarms for the exported control points. See "Testing BACnet alarm generation" to verify proper operation.

Testing BACnet alarm generation

Note: *(Prerequisite) In order to test Niagara BACnet alarm generation, you should have access to third-party BACnet tools for the remote BACnet clients. In addition, the AlarmClasses involved should be linked to a standard AlarmConsole recipient.*

Step 1 Force into an alarm condition one of the exported Niagara control points that is exported to BACnet (and assigned to an exported AlarmClass). For example, you can do this by adjusting an alarm high limit or low limit.

The alarm is routed via the exported AlarmClass to linked BacnetDestination (client) devices. This translates the alarm from a Niagara format to a BACnet format. At the same time, the alarm should be available in the standard Niagara alarm console.

Step 2 Using a third-party tool to access the remote BACnet client, verify it receives the alarm.

Step 3 Open the linked Niagara AlarmConsole and verify the alarm was received by Niagara.

Step 4 Using the third-party BACnet tool, issue an acknowledgment for the alarm.

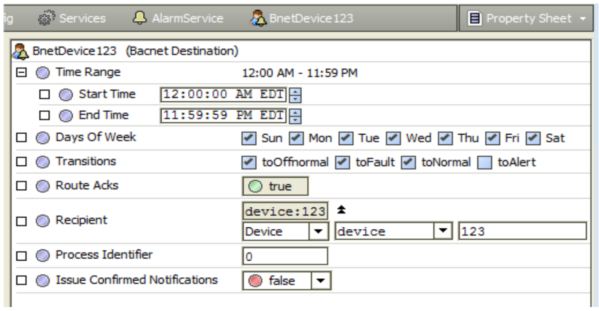
Step 5 Verify an acknowledgement is received by the station. The alarm acknowledgement will be saved in the alarm database. The alarm should also be removed from the list of unacknowledged alarms in the Niagara Alarm Console.

BacnetDestination (reference)

Available in the "Alarming" folder of the `bacnet` palette, the BacnetDestination component is a specialized alarm class recipient, for use by stations operating (and licensed) as a BACnet server. When linked as a recipient to an AlarmClass that is exported to BACnet (as a Notification Class object), each BacnetDestination specifies an entry under the "Recipient_List" for that Notification Class.

BacnetDestination properties

Figure 5-3 *BacnetDestination properties*



BnetDevice123 (Bacnet Destination)

- Time Range: 12:00 AM - 11:59 PM
  - Start Time: 12:00:00 AM EDT
  - End Time: 11:59:59 PM EDT
- Days Of Week: ☑ Sun ☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat
- Transitions: ☑ toOffnormal ☑ toFault ☑ toNormal □ toAlert
- Route Acks: true
- Recipient: device:123 — Device, device, 123
- Process Identifier: 0
- Issue Confirmed Notifications: false

Properties are as follows:

- **Time Range**
  Specifies the time range for when events can be sent, where the default is all times.
- **Days of Week**
  Specifies the days of week for when events can be sent, where the default is all days.
- **Transitions**
  Specifies which status transitions are sent, where the default is all but `toAlert`.
- **Route Acks**
  (Read only) Boolean, as to whether acks (acknowledgments) are routed to Niagara.
- **Recipient**
  Specifies the specific BACnet device to send events, using one of two definitions.
  - **Device**
    (Default) Unique BACnet Device object instance number; the default `-1` is undefined.
  - **Address**
    Provides an alternate address method, with the following separate fields:
    – **`Network Number`**: BACnet network number (1-65535); default `0` is undefined.
    – **`MAC Address`**: Data link layer MAC address of the device (default is `null`).
    – **`MAC Address Style`**: Selectable type for entered MAC address, as either: Unknown (default), Ethernet, IP, MSTP/Other
- **Process Identifier**
  (Integer) Specifies a process ID associated with all events sent from the station, meaningful to the receiving BACnet client. The default process identifier is `0`.
- **Issue Confirmed Notifications**
  Specifies whether unconfirmed or confirmed event notifications are sent from the station. The default is `false` (for unconfirmed).

Bacnet Plugin Guides

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

Summary information is provided here about the different:

- Views in module bacnet
- Views in module bacnetAws (AX-3.6 and later only)
- Views in module bacnetOws (AX-3.6 and later only)
- Views in module bacnetws (note the bacnetws module is *deprecated* starting in AX-3.6)

Views in bacnet module

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

- Bacnet Config Manager
- Bacnet Device Manager
- Bacnet Export Manager
- Bacnet File Export Manager
- Bacnet History Import Manager
- Bacnet Niagara Log Export Manager
- Bacnet Point Manager
- Bacnet Schedule Export Manager
- Bacnet Schedule Import Manager
- Bacnet Trend Multiple View
- Bacnet Virtual Property View
- Bdt Manager
- Fdt Manager

bacnet-Bacnet Config Manager

Use the Bacnet Config Manager to query a BACnet device for its objects. In this view, you can create, edit, access, and delete Bacnet *config* objects. The BacnetConfigManager is the default view for the BacnetConfigDeviceExt (`Config` container) under a BacnetDevice. The BacnetConfigManager is also the default view for any BacnetConfigFolders under the `Config` container. To view, right-click BacnetConfigDeviceExt or BacnetConfigFolder and select **Views > Bacnet Config Manager**. For more details, see "About the Config Device Ext container" on page 3-33.

bacnet-Bacnet Device Manager

Use the Bacnet Device Manager to create, edit, and access Bacnet devices. The BacnetDeviceManager is the default view on the BacnetNetwork. To view, double-click the BacnetNetwork, or right-click BacnetNetwork and select **Views > Bacnet Device Manager**. For more details, see "Bacnet Device Manager" on page 3-26.

bacnet-Bacnet Export Manager

The Bacnet Export Manager is the default view of the BacnetExportTable under the LocalBacnetDevice. You use it to export Niagara points and objects as Bacnet objects. To view, under the LocalBacnetDevice, right-click BacnetExportTable (`Export Table`) and select **Views > Bacnet Export Manager**. For more details, see "Bacnet Export Manager" on page 4-63.

bacnet-Bacnet File Export Manager

The Bacnet File Export Manager is one view of the BacnetExportTable under the LocalBacnetDevice. You use it to export files as Bacnet Files. To view, under the LocalBacnetDevice, right-click BacnetExportTable (Export Table) and select **Views > Bacnet File Export Manager**. For more details, see “Bacnet File Export Manager” on page 4-71.

bacnet-Bacnet History Import Manager

Use the Bacnet History Import Manager to import BACnet **Trend Logs** under a BacnetDevice as Niagara histories. The BacnetHistoryImportManager is the default view for the BacnetHistoryDeviceExt (Trend Logs container) under a BacnetDevice. To view, under a BacnetDevice right-click BacnetHistoryDeviceExt and select **Views > Bacnet History Import Manager**. For more details, see “About the Bacnet History Import Manager” on page 3-48.

Note starting in build 3.6.41, the Bacnet History Import Manager also lets you import BACnet **Trend Log Multiple** objects as Niagara histories. Each BacnetTrendLogMultipleImport descriptor provides a special view. For details, see “Bacnet Trend Multiple View” on page 3-51.

bacnet-Bacnet Niagara Log Export Manager

The Bacnet Niagara Log Export Manager is one view of the BacnetExportTable under the LocalBacnetDevice. You use it to export Niagara histories as Bacnet Trend Logs. To view, under the LocalBacnetDevice, right-click BacnetExportTable (Export Table) and select **Views > Bacnet Niagara Log Export Manager**. For more details, see “Bacnet Niagara Log Export Manager” on page 4-73.

bacnet-Bacnet Point Manager

Use the Bacnet Point Manager to query a BACnet device for its objects. In this view, you can create, edit, access, and delete Bacnet proxy points. The BacnetPointManager is the default view for the BacnetPointDeviceExt (Points container) under a BacnetDevice. The BacnetPointManager is also the default view for any BacnetPointFolders under the Points container. To view, right-click BacnetPointDeviceExt or BacnetPointFolder and select **Views > Bacnet Point Manager**. For more details, see “Bacnet Point Manager” on page 3-34.

bacnet-Bacnet Schedule Export Manager

Use the Bacnet Schedule Export Manager to export Niagara schedules in the station as Bacnet schedules to the target BacnetDevice. The BacnetScheduleExportManager is a view on the BacnetScheduleDeviceExt (Schedules container) under a BacnetDevice. To view, right-click BacnetScheduleDeviceExt and select **Views > Bacnet Schedule Export Manager**. For more details, see “About the Bacnet Schedule Export Manager” on page 3-44.

bacnet-Bacnet Schedule Import Manager

The Bacnet Schedule Import Manager is the default view on the BacnetScheduleDeviceExt (Schedules container) under a BacnetDevice. Use the BacnetScheduleImportManager to import Bacnet schedules residing in the target BacnetDevice (as Niagara schedules in the station). To view, right-click BacnetScheduleDeviceExt and select **Views > Bacnet Schedule Import Manager**. For more details, see “About the Bacnet Schedule Import Manager” on page 3-42.

bacnet-Bacnet Trend Multiple View

The **Bacnet Trend Multiple View** is the default view on a BacnetTrendLogMultipleImport descriptor, available in build 3.6.41 and later. To view, double-click the import descriptor in the Nav tree, or right-click the BacnetTrendLogMultipleImport component and select **Views > Bacnet Trend Multiple View**.

This view provides a tabular listing of sequenced, timestamped collections, where each collection (row) includes values for each trend item, along with status and a sequence number. For details, see “Bacnet Trend Multiple View” on page 3-51.

bacnet-Bacnet Virtual Property View

This is the default view on a BacnetVirtualProperty under a BacnetVirtualObject (AX-3.3 and later only). It simply shows the combined value and poll status of the virtual property component. For more details, see “About BacnetVirtualProperties (AX-3.3 and later)” on page 3-54.

bacnet-Bdt Manager

The BdtManager (Broadcast Distribution Table, or BDT) is the default view of the BroadcastDistributionTable under the Link container of an IpPort (BacnetIpLinkLayer). When the station is operating as a BBMD, Niagara maintains this table listing all other participating BBMDs, including their IP address and broadcast distribution masks. If necessary, this view allows you to manually edit the BDT. For related details, see "BACnet/IP and BBMDs" on page C-3.

bacnet-Fdt Manager

The Fdt Manager (Foreign Device Table, or FDT) is the default view of the ForeignDeviceTable under the Link container of an IpPort (BacnetIpLinkLayer). When the station is operating as a BBMD, this table lists other BACnet "foreign devices" that have registered with Niagara, including their IP address, time to live, and purge time. This view allows you to manually edit the FDT, if necessary, to support devices that cannot register themselves. For more details, see "Foreign Device Table" on page C-3.

Views in bacnetAws module

Summary information is provided on views specific to components in the bacnetAws module (applies only if a **BACnet AWS Supervisor**, where the bacnetAws and bacnetOws modules are used in *addition* to the bacnet module). Views specific to the bacnetAws module are as follows:

- Bacnet Aws Config Manager
- Bacnet Aws Device Manager
- Bacnet Aws History Import Manager

bacnetAws-Bacnet Aws Config Manager

Use the **Bacnet Aws Config Manager** in the **BACnet AWS Supervisor** to query a BACnet device for its objects. In this view, you can create, edit, access, and delete Bacnet *config* objects. The Bacnet Aws Config Manager is the default view for the BacnetAwsConfigDeviceExt (Config container) under a BacnetAwsDevice. The Bacnet Aws Config Manager is also the default view for any BacnetAwsConfigFolder under the Config container. To view, right-click BacnetAwsConfigDeviceExt or BacnetAwsConfigFolder and select **Views > Bacnet Aws Config Manager**. For more details, see "About the Config Device Ext container" on page 3-33.

The **Bacnet Aws Config Manager** provides all the same functions as the regular **Bacnet Config Manager** for a BacnetDevice in a BacnetNetwork, plus allows the creation or deletion of BACnet objects (if supported by the target BACnet device). For details, see "Bacnet Aws Config Manager" on page A-11.

bacnetAws-Bacnet Aws Device Manager

Use the **Bacnet Aws Device Manager** to create, edit, and access BacnetAwsDevices in a **BACnet AWS Supervisor** station. The Bacnet Aws Device Manager is the default view on the BacnetAwsNetwork. To view, double-click the BacnetAwsNetwork, or right-click the BacnetAwsNetwork and select **Views > Bacnet Aws Device Manager**.

The **Bacnet Aws Device Manager** provides all the same functions as the regular Bacnet Device Manager for a BacnetNetwork, plus offers additional buttons/functions for supervisory control of BACnet devices, as described in the BACnet specification B-AWS device profile. For more details, see "Bacnet Aws Device Manager" on page A-4.

bacnetAws-Bacnet Aws History Import Manager

Use the **Bacnet Aws History Import Manager** to import Bacnet Trend Logs under a BacnetAwsDevice as Niagara histories. The Bacnet Aws History Import Manager is the default view for the BacnetAwsHistoryDeviceExt (Trend Logs container) under a BacnetAwsDevice. To view, under a BacnetAwsDevice right-click the **Trend Logs** child and select **Views > Bacnet Aws History Import Manager**. For more details, see "Bacnet Aws History Import Manager" on page A-10.

Views in bacnetOws module

Summary information is provided on views specific to components in the bacnetOws module (applies if a **BACnet OWS Supervisor**, where the bacnetOws module is used in *addition* to the bacnet module, or to a **BACnet AWS Supervisor**, where both the bacnetOws *and* bacnetAws modules are used in addition to the bacnet module). Views specific to the bacnetOws module are as follows:

- Bacnet Ows Device Manager

bacnetOws-Bacnet Ows Device Manager

Use the **`Bacnet Ows Device Manager`** to create, edit, and access BacnetDevices in a **`BACnet OWS Supervisor`** station. The Bacnet Ows Device Manager is the default view on the BacnetOwsNetwork. To view, double-click the BacnetOwsNetwork, or right-click the network and select **`Views > Bacnet Ows Device Manager`**.

The Bacnet Ows Device Manager provides all the same functions as the regular Bacnet Device Manager for a BacnetNetwork, plus offers additional buttons/functions for supervisory control of BACnet devices, as described in the BACnet specification B-OWS device profile. For more details, see “Bacnet Ows Device Manager” on page A-15.

Views in bacnetws module

Summary information is provided on views specific to components in the `bacnetws` module (applies only if a *BACnet Supervisor*, where the bacnetws module is used in *addition* to the bacnet module).

Note: *Starting in AX-3.6, the* `bacnetws` *module is* deprecated *in favor of the* `bacnetAws` *and* `bacnetOws` *modules. There will be no further updates to the* `bacnetws` *module in point releases past AX-3.6.*

Views specific to the bacnetws module are as follows:

- Bacnet Ws Device Manager

bacnet-Bacnet Ws Device Manager

Use the BacnetWsDeviceManager to create, edit, and access BacnetWsDevices and BacnetDevices in a *BACnet Supervisor* station. The BacnetWsDeviceManager is the default view on the BacnetWsNetwork. To view, double-click the BacnetWsNetwork, or right-click the BacnetWsNetwork and select **`Views > Bacnet Ws Device Manager`**.

The BacnetWsDeviceManager provides all the same functions as the regular Bacnet Device Manager for a BacnetNetwork, plus offers additional buttons/functions for supervisory control of BACnet devices, as described in the BACnet specification B-OWS device profile. For more details, see “Bacnet Ws Device Manager” on page B-2.

CHAPTER 7

Bacnet Component Guides

These Component Guides provide summary information about:

- bacnet module components
- bacnetAws module components (AX-3.6 and later only)
- bacnetOws module components (AX-3.6 and later only)
- bacnetws module components (note the bacnetws module is *deprecated* starting in AX-3.6)

Components in bacnet module

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

- BacnetAlarmDeviceExt
- BacnetAnalogInput
- BacnetAnalogInputDescriptor
- BacnetAnalogOutput
- BacnetAnalogOutputDescriptor
- BacnetAnalogValue
- BacnetAnalogValueDescriptor
- BacnetAnalogValuePrioritizedDescriptor
- BacnetArray
- BacnetBinaryInput
- BacnetBinaryInputDescriptor
- BacnetBinaryOutput
- BacnetBinaryOutputDescriptor
- BacnetBinaryValue
- BacnetBinaryValueDescriptor
- BacnetBinaryValuePrioritizedDescriptor
- BacnetBooleanCovTrendLogExt
- BacnetBooeanIntervalTrendLogExt
- BacnetBooleanProxyExt
- BacnetBooleanScheduleDescriptor
- BacnetCalendar
- BacnetCalendarDescriptor
- BacnetClientLayer
- BacnetConfigDeviceExt
- BacnetConfigFolder
- BacnetDestination
- BacnetDevice
- BacnetDeviceFolder
- BacnetDeviceObject
- BacnetEnumCovTrendLogExt
- BacnetEnumIntervalTrendLogExt
- BacnetEnumProxyExt
- BacnetEnumScheduleDescriptor
- BacnetEthernetLinkLayer
- BacnetEventEnrollment
- BacnetEventHandler
- BacnetExportFolder

- BacnetExportTable
- BacnetFile
- BacnetFileDescriptor
- BacnetGroup
- BacnetHistoryDeviceExt
- BacnetHistoryImport
- BacnetListOf
- BacnetIpLinkLayer
- BacnetLoop
- BacnetLoopDescriptor
- BacnetMstpLinkLayer
- BacnetMultiPoll
- BacnetMultistateInput
- BacnetMultiStateInputDescriptor
- BacnetMultistateOutput
- BacnetMultiStateOutputDescriptor
- BacnetMultistateValue
- BacnetMultiStateValueDescriptor
- BacnetMultiStateValuePrioritizedDescriptor
- BacnetNetwork
- BacnetNetworkLayer
- BacnetNiagaraHistoryDescriptor
- BacnetNotificationClass
- BacnetNotificationClassDescriptor
- BacnetNumericCovTrendLogExt
- BacnetNumericIntervalTrendLogExt
- BacnetNumericScheduleDescriptor
- BacnetObject
- BacnetPoll
- BacnetPointDeviceExt
- BacnetPointFolder
- BacnetProgram
- BacnetRouterTable
- BacnetSchedule
- BacnetScheduleDeviceExt
- BacnetScheduleExport
- BacnetScheduleImportExt
- BacnetServerLayer
- BacnetStack
- BacnetStringCovTrendLogExt
- BacnetStringIntervalTrendLogExt
- BacnetStringProxyExt
- BacnetStringScheduleDescriptor
- BacnetTransportLayer
- BacnetTrendLog
- BacnetTrendLogAlarmSourceExt
- BacnetTrendLogDescriptor
- BacnetTrendLogMultiple
- BacnetTrendLogMultipleImport
- BacnetTuningPolicy
- BacnetTuningPolicyMap
- BacnetVirtualComponent
- BacnetVirtualGateway
- BacnetVirtualObject
- BacnetVirtualProperty
- BacnetWorker
- BacnetWorkerPool
- BroadcastDistributionTable
- ForeignDeviceTable
- LocalBacnetDevice
- NetworkPort
- OutOfServiceExt

bacnet-BacnetAlarmDeviceExt

Each BacnetDevice has an **Alarms** extension (BacnetAlarmDeviceExt). It is used if receiving any BACnet event notifications from this device. For related details, see "Receiving BACnet alarms" on page 5-83, including topics "Configuring to receive BACnet alarms" on page 5-84 and "BacnetAlarmDeviceExt (reference)" on page 5-85.

bacnet-BacnetAnalogInput

A Config object that represents a BACnet Analog Input object in its entirety. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetAnalogInputDescriptor

BacnetAnalogInputDescriptor exposes a numeric-type component as a BACnet Analog Input object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetAnalogOutput

A Config object that represents a BACnet Analog Output object in its entirety. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetAnalogOutputDescriptor

BacnetAnalogOutputDescriptor exposes a NumericWritable as a BACnet Analog Output object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetAnalogValue

A Config object that represents a BACnet Analog Value object in its entirety. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetAnalogValueDescriptor

BacnetAnalogValueDescriptor exposes a numeric-type component as a BACnet Analog Value object (non-commandable). You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetAnalogValuePrioritizedDescriptor

BacnetAnalogValuePrioritizedDescriptor exposes a NumericWritable as a writable (commandable) Bacnet Analog Value object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetArray

BacnetArray represents a Bacnet Array, which contains an indexed sequence of objects of a particular BACnet data type. Its elements are named "element1" through "element*N*" by default.

bacnet-BacnetBinaryInput

A Config object that represents a BACnet Binary Input object in its entirety. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetBinaryInputDescriptor

BacnetBinaryInputDescriptor exposes a boolean-type component as a BACnet Binary Input object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetBinaryOutput

A Config object that represents a BACnet Binary Output object in its entirety. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetBinaryOutputDescriptor

BacnetBinaryOutputDescriptor exposes a BooleanWritable as a BACnet Binary Output object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetBinaryValue

A Config object that represents a BACnet Binary Value object in its entirety. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetBinaryValueDescriptor

BacnetBinaryValueDescriptor exposes boolean-type component as a BACnet Binary Value object (non-commandable). You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetBinaryValuePrioritizedDescriptor

BacnetBinaryValuePrioritizedDescriptor exposes a BooleanWritable as a writable (commandable) BACnet Binary Value object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetBooleanCovTrendLogExt

An extension for collecting a Niagara history for the StatusBoolean out value of a component, using COV. Unlike when using the equivalent "standard" history extension (BooleanCOVHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see "About BacnetTrendLogExt extensions" on page 4-74.

bacnet-BacnetBooleanIntervalTrendLogExt

An extension for collecting a Niagara history for the StatusBoolean out value of a component, using a defined interval. Unlike when using the equivalent "standard" history extension (BooleanIntervalHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see "About BacnetTrendLogExt extensions" on page 4-74.

bacnet-BacnetBooleanProxyExt

This BacnetProxyExt maps a BACnet boolean value to a BooleanPoint or BooleanWritable.

bacnet-BacnetBooleanScheduleDescriptor

BacnetBooleanScheduleDescriptor exposes a Niagara BooleanSchedule as a BACnet Schedule object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetCalendar

A Config object that represents a BACnet Calendar object. Included among its properties is a Datelist property that contains calendar entries. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetCalendarDescriptor

BacnetCalendarDescriptor exposes a Niagara CalendarSchedule to BACnet as a Calendar object.You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetClientLayer

BacnetClientLayer (Client) represents the client side of the application layer of the Bacnet communications stack. Client resides under the BacnetStack (`Bacnet Comm`) of a BacnetNetwork, and has no available properties.

bacnet-BacnetConfigDeviceExt

BacnetConfigDeviceExt (Config device extension) is a frozen device extension under every BacnetDevice. It functions as the container for any Config-type objects, which represent individual BACnet objects in the device. By default, it contains a BacnetDeviceObject for the device's sole BACnet Device object.

For more details, see "About the Config Device Ext container" on page 3-33 and "About Bacnet Config objects" on page 3-34.

bacnet-BacnetConfigFolder

BacnetConfigFolder is the implementation of a folder under a BacnetConfigDeviceExt (Config). Typically, you add such folders using the **`New Folder`** button in the Bacnet Config Manager view of the Config container. Each BacnetConfigFolder has its own Bacnet Config Manager view.

bacnet-BacnetDestination

BacnetDestination represents a BACnet device that is to receive alarms from Niagara, as a type of alarm recipient. It resides with standard alarm recipients and alarm classes under the station's AlarmService. The BacnetDestination is available in the `bacnet` module "Alarming" folder.

For more details, see “Sending Alarms to BACnet” on page 5-85, including subsections “Configuring to send alarms to BACnet” on page 5-86 and “BacnetDestination (reference)” on page 5-87.

bacnet-BacnetDevice

BacnetDevice is a Niagara representation of a remote BACnet device. Each BacnetDevice resides under the station’s BacnetNetwork. Each BacnetDevice contains a full complement of device extensions (containers), including Points, Schedules, and Trend Logs (Histories), for modeling data (from that device) in the station. For more details, see “Bacnet Device components” on page 3-31, “About Bacnet Device’s Schedules” on page 3-42, and “About Bacnet Trend Logs (Histories)” on page 3-47.

bacnet-BacnetDeviceFolder

BacnetDeviceFolder is the Bacnet implementation of a folder under a BacnetNetwork. Typically, you add such folders using the **`New Folder`** button in the Bacnet Device Manager view of the BacnetNetwork. Each BacnetDeviceFolder has its own Bacnet Point Manager view. The BacnetDeviceFolder is also available in the `bacnet` palette.

bacnet-BacnetDeviceObject

A Config object that contains all the properties of a BACnet Device object as defined by the BACnet specification. Properties such as the device address, which are not BACnet Device properties, but which are associated with this device, are contained directly in this object. By default, the BacnetDeviceObject is available in the BacnetConfigDeviceExt.

See “About the Config Device Ext container” on page 3-33 for related details.

bacnet-BacnetEnumCovTrendLogExt

An extension for collecting a Niagara history for the StatusEnum out value of a component, using COV. Unlike when using the equivalent “standard” history extension (EnumCovHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see “About BacnetTrendLogExt extensions” on page 4-74.

bacnet-BacnetEnumIntervalTrendLogExt

An extension for collecting a Niagara history for the StatusEnum out value of a component, using a defined interval. Unlike when using the equivalent “standard” history extension (EnumIntervalHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see “About BacnetTrendLogExt extensions” on page 4-74.

bacnet-BacnetEnumProxyExt

This BacnetProxyExt handles multistate values in a BACnet device, and maps enumerated values in the device to the parent EnumPoint or EnumWritable.

bacnet-BacnetEnumScheduleDescriptor

BacnetEnumScheduleDescriptor exposes a Niagara EnumSchedule to BACnet as a Schedule object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see “About export descriptors” on page 4-61.

bacnet-BacnetEthernetLinkLayer

Tridium Bacnet Ethernet virtual link layer implementation. The BacnetEthernetLinkLayer is available under the `BACnetComm`, `NetworkPorts` folder of the bacnet palette.

bacnet-BacnetEventEnrollment

A Config object that represents a BACnet Event Enrollment object. Properties of this object describes an event that may be an error or alarm condition, referencing source object and property data, and specifies notification class and device recipients. For related details, see “About Bacnet Config objects” on page 3-34.

bacnet-BacnetEventHandler

BacnetEventHandler processes event notification messages, and the acknowledgement of those messages, including ConfirmedEventNotification, UnconfirmedEventNotification, and AcknowledgeAlarm. It is a frozen container slot of the BacnetServerLayer (Server under Bacnet Comm).

bacnet-BacnetExportFolder

BacnetExportFolder is the implementation of a folder to use under the LocalBacnetDevice’s BacnetExportTable to organize Bacnet server descriptors. Typically, you add such folders using the **`New Folder`** button in the different manager views of the BacnetExportTable, namely the Bacnet Export Manager, Bacnet File Export Manager, and Bacnet Niagara Log Export Manager.

Each BacnetExportFolder provides the same set of export manager views. The BacnetExportFolder is also available in the bacnet palette.

bacnet-BacnetExportTable

The Export Table is a frozen slot under the LocalBacnetDevice, and contains all the server (export) descriptors that expose station objects as BACnet objects. You use the different export manager views of the Export Table to export station objects. For more details, see "Bacnet server configuration overview" on page 4-60.

bacnet-BacnetFile

A Config object that represents a BACnet File object. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetFileDescriptor

BacnetFileDescriptor exposes a file under the station's folder as a BACnet File object. You use the Bacnet File Export Manager view of the BacnetExportTable to add, edit, delete, and access exported files. For more details, see "About file descriptors" on page 4-61.

All files in Niagara are accessed using the STREAM_ACCESS method.

bacnet-BacnetGroup

A Config object that represents a BACnet Group object. A Group object provides access to multiple properties of multiple objects in a read single operation. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetHistoryDeviceExt

BacnetHistoryDeviceExt (**Trend Logs**) is a frozen device extension under every BacnetDevice, and the container for BacnetHistoryImport components. The default view is the Bacnet History Import Manager, used to import data from Trend Log objects in the device, into the station as Niagara histories. For more details, see "About Bacnet Trend Logs (Histories)" on page 3-47.

bacnet-BacnetHistoryImport

BacnetHistoryImport defines the archive action to retrieve data from a BACnet Trend Log object into a Niagara history created by the addition of this component. You add BacnetHistoryImports (import Trend Logs) using the Bacnet History Import Manager view of the parent BacnetHistoryDeviceExt. For more details, see "BACnet Trend Log import notes" on page 3-48 and "BacnetHistoryImport properties" on page 3-50.

bacnet-BacnetIpLinkLayer

The Bacnet IP virtual link layer implementation, it appears as the Link node under an IpPort NetworkPort child of the BacnetComm, BacnetNetworkLayer. This Link container contains properties specifying a number of items, including Ethernet adapter, IP address, UDP port, IP device type, BBMD address, and registration lifetime.

This Link node is also the parent container for the BroadcastDistributionTable and ForeignDeviceTable. For more details, see "About Bacnet Comm: Network: Ip Port" on page 3-14.

bacnet-BacnetListOf

BacnetListOf represents a Bacnet ListOf sequence, which contains a non-indexed sequence of objects of a particular BACnet data type.

bacnet-BacnetLoop

A Config object that represents a BACnet Loop object. A Loop object provides a standardized "control loop" implementation. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetLoopDescriptor

BacnetLoopDescriptor exposes a kitControl LoopPoint to BACnet as a Loop object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-MstpLinkLayer

Tridium Bacnet MS/TP virtual link layer implementation. The BacnetMstpLinkLayer is available under the BACnetComm, NetworkPorts folder of the bacnet palette.

bacnet-BacnetMultiPoll

BacnetMultiPoll is used to configure and manage a group of Bacnet proxy points to be polled. The BacnetMultiPoll provides a flexible polling algorthim based on four "buckets."

Each pollable proxy point assigned to a BacnetMultiPoll is configured in one of three buckets: fast, normal, or slow. In addition there is a fourth bucket called the “dibs stack.” Whenever a point is subscribed it immediately gets “first dibs” and goes on the top of the dibs stack. The poll scheduler always polls the dibs before doing anything else. The dibs stack is polled last-in, first-out (LIFO). As long as entries are in the dibs stack they are polled as fast as possible with no artificial delays.

When the dibs stack is empty the scheduler attempts to poll the components in each bucket using an algorthim designed to create uniform network traffic. For example if the fast rate is configured to 5000ms and there are 5 components currently subscribed in the fast bucket, then the scheduler will attempt to poll each component with a second to delay.

Every ten seconds the poll scheduler rechecks the buckets for configuration changes. So if a point’s configuration is changed from slow to fast, it takes at most ten seconds for the change to take effect. Statistics are also updated every ten seconds. Statistics may be manually reset using the resetStatistics action.

For additional details, see “About poll components” in the *Drivers Guide*.

bacnet-BacnetMultistateInput

A Config object that represents a BACnet Multi-state Input object. For related details, see “About Bacnet Config objects” on page 3-34.

bacnet-BacnetMultiStateInputDescriptor

BacnetMultistateInputDescriptor exposes an EnumPoint as a BACnet Multi-state Input object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see “About export descriptors” on page 4-61

bacnet-BacnetMultistateOutput

A Config object that represents a BACnet Multi-state Output object. For related details, see “About Bacnet Config objects” on page 3-34.

bacnet-BacnetMultiStateOutputDescriptor

BacnetMultiStateOutputDescriptor exposes an EnumWritable as BACnet Multi-state Output object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see “About export descriptors” on page 4-61

bacnet-BacnetMultistateValue

A Config object that represents a BACnet Multi-state Value object. For related details, see “About Bacnet Config objects” on page 3-34.

bacnet-BacnetMultiStateValueDescriptor

BacnetMultiStateValueDescriptor exposes an EnumPoint as BACnet Multi-state Value object (non-commandable). You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see “About export descriptors” on page 4-61.

bacnet-BacnetMultiStateValuePrioritizedDescriptor

BacnetMultiStateValuePrioritizedDescriptor exposes an EnumWritable as a writable (commandable) BACnet Multi-state Value object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see “About export descriptors” on page 4-61.

bacnet-BacnetNetwork

BacnetNetwork is the base container for all Bacnet components in the station. In addition to being the network container for BacnetDevices and their child data objects (Bacnet proxy points), it contains the station’s BACnet communications protocol stack (Bacnet Comm), plus a Local Bacnet Device, which configures the station’s representation as a BACnet device.

Note: *Only one BacnetNetwork component is supported in any station, regardless of how many different BACnet link-layer protocols are being used.*

As with other NiagaraAX driver networks, the BacnetNetwork should reside under the station’s Drivers container. For general information, see “Niagara Bacnet Client Concepts” on page 3-11.

The default view of the BacnetNetwork is the Bacnet Device Manager.

bacnet-BacnetNetworkLayer

BacnetNetworkLayer represents the generic superclass for all network layer implementations. In practical terms, it is the container for all network ports, plus the BacnetRouterTable. The BacnetNetworkLayer resides under the BacnetStack (`Bacnet Comm`) of a BacnetNetwork or BacnetWsNetwork.

bacnet-BacnetNiagaraHistoryDescriptor

BacnetNiagaraHistoryDescriptor is the server log descriptor that exposes a standard Niagara history to BACnet as a Trend Log object, supporting only "by time" or "by index" requests for the trend log data. You use the Bacnet Niagara Log Export Manager view of the BacnetExportTable to add, edit, delete, and access these descriptors.

Alternatively, you can add one of several BacnetTrendLogExt (specialized history extension) to a point—this creates another type of log descriptor. For more details, see "About log descriptors" on page 4-61.

bacnet-BacnetNotificationClass

A Config object that represents a BACnet Notification Class object. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetNotificationClassDescriptor

This export descriptor allows a station's AlarmClass to be exposed to BACnet as a Notification Class object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetNumericCovTrendLogExt

An extension for collecting a Niagara history for the StatusNumeric out value of a component, using COV. Unlike when using the equivalent "standard" history extension (NumericCovHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see "About BacnetTrendLogExt extensions" on page 4-74.

bacnet-BacnetNumericIntervalTrendLogExt

An extension for collecting a Niagara history for the StatusNumeric out value of a component, using a defined interval. Unlike when using the equivalent "standard" history extension (NumericIntervalHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see "About BacnetTrendLogExt extensions" on page 4-74.

bacnet-BacnetNumericScheduleDescriptor

BacnetNumericScheduleDescriptor exposes a Niagara NumericSchedule to BACnet as a Schedule object. In the `bacnet` palette, it appears under the `Server` folder as `ServerNumSchedDesc`. However, you typically use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetObject

A general purpose Config object that provides properties of a BACnet object type that is not included in the "Config" folder of the `bacnet` palette, or if a BACnet OWS Supervisor or BACnet AWS Supervisor, not included in the "Config" folders of the `bacnetOws` and `bacnetAws` palette.

See "About the Config Device Ext container" on page 3-33 for related details.

bacnet-BacnetPoll

BacnetPoll is used to configure and manage a group of Bacnet proxy points to be polled. The BacnetPoll provides a flexible polling algorthim based on four "buckets."

Each pollable proxy point assigned to a BacnetPoll is configured in one of three buckets: fast, normal, or slow. In addition there is a fourth bucket called the "dibs stack." Whenever a point is subscribed it immediately gets "first dibs" and goes on the top of the dibs stack. The poll scheduler always polls the dibs before doing anything else. The dibs stack is polled last-in, first-out (LIFO). As long as entries are in the dibs stack they are polled as fast as possible with no artificial delays.

When the dibs stack is empty the scheduler attempts to poll the components in each bucket using an algorthim designed to create uniform network traffic. For example if the fast rate is configured to 5000ms and there are 5 components currently subscribed in the fast bucket, then the scheduler will attempt to poll each component with a second to delay.

Every ten seconds the poll scheduler rechecks the buckets for configuration changes. So if a point's configuration is changed from slow to fast, it takes at most ten seconds for the change to take effect. Statistics are also updated every ten seconds. Statistics may be manually reset using the resetStatistics action.

For additional details, see "About poll components" in the *Drivers Guide*.

bacnet-BacnetPointDeviceExt

BacnetPointDeviceExt (**`Points`**) is the Bacnet implementation of PointDeviceExt, a frozen device extension under every BacnetDevice. Its primary view is the BacnetPointManager. For more details, see "Bacnet Point Manager" on page 3-34.

bacnet-BacnetPointFolder

BacnetPointFolder is the Bacnet implementation of a folder under a BacnetDevice's Points container (BacnetPointDeviceExt). You add such folders using the **New Folder** button in the Bacnet Point Manager view of the Points component. Each BacnetPointFolder has its own Bacnet Point Manager view. The BacnetPointFolder is also available in the bacnet palette.

bacnet-BacnetProgram

A Config object that represents a BACnet Program object. A Program object lets a program running in the device to be controlled, and reports the program's present status. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetRouterTable

BacnetRouterTable stores the table of known routers to BACnet networks. In the station, the BacnetRouterTable is under the BacnetNetwork's **Bacnet Comm**, **Network** component. For more details, see "About Bacnet Comm: Network: Router Table" on page 3-14.

bacnet-BacnetSchedule

A Config object that represents a BACnet Schedule object. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetScheduleDeviceExt

BacnetScheduleDeviceExt (**Schedules**) is the container for BACnet schedules under a BacnetDevice. Its default view is the Bacnet Schedule Import Manager. Another view is the Bacnet Schedule Export Manager, to allow writing Niagara schedule configuration into a device's BACnet schedule. For details, see "About Bacnet Device's Schedules" on page 3-42.

bacnet-BacnetScheduleExport

BacnetScheduleExport is used to export the configuration of a Niagara schedule or calendar into a specific Schedule or Calendar object in a BACnet device. The schedule in the remote BACnet device is the "subordinate," and Niagara will periodically synchronize the values by writing its local values to the remote device using WriteProperty service requests. You add BacnetScheduleExports using the Bacnet Schedule Export Manager view. For details, see "About the Bacnet Schedule Export Manager" on page 3-44.

bacnet-BacnetScheduleImportExt

BacnetScheduleImportExt is a child extension of a schedule that is being imported from a BACnet device. The schedule in the remote BACnet device is the "master," and Niagara will periodically synchronize its local copy by reading the appropriate values. You add BacnetScheduleImportExts using the Bacnet Schedule Import Manager view. For details, see "About the Bacnet Schedule Import Manager" on page 3-42.

bacnet-BacnetServerLayer

BacnetServerLayer (Server) represents the server side of the application layer of the BACnet communications stack. It is the container for the BacnetWorker and BacnetEventHandler. BacnetServerLayer resides under the BacnetStack (Bacnet Comm) of a BacnetNetwork or BacnetWsNetwork.

See "About Bacnet Comm: Client, Server, and Transport" on page 3-16 for related details.

bacnet-BacnetStack

BacnetStack (**Bacnet Comm**) provides the protocol stack for BACnet communications for a BacnetNetwork, and is a frozen child container slot of the network. In turn, it has child frozen container slots BacnetClientLayer, BacnetServerLayer, BacnetTransportLayer, and BacnetNetworkLayer. For more details, see "About Bacnet Comm" on page 3-12.

bacnet-BacnetStringCovTrendLogExt

An extension for collecting a Niagara history for the StatusString out value of a component, using COV. Unlike when using the equivalent "standard" history extension (StringIntervalHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see "About BacnetTrendLogExt extensions" on page 4-74.

bacnet-BacnetStringIntervalTrendLogExt

An extension for collecting a Niagara history for the StatusString out value of a component, using a defined interval. Unlike when using the equivalent "standard" history extension (StringIntervalHistoryExt), the history created by this extension can be exported as a fully-compliant BACnet Trend Log object. For more details, see "About BacnetTrendLogExt extensions" on page 4-74.

bacnet-BacnetStringProxyExt

The BacnetProxyExt that handles the point configuration of a point of generic type in a BACnet device. It is the default for types NULL, OCTET_STRING, CHARACTER_STRING, BIT_STRING, DATE, TIME, and OBJECT_IDENTIFIER.

bacnet-BacnetStringScheduleDescriptor

BacnetStringScheduleDescriptor exposes a Niagara StringSchedule to BACnet as a Schedule object. You use the Bacnet Export Manager view of the BacnetExportTable to add, edit, delete, and access exported components. For more details, see "About export descriptors" on page 4-61.

bacnet-BacnetTransportLayer

BacnetTransportLayer (Transport) is the Tridium Transport Layer implementation. Transport resides under the BacnetStack (`Bacnet Comm`) of a BacnetNetwork or BacnetWsNetwork, and contains properties that typically do not require adjustment.

bacnet-BacnetTrendLog

A Config object that provides the configuration of a Trend Log object in a BACnet device (to access its log data, you must import that Trend Log under the **`Trend Logs`** extension of that Bacnet-Device). For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetTrendLogAlarmSourceExt

BacnetTrendLogAlarmSourceExt defines the intrinsic alarming/notification for a server-side Trend Log object exposed to BACnet. If needed, you paste it as a child of any of the BacnetTrendLogExt type extensions under the source Niagara component. It is found in the `bacnet` palette, under the `Trending`, `BACnetLogExtensions` folder.

bacnet-BacnetTrendLogDescriptor

BacnetTrendLogDescriptor is the server descriptor that exposes a Niagara history created with a BacnetTrendLogExt as a BACnet Trend Log object. These descriptors are automatically created in the root of the BacnetExportTable when you copy one of the various BacnetTrendLogExt extensions into a component. For related details, see "About BacnetTrendLogExt extensions" on page 4-74.

Note: *Unlike a history exported with a BacnetNiagaraHistoryDescriptor (created by a "standard" Niagara history extension), a history exported by a BacnetTrendLogDescriptor appears as a fully BACnet-compliant Trend Log object. This means that it supports "by sequence number" requests in addition to "by time" requests from external BACnet devices.*

bacnet-BacnetTrendLogMultiple

A Config object (**`TLM`**) that represents a BACnet Trend Log Multiple object (available starting in build 3.6.41 or later). A Trend Log Multiple object monitors and records values of one or more properties of one or more referenced objects, either in the same device as the Trend Log Multiple object or in an external device. For related details, see "About Bacnet Config objects" on page 3-34.

bacnet-BacnetTrendLogMultipleImport

(Available starting in build 3.6.41 or later.) BacnetTrendLogMultipleImport defines the archive action to retrieve data from a BACnet **`Trend Log Multiple`** object into a Niagara history created by the addition of this component. You add BacnetTrendLogMultipleImports (import Trend Log Multiples) using the Bacnet History Import Manager view of the device's parent BacnetHistoryDeviceExt (Trend Logs extension). This import descriptor also has its own special "Bacnet Trend Multiple View". For more details, see "About the Bacnet History Import Manager" on page 3-48, "BACnet Trend Log import notes" on page 3-48, and "Bacnet Trend Multiple View" on page 3-51.

bacnet-BacnetTuningPolicy

A tuning policy for the BacnetNetwork, with both standard NiagaraAX tuning policy properties, and additional properties related to client-side usage of the BACnet Subscribe_COV service.

For an explanation of driver tuning policies, refer to "About Tuning Policies" in the *Drivers Guide*. For details specific to Bacnet, see "Bacnet Tuning Policy notes" on page 3-24.

bacnet-BacnetTuningPolicyMap

A container for one or more BacnetTuningPolicy(ies). Typically, you create multiple tuning policies and assign Bacnet proxy points as needed, based upon the different BacnetComm, Network, port used (Ip Port, Ethernet Port, Mstp Port).

Note: *Each Bacnet network port has its own PollService, including 3 different poll rates (Fast, Normal, Slow). This varies from other driver networks, where a single PollService is used.*

For an explanation of driver tuning policies, refer to “About Tuning Policies” in the *Drivers Guide*. For details specific to Bacnet, see “Bacnet Tuning Policy notes” on page 3-24.

bacnet-BacnetVirtualComponent

A BacnetVirtualComponent is the AX-3.2 BACnet driver implementation of a Baja “virtual point”, where each BacnetVirtualComponent represents a BACnet object. You find these by expanding the property sheet of the BacnetVirtualGateway under each BacnetDevice and BacnetWsDevice. The property sheet of each BacnetVirtualComponent provides dynamic property values of that object.

Virtual components reside in the station’s “virtual component space,” and are not persisted in the station’s database in its component space (Config), like regular components. For a general explanation about Baja virtual components, refer to “About virtual component spaces” in the *Drivers Guide*.

Note: *AX-3.2 or higher is required for virtual components. Starting in AX-3.3, the BACnet design for virtual points changed such that BacnetVirtualComponents are replaced by BacnetVirtualObject components, each with BacnetVirtualProperty components.*

BacnetVirtualComponents can provide basic monitor access in Px views, or be used for one-off” read/ write access to configuration properties. For details, see “About Bacnet virtual points” on page 3-52.

bacnet-BacnetVirtualGateway

(AX-3.2 or later) The BacnetVirtualGateway is the Bacnet driver implementation of the Baja Virtual Gateway. A virtual gateway is a component that resides under the station’s component space (`Config`), and acts as a gateway to the station’s “virtual component space.” Note other object spaces are `Files` and `History`. For a general explanation about Baja virtual components, refer to “About virtual component spaces” in the *Drivers Guide*.

Starting in AX-3.2, each BacnetDevice and BacnetWsDevice has its own BacnetVirtualGateway, at the same level as its device extensions (Points, Schedules, and so on). Accessing components under the gateway dynamically adds them as “virtual points” while they are subscribed, but they exist only in memory (are not persisted in the station database like proxy points). When virtual points become unsubscribed, they are automatically cleaned up from the station database.

The “virtual point” level under a BacnetVirtualGateway varies as follows:

- In AX-3.3 and later, expanding a BacnetVirtualGateway results in that device’s BACnet objects to be listed underneath. Each is a BacnetVirtualObject that contains a number of BacnetVirtualProperty components to represent its properties. Thus, virtual points are at the “object, property level.”
- In AX-3.2 (only), expanding a BacnetVirtualGateway results in that device’s BACnet objects to be listed underneath as BacnetVirtualComponent components, where each has a property sheet that shows all the values for the BACnet object’s properties. Thus, virtual points are at the “object level.”

For more details, see “About Bacnet virtual points” on page 3-52.

bacnet-BacnetVirtualObject

A BacnetVirtualObject is the AX-3.3 and later Bacnet driver implementation of the *container* for Baja “virtual points”, where each BacnetVirtualObject represents a BACnet object. Find them by expanding the BacnetVirtualGateway under each BacnetDevice and BacnetWsDevice. Each BacnetVirtualObject contains some number of **`BacnetVirtualProperty`** child components, which represent the properties of that BACnet object. These BacnetVirtualProperty components act as “virtual points” to access data in the device.

Virtual components reside in the station’s “virtual component space,” and are not persisted in the station’s database in its component space (Config), like regular components. Usage of Bacnet virtual points is anticipated for monitor access in Px views, or for “one-off” read/write access to configuration properties. For more details, see “About Bacnet virtual points” on page 3-52.

bacnet-BacnetVirtualProperty

A BacnetVirtualPropery is the AX-3.3 and later Bacnet driver implementation of a property under a BacnetVirtualObject. Essentially, each represents an available “virtual point” under that BacnetDevice, via its parent BacnetVirtualObject and its parent BacnetVirtualGateway.

Usage of Bacnet virtual points is anticipated for monitor access in Px views, or for “one-off” read/write access to configuration properties. For more details, see “About Bacnet virtual points” on page 3-52.

bacnet-BacnetWorker

BacnetWorker (“Worker”) manages the queue and thread processing for reads and writes to BACnet. There are *at least* three frozen Worker slots under a BacnetNetwork. Sometimes, default configuration of Workers provides good BACnet driver performance. However, in some scenarios configuration adjustments may be needed. Note in AX-3.8, more Worker-related options are available.

For more details, see “Bacnet Workers” on page 3-19.

bacnet-BacnetWorkerPool

(AX-3.8) BacnetWorkerPool (**`WorkerPool`**) can be copied from `bacnet` palette and added as a child of a “Worker” component, including those (hidden “Worker” slots) of the BacnetNetwork, or a visible “Worker” slot of the BACnet server layer (**`Server`** under **`Bacnet Comm`**). In some cases, particularly in a large or busy system, the addition of a “Worker Pool” component can improve performance and possibly stop Bacnet “queue full” standard output messages.

For more details, see “Bacnet Workers”, including subtopic “Worker pool expansion” on page 3-20.

bacnet-BroadcastDistributionTable

The BroadcastDistributionTable is a frozen child slot of the BacnetIpLinkLayer (`Link`) node under an BacnetIpLinkLayer child of the `BacnetComm`, BacnetNetworkLayer. If the station is operating as a BBMD (BACnet Broadcast Management Device), this BDT table contains the IP addresses and broadcast distribution masks of all other known participating BBMDs.

Its default **`Bdt Manager`** view allows manually entering known BBMDs, if needed. All changes to the BDT are automatically propagated to the other BBMDs in the table. See “About BBMDs” on page C-3.

bacnet-ForeignDeviceTable

The ForeignDeviceTable is a frozen child slot of the BacnetIpLinkLayer (`Link`) node under an BacnetIpLinkLayer child of the `BacnetComm`, BacnetNetworkLayer. If the station is operating as a BACnet foreign device, this FDT table contains information about other BAcnet foreign devices registered with the station. Typically, this table is automatically populated, and devices are purged after the requested lifetime expires.

Its default **`Fdt Manager`** view allows manually entering foreign devices, if they are incapable of self-registration. Manually entered foreign devices are not purged from the FDT table. See “Foreign Device Table” on page C-3.

bacnet-LocalBacnetDevice

LocalBacnetDevice (**`Local Device`**) is the representation of Niagara as a BACnet device on the BACnet internetwork, and is a frozen container under the BacnetNetwork. Its child BacnetExportTable container provides server-side functions to export station objects as BACnet objects, and service BACnet client requests.

For related details, see “Bacnet Local Device” on page 3-21 and “Bacnet server configuration overview” on page 4-60.

bacnet-NetworkPort

Handles sending and receiving messages to and from the BACnet link layer. The different types of Network Ports are IpPort, EthernetPort, and MstpPort. For more details, see “About Bacnet Comm: Network” on page 3-13.

bacnet-OutOfServiceExt

Starting in AX-3.8 and “update releases” of AX-3.7 and AX-3.6, each NiagaraAX control point exported to BACnet is automatically given an “outOfServiceExt”. The extension has two properties:

- Out Of Service — (Boolean), by default is false. If set to true, the point’s current value is written to the Present Value property (below). Often, a remote BACnet client sets (or clears) this “Out of Service” state.
- Present Value — While “Out Of Service” is set to true, reflects the last written “Present Value”.

Components in bacnetAws module

- *Starting in* AX-3.6, *the* `bacnetAws` *module is used for a* **`BACnet AWS Supervisor`**, *along with the* `bacnetOws` *and (core)* `bacnet` *modules.* Therefore, in addition to the components listed below, most of the core bacnet module components and a couple of the bacnetOws module components are also available.
- A **`BACnet OWS Supervisor`** uses only the `bacnetOws` and (core) `bacnet` modules. Both of these Supervisor products were tested and certified as B-AWS or B-OWS respectively, by the BTL (BACnet Testing Laboratories).
- The `bacnetws` module for a **`BACnet Supervisor`** (without BTL certification) is deprecated in favor of the newer `bacnetOws` and `bacnetAws` modules.

Summary information is provided on components specific to the `bacnetAws` module, listed in alphabetical order.

- BacnetAccessDoor
- BacnetAccumulator
- BacnetAwsConfigDeviceExt
- BacnetAwsConfigFolder
- BacnetAwsDevice
- BacnetAwsDeviceFolder
- BacnetAwsHistoryDeviceExt
- BacnetAwsNetwork
- BacnetEventLog
- BacnetEventLogImport
- BacnetLoadControl
- BacnetPulseConverter
- BacnetStructuredView
- LocalBacnetAwsDevice

bacnetAws-BacnetAccessDoor

A Config object (**`Door`**) that represents a BACnet Access Door object. The Access Door object represents the physical characteristics of an access-controlled door. The object is available in the `bacnetAws` palette, in the “Config” folder. For related details, see “About Bacnet Config objects” on page 3-34.

bacnetAws-BacnetAccumulator

A Config object (**`ACC`**) that represents a BACnet Accumulator object. The Accumulator object represents the characteristics of a meter that indicates measurements by counting pulses. The object is available in the `bacnetAws` palette, in the “Config” folder. For related details, see “About Bacnet Config objects” on page 3-34.

bacnetAws-BacnetAwsConfigDeviceExt

BacnetAwsConfigDeviceExt (Config device extension) is a frozen device extension under every BacnetAwsDevice. It functions as the container for any config-type objects, which represent individual BACnet objects in the device. By default, it contains a BacnetDeviceObject for the device’s sole BACnet Device object.

For more details, see “About the Config Device Ext container” on page 3-33.

bacnetAws-BacnetAwsConfigFolder

BacnetAwsConfigFolder is the implementation of a folder under a BacnetAwsConfigDeviceExt (Config) of a BacnetAwsDevice. Typically, you add such folders using the **`New Folder`** button in the Bacnet Aws Config Manager view of the Config container. Each BacnetAwsConfigFolder has its own Bacnet Aws Config Manager view.

bacnetAws-BacnetAwsDevice

BacnetAwsDevice is a Niagara representation of a remote BACnet device in the BacnetAwsNetwork of a **`BACnet AWS Supervisor`** station. Each BacnetAwsDevice includes all the same properties and device extensions (containers) of Points, Schedules, and Trend Logs (Histories) as does a regular BacnetDevice component, for modeling data (from that device) in the station.

For related details, see “Bacnet Device components” on page 3-31, “About Bacnet Device’s Schedules” on page 3-42, and “About Bacnet Trend Logs (Histories)” on page 3-47. A BacnetAwsDevice has slightly different **`Trend Logs`** and **`Config`** device extensions than a regular BacnetDevice component, but is otherwise very similar. See “Bacnet Aws History Import Manager” on page A-10 and “Bacnet Aws Config Manager” on page A-11 for details unique to a BacnetAwsDevice component.

bacnetAws-BacnetAwsDeviceFolder

BacnetAwsDeviceFolder is the Bacnet Aws implementation of a folder under a BacnetAwsNetwork. Typically, you add such folders using the **`New Folder`** button in the Bacnet Aws Device Manager view of the BacnetAwsNetwork. Each BacnetAwsDeviceFolder has its own Bacnet Aws Device Manager view. The BacnetAwsDeviceFolder is also available in the `bacnetAws` palette.

bacnetAws-BacnetAwsHistoryDeviceExt

BacnetAwsHistoryDeviceExt (**`Trend Logs`**) is a frozen device extension under every BacnetAwsDevice in a BacnetAwsNetwork, and is the container for various log import components (BacnetHistoryImport, BacnetEventLogImport, BacnetTrendLogMultipleImport). The default view is the **`Bacnet Aws History Import Manager`**, used to import data from BACnet log objects in the device, as Niagara histories. For more details, see “About Bacnet Trend Logs (Histories)” on page 3-47, and “Bacnet Aws History Import Manager” on page A-10.

bacnetAws-BacnetAwsNetwork

BacnetAwsNetwork is the base container for all Bacnet components in a **BACnet AWS Supervisor** station (used *instead* of a BacnetNetwork). In addition to being the network container for BacnetAwsDevice components and their child data objects (e.g. Bacnet proxy points), it contains the station's BACnet communications protocol stack (Bacnet Comm), plus a LocalBacnetAwsDevice, which configures the station's representation as a BACnet device.

Note: *Only one BacnetAwsNetwork/BacnetOwsNetwork/BacnetNetwork component is supported in any station, regardless of how many different BACnet link-layer protocols are being used.*

As with other NiagaraAX driver networks, the BacnetAwsNetwork should reside under the station's Drivers container. For general information, see "Niagara Bacnet Client Concepts" on page 3-11.

The default view of the BacnetAwsNetwork is the **Bacnet Aws Device Manager**. For more details, see "BACnet AWS Supervisor features" on page A-4 and "Bacnet Aws Device Manager" on page A-4.

bacnetAws-BacnetCommand

A Config object (**CMD**) that represents a BACnet Command object. A Command object writes multiple values to multiple objects in multiple devices to complete a specific purpose. For related details, see "About Bacnet Config objects" on page 3-34.

bacnetAws-BacnetEventLog

A Config object (**ELOG**) that represents a BACnet Event Log object. An Event Log object records BACnet event notifications with timestamps and other pertinent data for subsequent retrieval. For related details, see "About Bacnet Config objects" on page 3-34.

bacnetAws-BacnetEventLogImport

BacnetEventLogImport defines the archive action to retrieve data from a BACnet Event Log object into a Niagara history created by the addition of this component. You add BacnetEventLogImports (import Event Logs) using the Bacnet Aws History Import Manager view of the parent BacnetAwsHistoryDeviceExt. For more details, see "BACnet Trend Log import notes" on page 3-48 and "Bacnet Aws History Import Manager" on page A-10.

bacnetAws-BacnetLoadControl

A Config object (**LCO**) that represents a BACnet Load Control object. A Load Control object allows external control over the shedding of a load that it controls. For related details, see "About Bacnet Config objects" on page 3-34.

bacnetAws-BacnetPulseConverter

A Config object (**PC**) that represents a BACnet Pulse Converter object. A Pulse Converter object monitors counts or pulses that typically represent a metered process, such as water or energy usage. For related details, see "About Bacnet Config objects" on page 3-34.

bacnetAws-BacnetStructuredView

A Config object (**SVO**) that represents a BACnet Structured View object. A Structured View object acts as a container for references to other BACnet objects, possibly including other Structured View objects. It provides an organizational (and potentially hierarchical) interface to BACnet objects in a system. For related details, see "About Bacnet Config objects" on page 3-34.

bacnetAws-LocalBacnetAwsDevice

LocalBacnetAwsDevice (**Local Device**) is the representation of Niagara as a BACnet device on the BACnet internetwork for a **BACnet AWS Supervisor**, and is a frozen container of a BacnetAwsNetwork. The LocalBacnetAwsDevice has all the same properties and components of a regular LocalBacnetDevice, including a child BacnetExportTable container.

Note: *At the time of this document update, a* **BACnet AWS Supervisor** *is not licensed for server operation, as those functions were not part of the BTL certification testing for B-AWS compliance. Therefore, the* **ExportTable** *under a LocalBacnetAwsDevice is not typically used.*

For related details, see "Bacnet Local Device" on page 3-21 and also "BACnet OWS and AWS Supervisors overview" on page A-2.

Components in bacnetOws module

- Starting in AX-3.6, the `bacnetOws` module is used for a **BACnet OWS Supervisor**, along with the (core) `bacnet` module. Therefore, in addition to the components listed below, most of the core bacnet module components are also available.
- A **BACnet AWS Supervisor** also uses the `bacnetOws` module, along with the `bacnetAws` and

(core) `bacnet` modules. Both Supervisor products were tested and certified as B-OWS or B-AWS respectively, by the BTL (BACnet Testing Laboratories).

- The `bacnetws` module for a **BACnet Supervisor** (without BTL certification) is deprecated in favor of the newer `bacnetOws` and `bacnetAws` modules.

Summary information is provided on components specific to the `bacnetOws` module, listed in alphabetical order.

- BacnetOwsDeviceFolder
- BacnetOwsNetwork
- LocalBacnetOwsDevice

bacnetOws-BacnetOwsDeviceFolder

BacnetOwsDeviceFolder is the device folder implementation under a BacnetOwsNetwork. Typically, you add such folders using the **New Folder** button in the Bacnet Ows Device Manager view of the BacnetOwsNetwork. Each BacnetOwsDeviceFolder has its own Bacnet Ows Device Manager view. The BacnetOwsDeviceFolder is also available in the `bacnetOws` palette.

bacnetOws-BacnetOwsNetwork

BacnetOwsNetwork is the base container for all Bacnet components in a **BACnet OWS Supervisor** station (used *instead* of a BacnetNetwork). In addition to being the network container for bacnetOws module components and their child data objects (e.g. Bacnet proxy points), it contains the station's BACnet communications protocol stack (Bacnet Comm), plus a LocalBacnetOwsDevice, which configures the station's representation as a BACnet device.

Note: *Only one BacnetOwsNetwork/BacnetAwsNetwork/BacnetNetwork component is supported in any station, regardless of how many different BACnet link-layer protocols are being used.*

As with other NiagaraAX driver networks, the BacnetOwsNetwork should reside under the station's Drivers container. For general information, see "Niagara Bacnet Client Concepts" on page 3-11.

The default view of the BacnetOwsNetwork is the **Bacnet Ows Device Manager**. For more details, see "BACnet OWS Supervisor features" on page A-15 and "Bacnet Ows Device Manager" on page A-15.

bacnetOws-LocalBacnetOwsDevice

LocalBacnetOwsDevice (**Local Device**) is the representation of Niagara as a BACnet device on the BACnet internetwork for a **BACnet OWS Supervisor**, and is a frozen container of a BacnetOwsNetwork. The LocalBacnetOwsDevice has all the same proprerties and components of a regular **LocalBacnetDevice**, including a child **Export Table** component.

Note: *At the time of this document update, a BACnet OWS Supervisor is not licensed for server operation, as those functions were not part of the BTL certification testing for B-AWS compliance. Therefore, the* **ExportTable** *under a LocalBacnetAwsDevice is not typically used.*

For related details, see "Bacnet Local Device" on page 3-21 and also "BACnet OWS and AWS Supervisors overview" on page A-2.

Components in bacnetws module

Note: *Starting in AX-3.6, the* `bacnetws` *module for a* **BACnet Supervisor** *is deprecated in favor of new modules* `bacnetOws` *and* `bacnetAws`*. These modules are used along with the (core) bacnet module for a* **BACnet OWS Supervisor** *or a* **BACnet AWS Supervisor** *products, which were tested and certified as B-OWS or B-AWS respectively, by the BTL (BACnet Testing Laboratories).*

The `bacnetws`*-based* **BACnet Supervisor** *is not BTL certified. See "BACnet AWS and OWS Supervisors" on page A-1 for more details on the B-OWS and B-AWS Supervisors.*

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

- BacnetWsClientLayer
- BacnetWsDevice
- BacnetWsNetwork
- BacnetWsStack
- LocalBacnetWsDevice

bacnetws-BacnetWsClientLayer

BacnetWsClientLayer (Client) represents the client side of the application layer of the Bacnet communications stack. Client resides under the BacnetWsStack (`Bacnet Comm`) of a BacnetWsNetwork, and has no available properties.

bacnetws-BacnetWsDevice

BacnetWsDevice is a Niagara representation of a remote BACnet device in the BacnetWsNetwork of a **BACnet Supervisor** station. Each BacnetWsDevice includes all the same properties and device extensions (containers) of Points, Schedules, and Trend Logs (Histories) as does a regular BacnetDevice component, for modeling data (from that device) in the station.

For more details, see “Bacnet Device components” on page 3-31, “About Bacnet Device’s Schedules” on page 3-42, and “About Bacnet Trend Logs (Histories)” on page 3-47. Note that a BacnetWsDevice also provides additional actions not available on a regular BacnetDevice, providing supervisory control functions. These actions are: Comm Control, Reinitialize Device, Get Enrollment Summary, Get Event Information. See “BacnetWsDevice additions” on page B-9 for related details.

bacnetws-BacnetWsNetwork

BacnetWsNetwork is the base container for all Bacnet components in a **BACnet Supervisor** station (used *instead* of a BacnetNetwork). In addition to being the network container for BacnetWsDevices and/or BacnetDevices and their child data objects (Bacnet proxy points), it contains the station’s BACnet communications protocol stack (Bacnet Comm), plus a LocalBacnetWsDevice, which configures the station’s representation as a BACnet device.

Note: *Only one BacnetWsNetwork/BacnetNetwork component is supported in any station, regardless of how many different BACnet link-layer protocols are being used.*

As with other NiagaraAX driver networks, the BacnetWsNetwork should reside under the station’s Drivers container. For general information, see “Niagara Bacnet Client Concepts” on page 3-11.

The default view of the BacnetNetwork is the **Bacnet Ws Device Manager**. For more details, see “BACnet Supervisor” on page B-1 and “Bacnet Ws Device Manager” on page B-2.

bacnetws-BacnetWsStack

BacnetWsStack (**Bacnet Comm**) provides the protocol stack for BACnet communications for a **BACnet Supervisor** station, and is a frozen child container slot of its BacnetWsNetwork. In turn, it has child frozen container slots BacnetClientLayer, BacnetServerLayer, BacnetTransportLayer, and BacnetNetworkLayer. For more details, see “About Bacnet Comm” on page 3-12.

bacnetws-LocalBacnetWsDevice

LocalBacnetWsDevice (**Local Device**) is the representation of Niagara as a BACnet device on the BACnet internetwork for a **BACnet Supervisor**, and is a frozen container of a BacnetWsNetwork. The LocalBacnetWsDevice has all the same proprerties and components of a regular LocalBacnetDevice, including a child BacnetExportTable container that provides server-side functions to export station objects as BACnet objects and service BACnet client requests.

For related details, see “Bacnet Local Device” on page 3-21 and “Bacnet server configuration overview” on page 4-60.

The LocalBacnetWsDevice also provides additional properties not available on a LocalBacnetDevice, used in operation as a BACnet time-synch *master*. See “LocalBacnetWsDevice additions” on page B-9.

Appendix A

BACnet AWS and OWS Supervisors

Concurrent with this document update in the AX-3.6 timeframe, and starting in a maintenance release of AX-3.6, there are now two BTL[1]-certified BACnet Supervisor products:

- **`BACnet AWS Supervisor`** — A specially licensed Supervisor that provides client supervisory access and control of BACnet devices, as a BACnet "Advanced Operator Workstation" (B-AWS).
- **`BACnet OWS Supervisor`** — A specially licensed Supervisor that provides client access and control of BACnet devices, as a BACnet "Operator Workstation" (B-OWS).

A **`BACnet AWS Supervisor`** or **`BACnet OWS Supervisor`** lets you interact with BACnet devices from a supervisory standpoint, as described in the BACnet Specification for the B-AWS or B-OWS device profiles. Each Supervisor includes most of the same functionality as the regular NiagaraAX BACnet driver, as described in other sections of this document.

Note: *At the time of this document, one exception is that neither the BACnet AWS or BACnet OWS Supervisor is licensed for "server side" (BACnet export) of objects, unlike the regular NiagaraAX BACnet driver that runs on a JACE, or the previous "BACnet Supervisor" product. Typically this is not an issue, as a BACnet workstation's purpose is to be the ultimate "authoritative client" to all other BACnet devices.*

Note that each product above requires a Windows PC platform running AX-3.6 or later, and uses a different licensing scheme (and Niagara modules) than the previous BACnet Supervisor product, described in another appendix. That previous BACnet Supervisor (based on the now-deprecated `bacnetws` module), adheres to the B-OWS device profile—however, it is *not* BTL-certified.

Note: *Because of the new* `bacnetAws` *and* `bacnetOws` *module scheme, an existing* **`BACnet Supervisor`** *station database (using the* **`BacnetWsNetwork`** *from the* `bacnetws` *module) is not directly upgradable to run as a* **`BACnet AWS Supervisor`** *or* **`BACnet OWS Supervisor`**. *However, an "offline conversion utility" is provided in the* `bacnetAws` *and* `bacnetOws` *modules. This utility converts the saved station database (config.bog) of a previous* **`BACnet Supervisor`** *to one compatible with one of the new Supervisor types. See **"Converting to a BACnet AWS or OWS Supervisor station"** on page A-18, and also the leading Note: in the appendix about the "BACnet Supervisor" on page B-1.*

This appendix describes the *additional* features and functionality specific to a **`BACnet AWS Supervisor`** and **`BACnet OWS Supervisor`**, and has the following main sections:

[1] BTL is BACnet Testing Laboratories, where BTL certification means the product/device was tested by an official BACnet Testing Laboratory to certify its BACnet PICS (Protocol Implementation Conformance Statement).

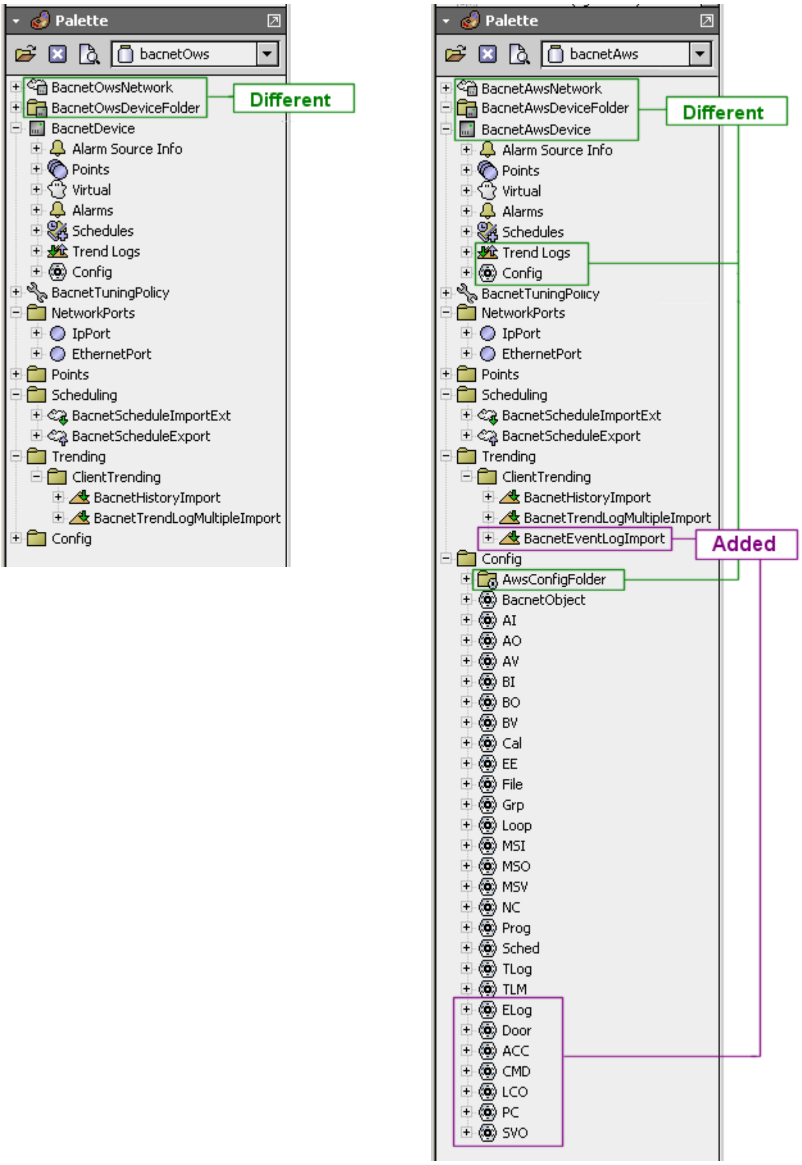
BACnet OWS and AWS Supervisors overview

Each BTL-certified BACnet Supervisor type (OWS, AWS) uses much of the same software as the regular BACnet driver, based on the (core) `bacnet` module. Other modules "extend" for enhanced operations.

- A BACnet OWS Supervisor uses the `bacnet` module and the `bacnetOws` module.
- A BACnet AWS Supervisor uses the `bacnetAws` module, in *addition* to modules `bacnet` and `bacnetOws`.

For convenience, the *palette* in the `bacnetOws` and `bacnetAws` modules include all available components, including all the `bacnet` components (except server-related ones). The palette for the bacnetOws and bacnetAws modules are shown expanded in Table A-1 below, with most items unchanged from the `bacnet` palette not expanded.

Table A-1 *Palettes for bacnetOws (left) and bacnetAws (right), with differences from bacnet palette*



```
Palette
bacnetOws
+ BacnetOwsNetwork
+ BacnetOwsDeviceFolder
- BacnetDevice
    + Alarm Source Info
    + Points
    + Virtual
    + Alarms
    + Schedules
    + Trend Logs
    + Config
+ BacnetTuningPolicy
- NetworkPorts
    + IpPort
    + EthernetPort
+ Points
- Scheduling
    + BacnetScheduleImportExt
    + BacnetScheduleExport
- Trending
    - ClientTrending
        + BacnetHistoryImport
        + BacnetTrendLogMultipleImport
+ Config
```

```
Palette
bacnetAws
+ BacnetAwsNetwork
- BacnetAwsDeviceFolder
- BacnetAwsDevice
    + Alarm Source Info
    + Points
    + Virtual
    + Alarms
    + Schedules
    + Trend Logs
    + Config
+ BacnetTuningPolicy
- NetworkPorts
    + IpPort
    + EthernetPort
+ Points
- Scheduling
    + BacnetScheduleImportExt
    + BacnetScheduleExport
- Trending
    - ClientTrending
        + BacnetHistoryImport
        + BacnetTrendLogMultipleImport
        + BacnetEventLogImport
- Config
    + AwsConfigFolder
    + BacnetObject
    + AI
    + AO
    + AV
    + BI
    + BO
    + BV
    + Cal
    + EE
    + File
    + Grp
    + Loop
    + MSI
    + MSO
    + MSV
    + NC
    + Prog
    + Sched
    + TLog
    + TLM
    + ELog
    + Door
    + ACC
    + CMD
    + LCO
    + PC
    + SVO
```

As shown above, the network component and a few others are *different* in the `bacnetOws` and `bacnetAws` modules, often with a different default view. For example, there is a **`Bacnet Aws Device Manager`** view and a **`Bacnet Ows Device Manager`** view on the two types of network and device folder components. The `bacnetAws` palette provides *additional* components, not found in the `bacnet` module.

Note that "server side" components found in the **`bacnet`** palette are missing, as currently neither the BACnet AWS Supervisor nor the BACnet OWS Supervisor are licensed for server operation. Also, there is no "**`Mstp`**" port under the **`NetworkPorts`** folder—a port type valid only for QNX-based JACE hosts.

As usual, you seldom need to work from the palette when engineering a **BACnet AWS Supervisor** or **BACnet OWS Supervisor** station. Instead, the various Bacnet manager views simplify component creation, enforcing proper component hierarchy. For example, the **Bacnet Aws Config Manager** view on the **Config** device extension of a **BacnetAwsDevice** provides a "Discover" routine, finding the appropriate types of Config objects to represent discovered BACnet objects (including "added" ones).

The **Bacnet Aws Config Manager** is also unique in that you can *add and delete BACnet objects* in the selected device—providing that the device supports these BACnet services.

Different AWS and OWS manager views are explained in other subsections of this appendix.

AWS and OWS Supervisor module and license requirements

- BACnet AWS Supervisor requirements
- BACnet OWS Supervisor requirements

BACnet AWS Supervisor requirements

To use a **BACnet AWS Supervisor**, these things must be in place:

- The Supervisor PC must be running AX-3.6 or later (build 3.6.41 or higher), with the `bacnet`, `bacnetOws`, and `bacnetAws` modules in its modules folder (also applies for any other Workbench PC with which you access the **BACnet AWS Supervisor**).
- The Supervisor PC must have the `bacnet` feature, `bacnetOws` feature, and `bacnetAws` feature in its Tridium license. Otherwise, the station's **BacnetAwsNetwork** will remain in fault. Note that license may also have limits on the number of devices, points, and so forth.

BACnet OWS Supervisor requirements

To use a **BACnet OWS Supervisor**, these things must be in place:

- The Supervisor PC must be running AX-3.6 or later (build 3.6.41 or higher), with both the `bacnet` and `bacnetOws` modules in its modules folder (also applies for any other Workbench PC with which you access the **BACnet OWS Supervisor**).
- The Supervisor PC must have both the `bacnet` feature and `bacnetOws` feature in its Tridium license. Otherwise, the station's **BacnetOwsNetwork** will remain in fault. Note that license may also have limits on the number of devices, points, and so forth.

BACnet AWS Supervisor features

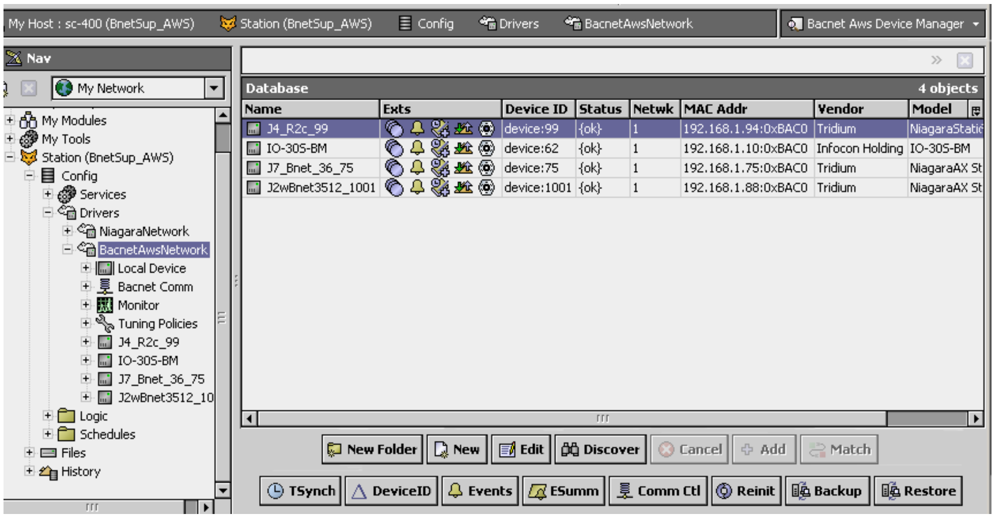
Several enhanced features and additions are provided with a **`BACnet AWS Supervisor`**, including extended views specific to a **`BacnetAwsNetwork`**:

- Bacnet Aws Device Manager
- Bacnet Aws History Import Manager
- Bacnet Aws Config Manager
- Additional BACnet AWS Supervisor features

Bacnet Aws Device Manager

The Bacnet Aws Device Manager (Figure A-1) is the default view for the BacnetAwsNetwork (and any BacnetAwsDeviceFolder), and provides all the same features as the regular Bacnet Device Manager.

Figure A-1 *Bacnet Aws Device Manager is default view for BacnetAwsNetwork*



Name	Exts	Device ID	Status	Netwk	MAC Addr	Vendor	Model
J4_R2c_99		device:99	{ok}	1	192.168.1.94:0xBAC0	Tridium	NiagaraStatic
IO-30S-BM		device:62	{ok}	1	192.168.1.10:0xBAC0	Infocon Holding	IO-30S-BM
J7_Bnet_36_75		device:75	{ok}	1	192.168.1.75:0xBAC0	Tridium	NiagaraAX St
J2wBnet3512_1001		device:1001	{ok}	1	192.168.1.88:0xBAC0	Tridium	NiagaraAX St

You discover BACnet devices in this view the same as in the **`Bacnet Device Manager`** view. For details, see "About Bacnet Device Find Parameters" on page 3-27. When doing a discover, the **`Add`** dialog provides the same parameters for adding BacnetAwsDevices as for BacnetDevices. See "BacnetDevice properties" on page 3-31.

Note: *All devices are represented as* **`BacnetAwsDevices`** *in a BacnetAwsNetwork. Nearly identical to a BacnetDevice, a BacnetAwsDevice differs by its device extensions "Trend Logs" and "Config", and the default views on those extensions.*

The bottom row of buttons in the **`Bacnet Aws Device Manager`** include two that are in the standard **`Bacnet Device Manager`**: **`TSynch`** and **`DeviceID`**. For more details, see "Timesynch (TSynch) function" on page 3-29 and "Device ID function" on page 3-29.

Several *additional* buttons are in the bottom row in the **`Bacnet Aws Device Manager`**. These correspond to additional functions that can be performed on BacnetAwsDevices.

These additional buttons available in the **`Bacnet Aws Device Manager`** are:

- **`Events`** — See "Event Information" on page A-4.
- **`ESumm`** — See "Enrollment Summary" on page A-5.
- **`Comm Ctl`** — See "Communication Control" on page A-6.
- **`Reinit`** — See "Reinitialize Device" on page A-7.
- **`Backup`** — See "Backup Device" on page A-7.
- **`Restore`** — See "Restore Device" on page A-9.

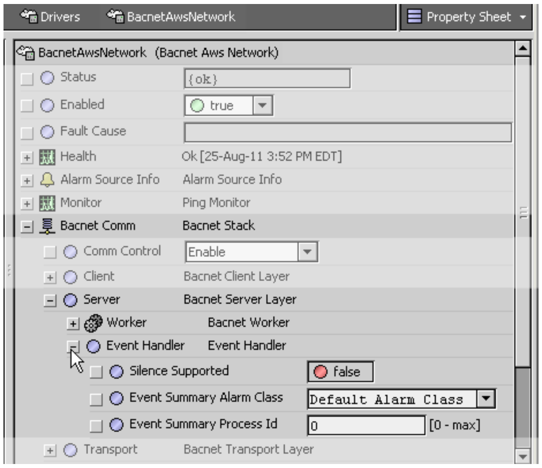
Event Information

Events (Also in **`Bacnet Ows Device Manager`**) This button in the Bacnet Aws Device Manager retrieves event information from the selected device, requesting all outstanding events from the BACnet device. The purpose of this feature is that if Niagara somehow missed receiving an event, you can still retrieve and acknowledge this event using the Event Information service. For example, the Niagara station may not have been running at the time the original event occurred.

If the GetEventInformation service is supported, this will be used to retrieve events. The events retrieved in this manner contain enough information to generate a valid acknowledgment to the remote device. These event summaries will be propagated to the Niagara Alarm Service through the alarm class assigned in the eventSummaryAlarmClass property of the Event Handler.

Event Handler configuration is contained within the Server layer configuration of the network's BacnetComm container, as shown in Figure A-2.

Figure A-2 *Event Summary Alarm Class (BacnetAwsNetwork)*



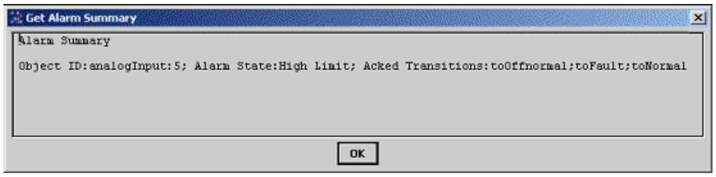
BacnetAwsNetwork (Bacnet Aws Network)	
Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [25-Aug-11 3:52 PM EDT]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Bacnet Comm	Bacnet Stack
Comm Control	Enable
Client	Bacnet Client Layer
Server	Bacnet Server Layer
Worker	Bacnet Worker
Event Handler	Event Handler
Silence Supported	false
Event Summary Alarm Class	Default Alarm Class
Event Summary Process Id	0 [0 - max]
Transport	Bacnet Transport Layer

If the GetEventInformation service is not supported, but the GetAlarmSummary service is supported, the events will be retrieved using GetAlarmSummary. The GetAlarmSummary service retrieves only events with a Notify_Type of "`alarm`", so events with a Notify_Type of "`event`" are not retrieved.

Note: *The GetAlarmSummary service has been* deprecated *by the BACnet Committee, and manufacturers should no longer be making devices that use it. Typically, this is seen only in older devices.*

Event summaries retrieved using GetAlarmSummary do not have enough information to generate a valid acknowledgment, so they are not propagated to the Niagara Alarm Service. In this case, a dialog box is presented to the user, such as shown in Figure A-3.

Figure A-3 *Alarm Summary dialog, showing a single object in alarm*

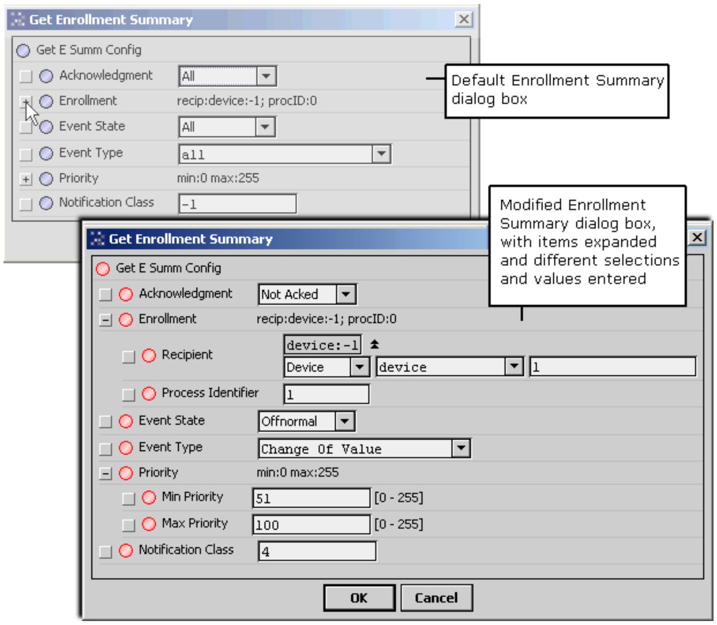


```
Alarm Summary
Object ID:analogInput:5; Alarm State:High Limit; Acked Transitions:toOffnormal;toFault;toNormal
```

Enrollment Summary

ESumm (Also in **`Bacnet Ows Device Manager`**) This button in the Bacnet Aws Device Manager retrieves a summary of all objects within the device that match a set of filter criteria that you define. When you click the Enrollment Summary button, a filter configuration dialog box is displayed, as shown in Figure A-4.

Figure A-4 *Get Enrollment Summary request configuration dialog*



Get Enrollment Summary

Get E Summ Config

○ Acknowledgment	All	
○ Enrollment	recip:device:-1; procID:0	
○ Event State	All	
○ Event Type	all	
○ Priority	min:0 max:255	
○ Notification Class	-1	

Default Enrollment Summary dialog box

Get Enrollment Summary

Get E Summ Config

○ Acknowledgment	Not Acked	
○ Enrollment	recip:device:-1; procID:0	
○ Recipient	device:-1 Device · device · 1	
○ Process Identifier	1	
○ Event State	Offnormal	
○ Event Type	Change Of Value	
○ Priority	min:0 max:255	
○ Min Priority	51	[0 - 255]
○ Max Priority	100	[0 - 255]
○ Notification Class	4	

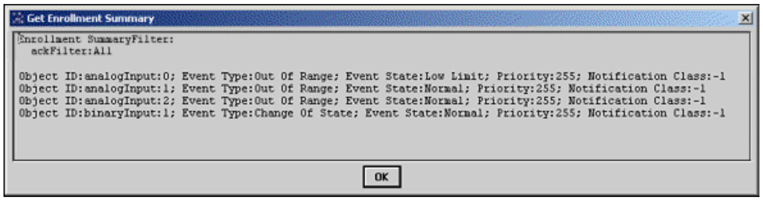
Modified Enrollment Summary dialog box, with items expanded and different selections and values entered

Here you configure the various options to define the specific types of objects you wish to locate, and then click **OK**. The device is queried for a list of objects that meet the specified filter criteria, and the result is displayed in a dialog box, as shown in Figure A-5.

Filter options include the following:

- **Acknowledgement**
  Filter on points that are acknowledged, or unacknowledged, or leave the filter open to all points.
- **Enrollment**
  Filter on only event mechanisms that have a specific recipient, definable either by device id or by device address.
- **Event State**
  Specify a particular event state, to restrict the request to include only points that are currently in a specific event state.
- **Event Type**
  Restrict the filter to objects that use a particular event type algorithm.
- **Priority**
  Choose to only include objects where the priority of the last transition is within certain bounds.
- **Notification Class**
  Include only objects and event mechanisms where the notification class used to route the event notifications is equal to a specific number.

Figure A-5 *Example Get Enrollment Summary response dialog*



Get Enrollment Summary

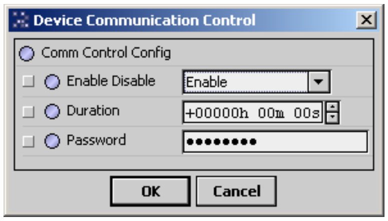
```
Enrollment SummaryFilter:
  ackFilter:All

Object ID:analogInput:0; Event Type:Out Of Range; Event State:Low Limit; Priority:255; Notification Class:-1
Object ID:analogInput:1; Event Type:Out Of Range; Event State:Normal; Priority:255; Notification Class:-1
Object ID:analogInput:2; Event Type:Out Of Range; Event State:Normal; Priority:255; Notification Class:-1
Object ID:binaryInput:1; Event Type:Change Of State; Event State:Normal; Priority:255; Notification Class:-1
```

Communication Control

Comm Ctl This button in the Bacnet Aws Device Manager allows you to control a device's ability to generate traffic on the network. This could be useful in a diagnostic situation, where you want to temporarily eliminate all traffic except from a single device. Or you may have a faulty device that is sending extraneous data, and you wish to "silence it" until you can identify the cause of the problem. When you click this button, a dialog box appears which allows you to configure the request, as shown in Figure A-6.

Figure A-6 *Device Communication Control configuration dialog*



Device Communication Control

○ Comm Control Config
□ ○ Enable Disable: Enable
□ ○ Duration: +00000h 00m 00s
□ ○ Password: ••••••••

Configuration options include the following:

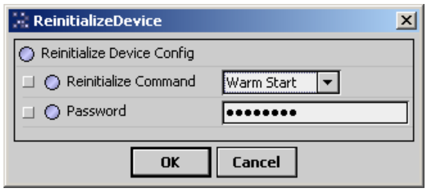
- **Enable Disable**
  You can choose to Enable or Disable the device from sending any BACnet requests. You can also choose to "Disable Initiation," which allows the device to respond to BACnet requests, while being prevented from initiating any requests of its own. Note that the sending of a single I-Am request is allowed if a Who-Is request that matches the device's device id is received.
- **Duration**
  You can configure the length of time for which communications will be disabled, after which the communications will automatically be re-enabled for the device.
- **Password**
  Some devices may require a password to disable their communications. If a device requires a password, you may enter it here.

When you click **OK**, a request to control the communication is sent to the device. The result is displayed in a dialog box, with either a success message, or a message including the error returned by the device.

Reinitialize Device

Reinit This button in the Bacnet Aws Device Manager allows you to restart a device. In the popup dialog (Figure A-7), you may choose either "Cold Start" or "Warm Start."

Figure A-7 *Reinitialize Device configuration dialog*



ReinitializeDevice

○ Reinitialize Device Config
□ ○ Reinitialize Command: Warm Start
□ ○ Password: ••••••••

Note: *The specific meaning of terms Cold Start and Warm Start is left up to the device's manufacturer to define, so make sure you understand exactly what each procedure entails for the device in question.*

As shown above, this dialog has two fields:

- **Reinitialize Command**
  Select either Warm Start (default) or Cold Start (see Note: above).
- **Password**
  Enter the password required by the device to invoke this command. Some devices may not require a password, so if you do not enter anything, no password is sent to the device.

When you click **OK**, the reinitialize request is sent to the device. The result is displayed in a dialog box, as either a success or failure.

Backup Device

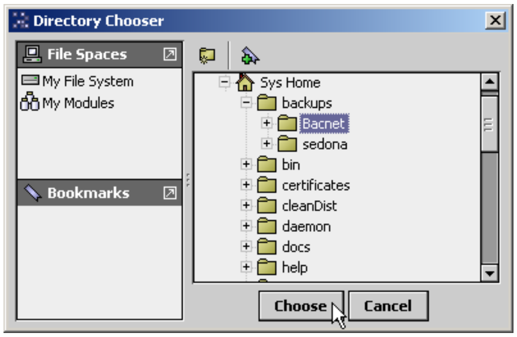
Backup This button in the Bacnet Aws Device Manager allows you to backup the selected device's configuration, as one or more "restorable" files on the **BACnet AWS Supervisor** PC. If needed, you can use the Restore function later to reinstall this backup.

Note: *The target device must support the BACnet DM-BR-B BIBB, as part of the B-BC device profile conformance. Both the BACnet specification 135-2008 and addendum 135-2008n are supported.*

In addition, the device may require a password before it initiates a backup or a restore. Otherwise, a backup or restore job will immediately fail, showing the associated reason in the job log details. For example, you may see "Unrecognized Service" or "Security: Password Failure".

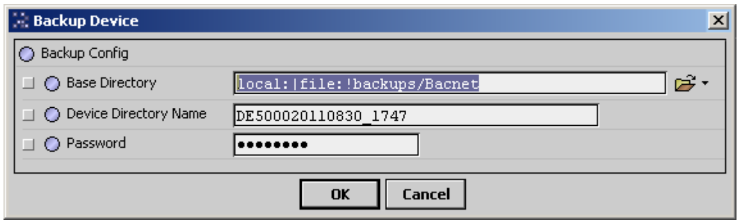
When you click **Backup**, the standard **Directory Chooser** appears, in which you specify the target directory for the backup file(s). If needed, use the “new folder” control, as was done in the example shown in Figure A-8 to make a “Bacnet” folder under the system “backups” folder.

Figure A-8 *Directory Chooser dialog to specify backup directory*



After you choose the target directory, the **Backup Device** dialog appears, as shown in Figure A-9.

Figure A-9 *Backup Device configuration dialog*



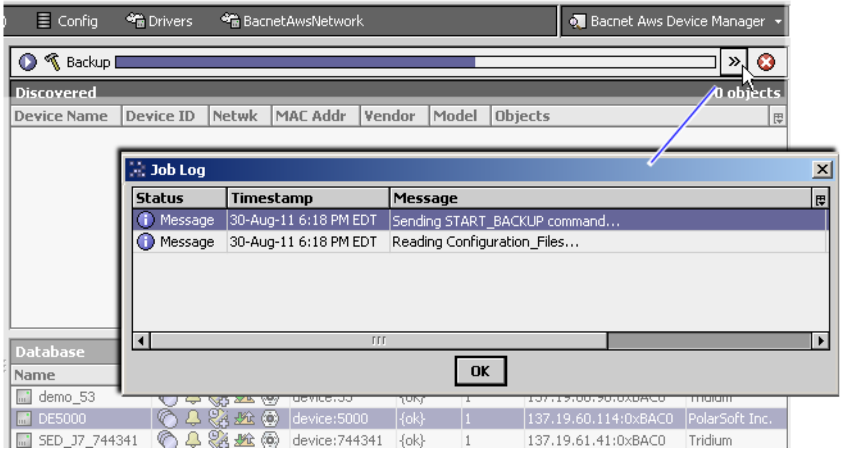
□ ○ Base Directory	local:|file:!backups/Bacnet
□ ○ Device Directory Name	DE500020110830_1747
□ ○ Password	••••••••

Configuration fields in this dialog include the following:

- **Base Directory**
  Reflects the ord for the local target directory, as previously chosen in the **Directory Chooser**. If needed, you can modify it using file ord syntax.
- **Device Directory Name**
  This specifies the subdirectory that will be made under the base directory for the device’s backup file(s). This defaults to *<BacnetAwsDeviceName><YYYYMMDD_HHMM>*, where the last portion is a timestamp that reflects when the backup was initiated. If desired, you can modify.
- **Password**
  Some devices may require a password to initiate a backup and/or a restore. If a device requires a password, you may enter it here.

When you click **OK**, the backup request is sent to the device. A Backup job is started, and as shown in Figure A-9 you can click on the job log control near the top of the manager to see the progress of the backup.

Figure A-10 *Backup job started, Job Log details dialog*



Status	Timestamp	Message
Message	30-Aug-11 6:18 PM EDT	Sending START_BACKUP command...
Message	30-Aug-11 6:18 PM EDT	Reading Configuration_Files...

demo_53	device:53	{ok}	1	137.19.60.98:0xBAC0	Tridium
DE5000	device:5000	{ok}	1	137.19.60.114:0xBAC0	PolarSoft Inc.
SED_J7_744341	device:744341	{ok}	1	137.19.61.41:0xBAC0	Tridium

When the job completes, it will post a success or failed status. A successful backup has the specified device subdirectory under the target base directory, containing one or more backup files for the device.

Note: *An invalid backup job typically fails immediately. However, a valid backup job may take several minutes to complete, depending on the implementation in the target BACnet device. Typically, the device executes some "preparation routine" first, before assembling and sending the backup files. In the case of JACE stations running the Bacnet driver, there is a set "server side" routine as well as associated properties, found in the LocalBacnetDevice of the BacnetNetwork. For more details, see "Local Device backup and restore properties" on page 4-80.*

Restore Device

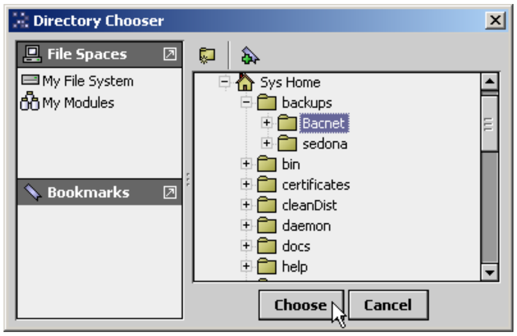
Restore This button in the Bacnet Aws Device Manager allows you to restore a previous backup to the selected device, where a backup is one or more "restorable" files saved on the **BACnet AWS Supervisor** PC. See "Backup Device" on page A-7.

Note: *The target device must support the BACnet DM-BR-B BIBB, as part of the B-BC device profile conformance. Both BACnet specification addendums 135-2008 and 135-2008n are supported.*

In addition, you may require a device password to have it initiate a backup or a restore. Otherwise, a backup or restore job will immediately fail, showing the associated reason in the job log details. For example, you may see "Unrecognized Service" or "Security: Password Failure".

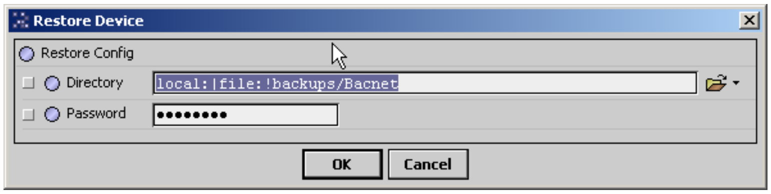
When you click **Restore**, the standard **Directory Chooser** appears, in which you navigate to the directory that contains the backup file(s), as shown in Figure A-11.

Figure A-11 *Directory Chooser dialog to specify directory with backup file(s)*



After you choose the source directory, the **Restore Device** dialog appears, as shown in Figure A-12.

Figure A-12 *Restore Device configuration dialog*



○ Directory: local:|file:!backups/Bacnet

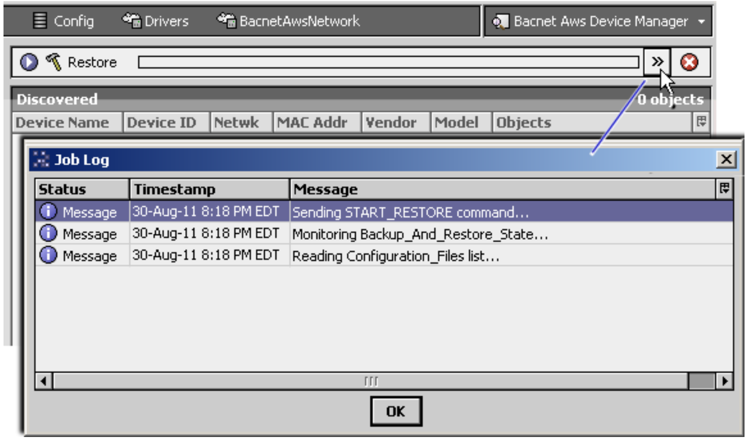
○ Password: ••••••••

Configuration fields in this dialog include the following:

- **Directory**
  Reflects the ord for the local source directory, as previously chosen in the **Directory Chooser**. If needed, you can modify it using file ord syntax.
- **Password**
  Some devices may require a password to initiate a backup and/or a restore. If a device requires a password, you may enter it here.

When you click **OK**, the restore backup request is sent to the device. A Restore job is started, and as shown in Figure A-13 you can click on the job log control near the top of the manager to see the job progress.

Figure A-13 *Restore job started, Job Log details dialog*



Status	Timestamp	Message
Message	30-Aug-11 8:18 PM EDT	Sending START_RESTORE command…
Message	30-Aug-11 8:18 PM EDT	Monitoring Backup_And_Restore_State…
Message	30-Aug-11 8:18 PM EDT	Reading Configuration_Files list…

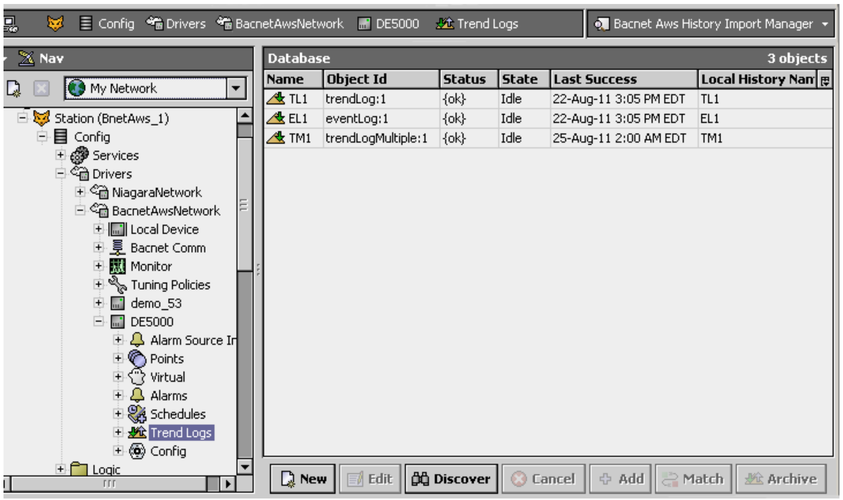
When the job completes, it will post a success or failed status. Successful restores will install the backup file(s) found in the source directory, and typically re-initialize (reboot) the device.

Note: *An invalid restore job typically fails immediately. However, a valid restore job may take several minutes to complete, depending on the implementation in the target BACnet device. Typically, the device executes some "preparation routine" first, before accepting the backup files. In the case of JACE stations running the Bacnet driver, there is a set "server side" routine as well as associated properties, found in the LocalBacnetDevice of the BacnetNetwork. For more details, see "About backup and restore operations" on page 4-80.*

Bacnet Aws History Import Manager

The Bacnet Aws History Import Manager (Figure A-14) is the default view for the **Trend Logs** device extension under any **BacnetAwsDevice** (BacnetAwsHistoryImportExt).

Figure A-14 *Bacnet Aws History Import Manager is default view for BacnetAwsHistoryImportExt*



Name	Object Id	Status	State	Last Success	Local History Nam
TL1	trendLog:1	{ok}	Idle	22-Aug-11 3:05 PM EDT	TL1
EL1	eventLog:1	{ok}	Idle	22-Aug-11 3:05 PM EDT	EL1
TM1	trendLogMultiple:1	{ok}	Idle	25-Aug-11 2:00 AM EDT	TM1

This view provides all the same features as the regular **Bacnet History Import Manager**. You typically use the Discover feature to find BACnet *Trend Log* objects in a device, and add import descriptors for them, which makes corresponding Niagara histories in the station. See "About the Bacnet History Import Manager" on page 3-48 for more details.

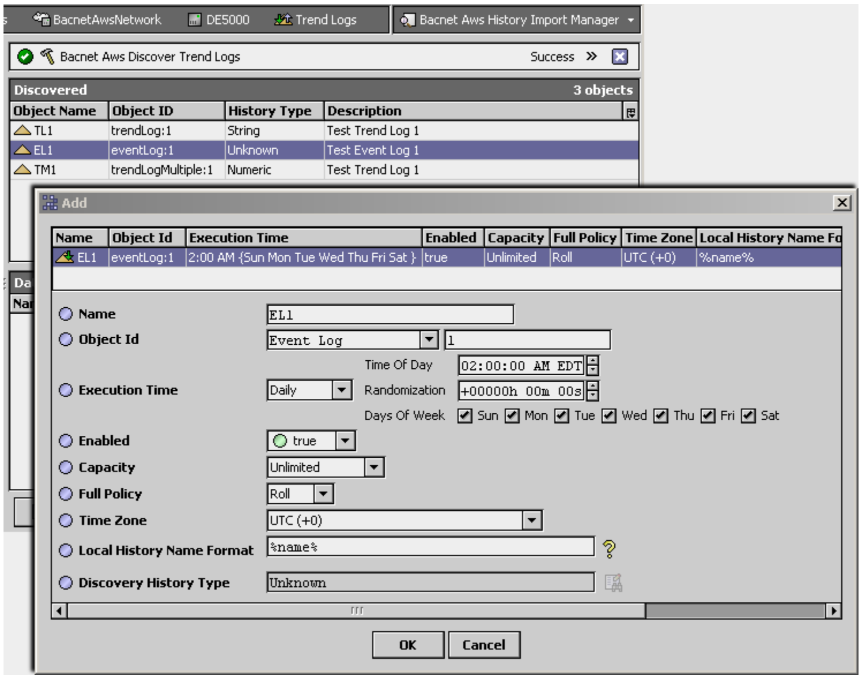
Note the **Bacnet Aws History Import Manager** also supports the discovery and addition of BACnet *Trend Log Multiple* objects, as does the regular BACnet driver starting in build 3.6.41. Corresponding import descriptors result in *multiple* Niagara histories, and each import descriptor has a special view (see "Bacnet Trend Multiple View" on page 3-51 for related details).

In addition to Trend Log and Trend Log Multiple object support, the **Bacnet Aws History Import Manager** provides *additional history import support* for BACnet *Event Log* objects.

An **Event Log** object records BACnet event notifications with timestamps and other pertinent data for subsequent retrieval. Each imported Event Log results in one Niagara history in the Supervisor station.

Figure A-15 shows the Add dialog (with defaults) for adding a discovered Event Log to import.

Figure A-15 *Event Log object discovered by BACnet AWS Supervisor, default Add dialog*



Bacnet Aws Discover Trend Logs Success

Discovered 3 objects

Object Name	Object ID	History Type	Description
TL1	trendLog:1	String	Test Trend Log 1
EL1	eventLog:1	Unknown	Test Event Log 1
TM1	trendLogMultiple:1	Numeric	Test Trend Log 1

Name	Object Id	Execution Time	Enabled	Capacity	Full Policy	Time Zone	Local History Name Fo
EL1	eventLog:1	2:00 AM {Sun Mon Tue Wed Thu Fri Sat }	true	Unlimited	Roll	UTC (+0)	%name%

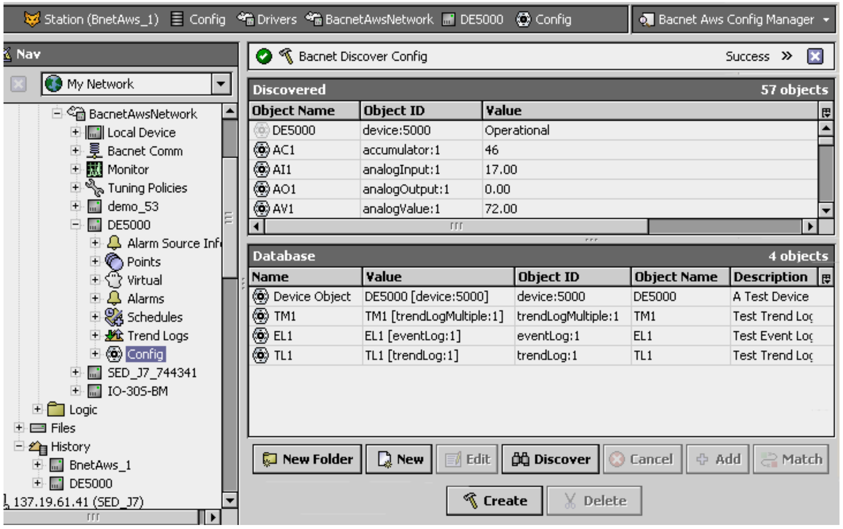
Name	EL1
Object Id	Event Log 1
Execution Time	Daily; Time Of Day 02:00:00 AM EDT; Randomization +00000h 00m 00s; Days Of Week ☑ Sun ☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat
Enabled	true
Capacity	Unlimited
Full Policy	Roll
Time Zone	UTC (+0)
Local History Name Format	%name%
Discovery History Type	Unknown

For related details, see "About the Bacnet History Import Manager" on page 3-48, and "BACnet Trend Log import notes" on page 3-48).

Bacnet Aws Config Manager

The Bacnet Aws Config Manager (Figure A-16) is the default view for the `Config` device extension under any `BacnetAwsDevice` (BacnetAwsConfigDeviceExt).

Figure A-16 *Bacnet Aws Config Manager is default view for BacnetAwsConfigDeviceExt*



Bacnet Discover Config Success

Discovered 57 objects

Object Name	Object ID	Value
DE5000	device:5000	Operational
AC1	accumulator:1	46
AI1	analogInput:1	17.00
AO1	analogOutput:1	0.00
AV1	analogValue:1	72.00

Database 4 objects

Name	Value	Object ID	Object Name	Description
Device Object	DE5000 [device:5000]	device:5000	DE5000	A Test Device
TM1	TM1 [trendLogMultiple:1]	trendLogMultiple:1	TM1	Test Trend Lo(
EL1	EL1 [eventLog:1]	eventLog:1	EL1	Test Event Lo(
TL1	TL1 [trendLog:1]	trendLog:1	TL1	Test Trend Lo(

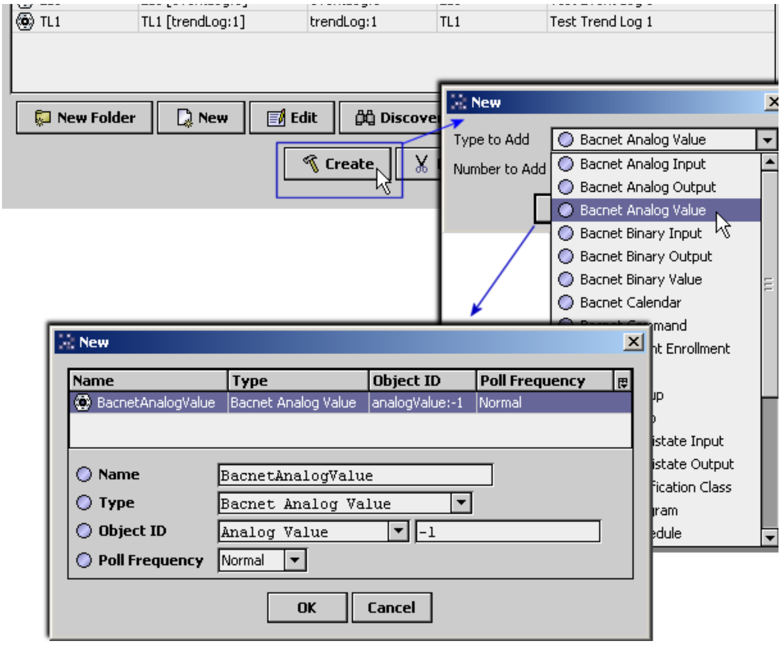
This view provides all the same features as the regular **`Bacnet Config Manager`**. For details, see "About the Config Device Ext container" on page 3-33, and "About Bacnet Config objects" on page 3-34).

The **`Bacnet Aws Config Manager`** also provides two additional buttons:

- **Create**
  To create one or more new BACnet object in the remote BACnet device. A popup **New** dialog allows for selection of BACnet object type and numbers to add.
- **Delete**
  To delete selected BACnet object(s) in the remote BACnet device.

The selected BACnet device must support such operations for these buttons to be available, otherwise they remain dimmed. Figure A-17 shows the default **New** dialogs from selecting an Analog Value type object to create in the BACnet device.

Figure A-17 Create new BACnet object example from Bacnet Aws Config Manager (defaults shown)



TL1	TL1 [trendLog:1]	trendLog:1	TL1	Test Trend Log 1

Name	Type	Object ID	Poll Frequency
BacnetAnalogValue	Bacnet Analog Value	analogValue:-1	Normal

○ Name: BacnetAnalogValue
○ Type: Bacnet Analog Value
○ Object ID: Analog Value -1
○ Poll Frequency: Normal

Name is the BACnet object name, and will also be the Niagara name for the Config object created in the station. Typically you change the name from defaults, e.g. “`BacnetAnalogValue`” as shown above.

Note: *Enter a valid instance number for the Object ID, replacing the default* `-1` *value. This number should be unique from any other instance (of the same object type) in this device. For example, if the device already has Analog Value objects, using instance numbers 0 through 5, enter unique Object ID instance number(s) above 5. Instance number range is from 0 to 4194302.*

Additional BACnet AWS Supervisor features

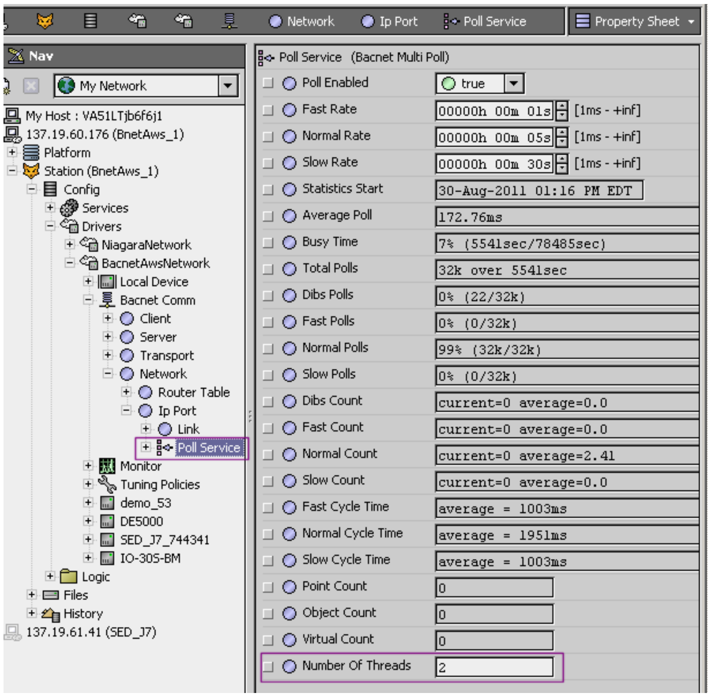
Apart from the different manager views on the **`BacnetAwsNetwork`**, and device extensions **`Trend Logs`** and **`Config`** for each **`BacnetAwsDevice`** (supporting additional BACnet object types), a BacnetAwsNetwork also supports the following:

- Configurable threads for polling
- Configurable threads for proxy point writes

Configurable threads for polling

If needed, in a BacnetAwsNetwork (or BacnetOwsNetwork) you can configure for *more than two processing threads* for the **`Poll Service`** used by the **`Bacnet Comm`**’s network port—that is, the **`IpPort`** and/or **`Ethernet`** port. Find this on the property sheet of the Poll Scheduler component, as shown in Figure A-18.

Figure A-18 *Number Of Threads property in a port's PollService is configurable over 2, if needed*



Poll Service (Bacnet Multi Poll)

Poll Enabled	true
Fast Rate	00000h 00m 01s [1ms - +inf]
Normal Rate	00000h 00m 05s [1ms - +inf]
Slow Rate	00000h 00m 30s [1ms - +inf]
Statistics Start	30-Aug-2011 01:16 PM EDT
Average Poll	172.76ms
Busy Time	7% (5541sec/78485sec)
Total Polls	32k over 5541sec
Dibs Polls	0% (22/32k)
Fast Polls	0% (0/32k)
Normal Polls	99% (32k/32k)
Slow Polls	0% (0/32k)
Dibs Count	current=0 average=0.0
Fast Count	current=0 average=0.0
Normal Count	current=0 average=2.41
Slow Count	current=0 average=0.0
Fast Cycle Time	average = 1003ms
Normal Cycle Time	average = 1951ms
Slow Cycle Time	average = 1003ms
Point Count	0
Object Count	0
Virtual Count	0
Number Of Threads	2

In a regular BacnetNetwork (typically hosted by a JACE), this property is fixed (read-only) at 2 threads. In some cases with the **`BacnetAwsNetwork`** hosted by the PC platform of a BACnet AWS Supervisor (or with the **`BacnetOwsNetwork`** hosted by the PC platform of a BACnet OWS Supervisor), increasing this number over 2 may provide some performance enhancements.

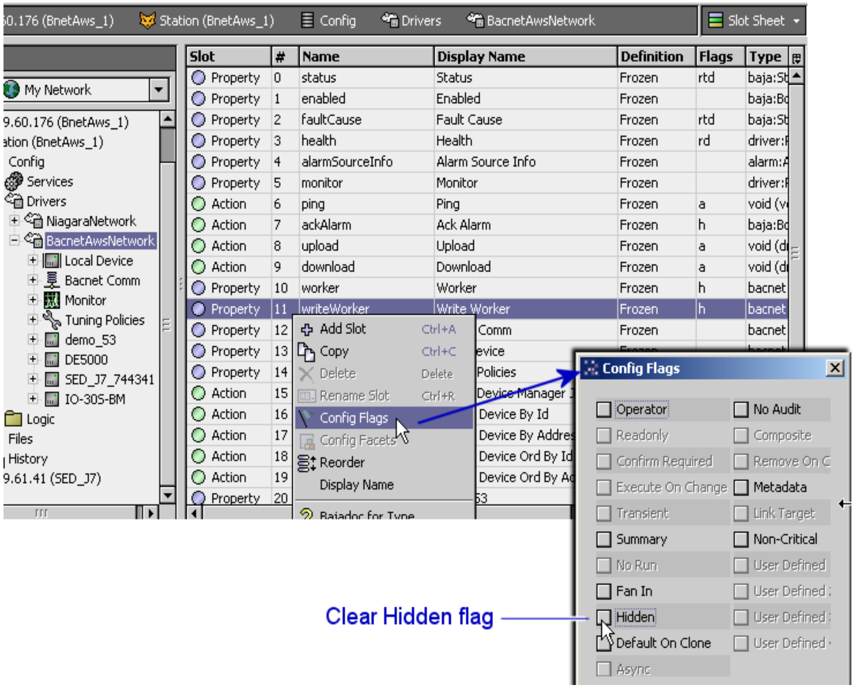
For related details, see "Basic Bacnet polling design" on page 3-17.

Configurable threads for proxy point writes

If needed, in a BacnetAwsNetwork (or BacnetOwsNetwork) you can configure the number of processing threads to be used for proxy point writes. In a regular BacnetNetwork (typically hosted by a JACE), this is fixed at a *single* thread: the "writeWorker" thread.

For any Bacnet network, the "WriteWorker" component is a "hidden" slot. To access it from the property sheet, go to the network's *slot sheet* and clear the hidden flag, as shown in Figure A-19.

Figure A-19 *Clearing hidden flag from slot sheet of BacnetAwsNetwork*



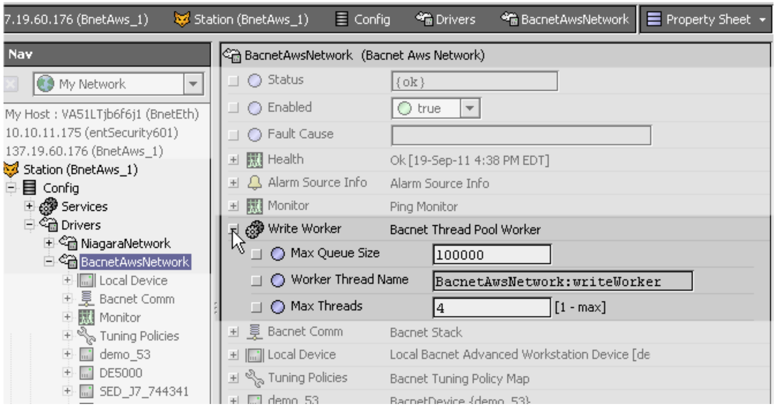
Slot	#	Name	Display Name	Definition	Flags	Type
Property	0	status	Status	Frozen	rtd	baja:St
Property	1	enabled	Enabled	Frozen		baja:Bo
Property	2	faultCause	Fault Cause	Frozen	rtd	baja:St
Property	3	health	Health	Frozen	rd	driver:F
Property	4	alarmSourceInfo	Alarm Source Info	Frozen		alarm:A
Property	5	monitor	Monitor	Frozen		driver:F
Action	6	ping	Ping	Frozen	a	void (v
Action	7	ackAlarm	Ack Alarm	Frozen	h	baja:Bo
Action	8	upload	Upload	Frozen	a	void (d
Action	9	download	Download	Frozen	a	void (d
Property	10	worker	Worker	Frozen	h	bacnet
Property	11	writeWorker	Write Worker	Frozen	h	bacnet
Property	12		Comm	Frozen		bacnet
Property	13		evice			
Property	14		Policies			
Action	15		Device Manager			
Action	16		Device By Id			
Action	17		Device By Addres			
Action	18		Device Ord By Id			
Action	19		Device Ord By Ad			
Property	20		53			

Config Flags

□ Operator	□ No Audit
□ Readonly	□ Composite
□ Confirm Required	□ Remove On C
□ Execute On Change	□ Metadata
□ Transient	□ Link Target
□ Summary	□ Non-Critical
□ No Run	□ User Defined
□ Fan In	□ User Defined
□ Hidden	□ User Defined
□ Default On Clone	□ User Defined
□ Async	

Then from the network's *property sheet*, expand the **`Write Worker`** container for child properties.

Figure A-20 *Write Worker for BacnetAwsNetwork (or BacnetOwsNetwork) has "Max Threads" property*



BacnetAwsNetwork (Bacnet Aws Network)

Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [19-Sep-11 4:38 PM EDT]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Write Worker	Bacnet Thread Pool Worker
Max Queue Size	100000
Worker Thread Name	BacnetAwsNetwork:writeWorker
Max Threads	4 [1 - max]
Bacnet Comm	Bacnet Stack
Local Device	Local Bacnet Advanced Workstation Device [de
Tuning Policies	Bacnet Tuning Policy Map
demo_53	BacnetDevice {demo_53}

As shown in Figure A-20, the default "Max Threads" value for the Write Worker in a BacnetAwsNetwork is 4 threads. You can adjust it upwards if needed. This may be useful if you have a large number of writable Bacnet proxy points that are configured in their tuning policy to writeOnStart.

BACnet OWS Supervisor features

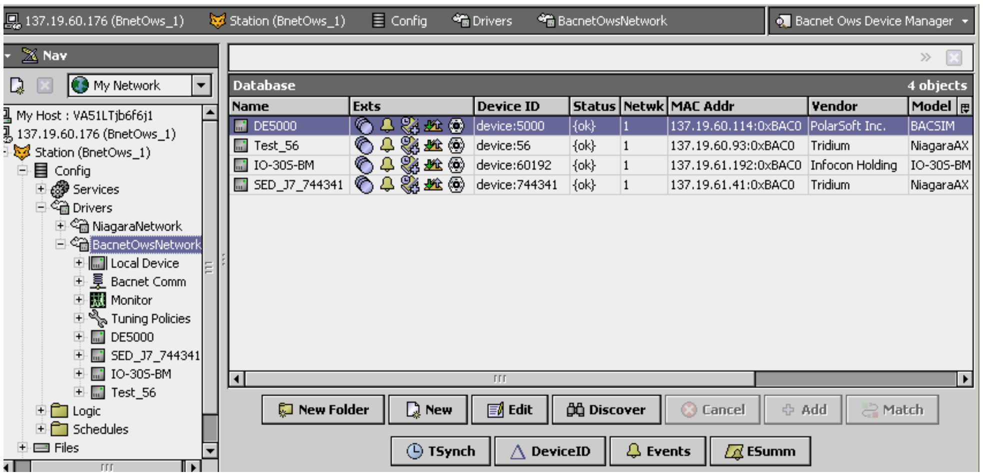
A few features and additions are provided with a **BACnet OWS Supervisor** over the regular BACnet driver, including an extended view specific to a **BacnetOwsNetwork**:

- Bacnet Ows Device Manager
- Additional BACnet OWS Supervisor features

Bacnet Ows Device Manager

The Bacnet Ows Device Manager (Figure A-21) is the default view for the **BacnetOwsNetwork** (and any BacnetOwsDeviceFolder), and provides all the same features as the regular **Bacnet Device Manager**.

Figure A-21 *Bacnet Ows Device Manager is default view for BacnetOwsNetwork*



Name	Exts	Device ID	Status	Netwk	MAC Addr	Vendor	Model
DE5000		device:5000	{ok}	1	137.19.60.114:0xBAC0	PolarSoft Inc.	BACSIM
Test_56		device:56	{ok}	1	137.19.60.93:0xBAC0	Tridium	NiagaraAX
IO-30S-BM		device:60192	{ok}	1	137.19.61.192:0xBAC0	Infocon Holding	IO-30S-BM
SED_J7_744341		device:744341	{ok}	1	137.19.61.41:0xBAC0	Tridium	NiagaraAX

You discover BACnet devices in this view the same as in the **Bacnet Device Manager** view. For details, see "About Bacnet Device Find Parameters" on page 3-27. When doing a discover, the **Add** dialog provides standard fields for **BacnetDevice** components. See "BacnetDevice properties" on page 3-31.

Note: *All devices are represented as standard* **BacnetDevices** *in a BacnetOwsNetwork.*

The bottom row of buttons in the **Bacnet Ows Device Manager** include two that are in the standard **Bacnet Device Manager**: **TSynch** and **DeviceID**. For more details, see "Timesynch (TSynch) function" on page 3-29 and "Device ID function" on page 3-29.

Two *additional* buttons are in the bottom row in the **Bacnet Ows Device Manager**. These correspond to additional B-OWS functions that can be performed on BacnetDevices.

These additional buttons available in the **Bacnet Ows Device Manager** are:

- **Events** — See "Event Information" on page A-15.
- **ESumm** — See "Enrollment Summary" on page A-5.

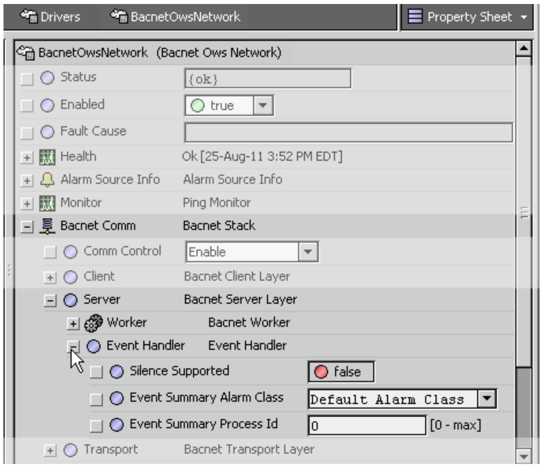
Event Information

Events (Also in the **Bacnet Aws Device Manager**) This button in the Bacnet Ows Device Manager retrieves event information from the selected device, requesting all outstanding events from the BACnet device. The purpose of this feature is that if Niagara somehow missed receiving an event, you can still retrieve and acknowledge this event using the Event Information service. For example, the Niagara station may not have been running at the time the original event occurred.

If the GetEventInformation service is supported, this will be used to retrieve events. The events retrieved in this manner contain enough information to generate a valid acknowledgment to the remote device. These event summaries will be propagated to the Niagara Alarm Service through the alarm class assigned in the eventSummaryAlarmClass property of the Event Handler.

Event Handler configuration is contained within the Server layer configuration of the network's BacnetComm container, as shown in Figure A-22.

Figure A-22 *Event Summary Alarm Class (BacnetOwsNetwork)*



BacnetOwsNetwork (Bacnet Ows Network)

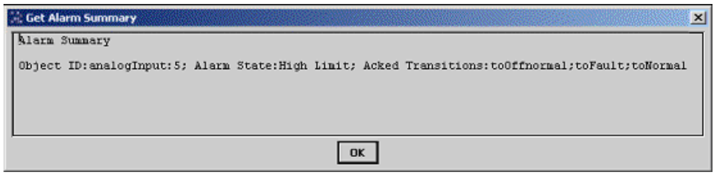
Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [25-Aug-11 3:52 PM EDT]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Bacnet Comm	Bacnet Stack
Comm Control	Enable
Client	Bacnet Client Layer
Server	Bacnet Server Layer
Worker	Bacnet Worker
Event Handler	Event Handler
Silence Supported	false
Event Summary Alarm Class	Default Alarm Class
Event Summary Process Id	0 [0 - max]
Transport	Bacnet Transport Layer

If the GetEventInformation service is not supported, but the GetAlarmSummary service is supported, the events will be retrieved using GetAlarmSummary. The GetAlarmSummary service retrieves only events with a Notify_Type of "`alarm`", so events with a Notify_Type of "`event`" are not retrieved.

Note: *The GetAlarmSummary service has been* deprecated *by the BACnet Committee, and manufacturers should no longer be making devices that use it. Typically, this is seen only in older devices.*

Event summaries retrieved using GetAlarmSummary do not have enough information to generate a valid acknowledgment, so they are not propagated to the Niagara Alarm Service. In this case, a dialog box is presented to the user, such as shown in Figure A-23.

Figure A-23 *Alarm Summary dialog, showing a single object in alarm*



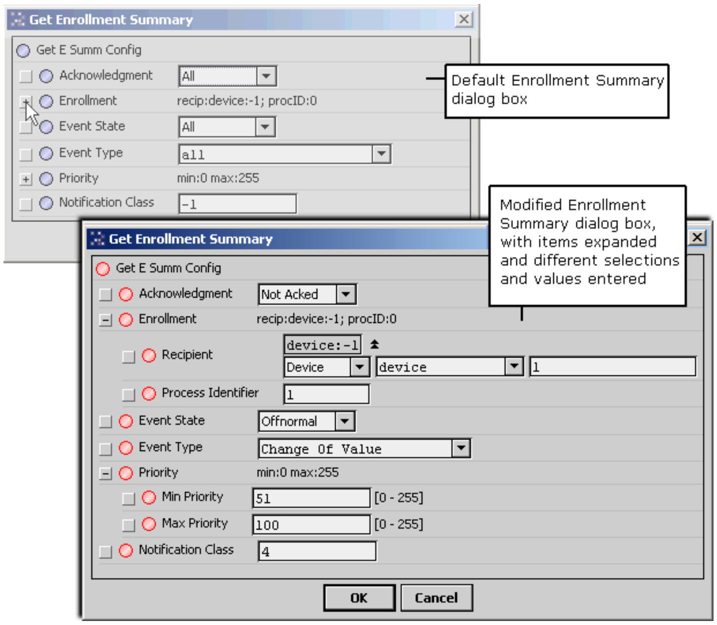
```
Alarm Summary

Object ID:analogInput:5; Alarm State:High Limit; Acked Transitions:toOffnormal;toFault;toNormal
```

Enrollment Summary

ESumm (Also in the **`Bacnet Aws Device Manager`**) This button in the Bacnet Ows Device Manager retrieves a summary of all objects within the device that match a set of filter criteria that you define. When you click the Enrollment Summary button, a filter configuration dialog box is displayed, as shown in Figure A-24.

Figure A-24 *Get Enrollment Summary request configuration dialog*



Get Enrollment Summary

Get E Summ Config

- Acknowledgment: All
- Enrollment: recip:device:-1; procID:0
- Event State: All
- Event Type: all
- Priority: min:0 max:255
- Notification Class: -1

Default Enrollment Summary dialog box

Modified Enrollment Summary dialog box, with items expanded and different selections and values entered

Get Enrollment Summary

Get E Summ Config

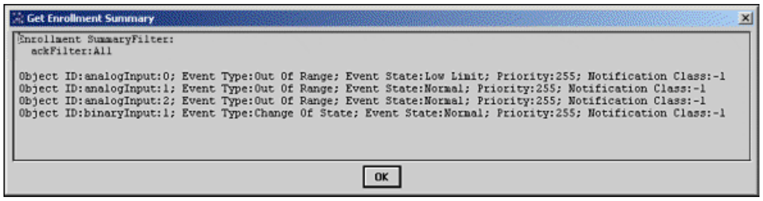
- Acknowledgment: Not Acked
- Enrollment: recip:device:-1; procID:0
  - Recipient: device:-1; Device; device; 1
  - Process Identifier: 1
- Event State: Offnormal
- Event Type: Change Of Value
- Priority: min:0 max:255
  - Min Priority: 51 [0 - 255]
  - Max Priority: 100 [0 - 255]
- Notification Class: 4

Here you configure the various options to define the specific types of objects you wish to locate, and then click **OK**. The device is queried for a list of objects that meet the specified filter criteria, and the result is displayed in a dialog box, as shown in Figure A-25.

Filter options include the following:

- **Acknowledgement**
  Filter on points that are acknowledged, or unacknowledged, or leave the filter open to all points.
- **Enrollment**
  Filter on only event mechanisms that have a specific recipient, definable either by device id or by device address.
- **Event State**
  Specify a particular event state, to restrict the request to include only points that are currently in a specific event state.
- **Event Type**
  Restrict the filter to objects that use a particular event type algorithm.
- **Priority**
  Choose to only include objects where the priority of the last transition is within certain bounds.
- **Notification Class**
  Include only objects and event mechanisms where the notification class used to route the event notifications is equal to a specific number.

Figure A-25 *Example Get Enrollment Summary response dialog*



Get Enrollment Summary

```
Enrollment SummaryFilter:
  ackFilter:All

Object ID:analogInput:0; Event Type:Out Of Range; Event State:Low Limit; Priority:255; Notification Class:-1
Object ID:analogInput:1; Event Type:Out Of Range; Event State:Normal; Priority:255; Notification Class:-1
Object ID:analogInput:2; Event Type:Out Of Range; Event State:Normal; Priority:255; Notification Class:-1
Object ID:binaryInput:1; Event Type:Change Of State; Event State:Normal; Priority:255; Notification Class:-1
```

Additional BACnet OWS Supervisor features

Apart from the different manager view on the **BacnetOwsNetwork**, a BacnetOwsNetwork supports configuration of how many processing threads are used for proxy point polling and proxy point writes. For more details, see "Configurable threads for polling" on page A-12, and "Configurable threads for proxy point writes" on page A-13.

Converting to a BACnet AWS or OWS Supervisor station

Included in the AX-3.6 and later `bacnetAws` and `bacnetOws` modules are three *offline* “station BOG file converter” utilities.

- Two of them convert the `config.bog` database file of a prior **`BACnet Supervisor`** station (running a “Bacnet*Ws*Network”) to one that can run as either a **`BACnet AWS Supervisor`** (running a “Bacnet*Aws*Network”) or as a **`BACnet OWS Supervisor`** (running a Bacnet*Ows*Network).
- There is also a converter for a **`BACnet OWS Supervisor`**, to upgrade its `config.bog` database file to be compatible as a **`BACnet AWS Supervisor`**.

You run these offline converters in the AX-3.6 or later Workbench console, entering a command that specifies the location of the source `config.bog` file. Upon completion, the utility stores a “backup” copy of the original file as `preMigrate_config.bog`. The new `config.bog` is then ready to run as one of the new Supervisor types.

Converting from a BACnet Supervisor (BacnetWsNetwork)

Note: *Remember that different host licensing is required. See “AWS and OWS Supervisor module and license requirements” on page A-3.*

Running the station BOG file offline converter

Step 1 Using Workbench, save the existing **`BACnet Supervisor`** station, such that its `config.bog` file is up-to-date.

Step 2 Using a Workbench platform connection to the Supervisor, stop the station.

Step 3 Copy its entire station folder to the AX-3.6 or later Workbench stations directory.

Step 4 In AX-3.6 or later Workbench, open a console window.
You can use the Workbench menubar option **`Window > Console`** (shortcut F3) to open a console area at the bottom of the Workbench view.

The command prompt shows the current working directory, e.g. `d:\niagara\niagara-3.6.36>`

Step 5 Change directory (**`cd`**) to the station folder under the stations subdirectory, for example:

```
d:\niagara\niagara-3.6.36> cd stations\MyBacSupvsr
```

```
d:\niagara\niagara-3.6.36\stations\MyBacSupvsr
```

Note: *You can skip the step above, but it makes it easier to issue the command. Otherwise, you must include the relative path for the target `config.bog` file in the utility command.*

Step 6 From the stations folder for the BACnet Supervisor, enter *one* of the following commands:

- For a **`BACnet AWS Supervisor`** conversion, enter:

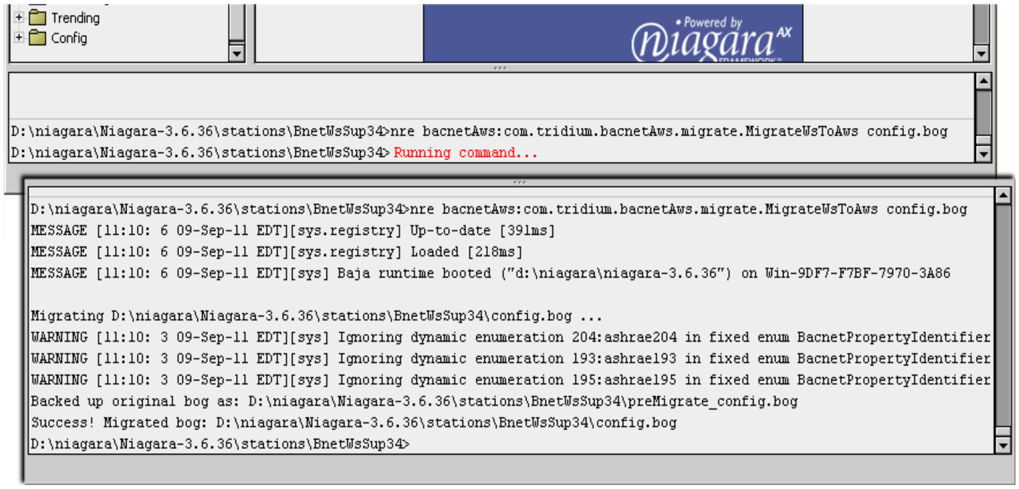
  ```
  nre bacnetAws:com.tridium.bacnetAws.migrate.MigrateWsToAws config.bog
  ```

- Or, for a **`BACnet OWS Supervisor`** conversion, enter:

  ```
  nre bacnetOws:com.tridium.bacnetOws.migrate.MigrateWsToOws config.bog
  ```

This utility starts running, and after several seconds should complete with console results similar to those shown in Figure A-26 below.

Figure A-26 *Example converting BOG file for station BnetWsSup34 to one compatible as AWS Supervisor*



```
D:\niagara\Niagara-3.6.36\stations\BnetWsSup34>nre bacnetAws:com.tridium.bacnetAws.migrate.MigrateWsToAws config.bog
D:\niagara\Niagara-3.6.36\stations\BnetWsSup34> Running command...
```

```
D:\niagara\Niagara-3.6.36\stations\BnetWsSup34>nre bacnetAws:com.tridium.bacnetAws.migrate.MigrateWsToAws config.bog
MESSAGE [11:10: 6 09-Sep-11 EDT][sys.registry] Up-to-date [391ms]
MESSAGE [11:10: 6 09-Sep-11 EDT][sys.registry] Loaded [218ms]
MESSAGE [11:10: 6 09-Sep-11 EDT][sys] Baja runtime booted ("d:\niagara\niagara-3.6.36") on Win-9DF7-F7BF-7970-3A86

Migrating D:\niagara\Niagara-3.6.36\stations\BnetWsSup34\config.bog ...
WARNING [11:10: 3 09-Sep-11 EDT][sys] Ignoring dynamic enumeration 204:ashrae204 in fixed enum BacnetPropertyIdentifier
WARNING [11:10: 3 09-Sep-11 EDT][sys] Ignoring dynamic enumeration 193:ashrae193 in fixed enum BacnetPropertyIdentifier
WARNING [11:10: 3 09-Sep-11 EDT][sys] Ignoring dynamic enumeration 195:ashrae195 in fixed enum BacnetPropertyIdentifier
Backed up original bog as: D:\niagara\Niagara-3.6.36\stations\BnetWsSup34\preMigrate_config.bog
Success! Migrated bog: D:\niagara\Niagara-3.6.36\stations\BnetWsSup34\config.bog
D:\niagara\Niagara-3.6.36\stations\BnetWsSup34>
```

Results with “Success” mean you can start and run the converted **Bacnet Supervisor** station as either a **BACnet AWS Supervisor** or **BACnet OWS Supervisor**, as appropriate.

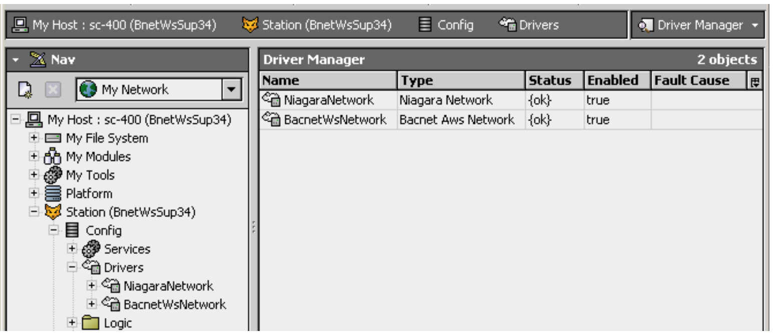
Note: *A few warnings may occur in the conversion, as shown in Figure A-26 above. In this case, this occurred because the source BACnet Supervisor station database was from a host running AX-3.4—and it had uploaded BACnet properties that were “unknown” to that Bacnet driver version, but known in AX-3.6.*

Notes on a converted BACnet Supervisor station

The following notes apply to a **BACnet Supervisor** station that has been converted to a **BACnet AWS Supervisor** or **BACnet OWS Supervisor** station:

- Following the conversion, components like the BacnetWsNetwork and BacnetWsDevice (from the `bacnetws` module) are changed to the BacnetAwsNetwork or BacnetAwsDevice, and so on. However, *names* of all components *remain unchanged*. This means if you used default names, like “BacnetWsNetwork” for the top network component, it remains that way. See Figure A-27.

Figure A-27 Converted AWS Supervisor with BacnetAwsNetwork named “BacnetWsNetwork”



Name	Type	Status	Enabled	Fault Cause
NiagaraNetwork	Niagara Network	{ok}	true	
BacnetWsNetwork	Bacnet Aws Network	{ok}	true	

To reduce confusion, you should *rename* such components as appropriate. Note that the default views for such changed components will automatically be the correct ones—for example, the **Bacnet Aws Device Manager**, the **Bacnet Aws Config Manager**, and so forth.

Converting from a BACnet OWS Supervisor to a BACnet AWS Supervisor

Note: *Remember that different host licensing is required. See “AWS and OWS Supervisor module and license requirements” on page A-3.*

Running the station BOG file offline converter

Step 1 Using Workbench, save the existing **BACnet OWS Supervisor** station, such that its `config.bog` file is up-to-date.

Step 2 Using a Workbench platform connection to the Supervisor, stop the station.

Step 3 If not already present, copy its entire station folder to the AX-3.6 or later Workbench stations directory.

Step 4 In AX-3.6 or later Workbench, open a console window.
You can use the Workbench menubar option **Window > Console** (shortcut F3) to open a console area at the bottom of the Workbench view.
The command prompt shows the current working directory, e.g. `d:\niagara\niagara-3.6.36>`

Step 5 Change directory (**cd**) to the station folder under the stations subdirectory, for example:

```
d:\niagara\niagara-3.6.36> cd stations\BnetSup_OWS

d:\niagara\niagara-3.6.36\stations\BnetSup_OWS
```

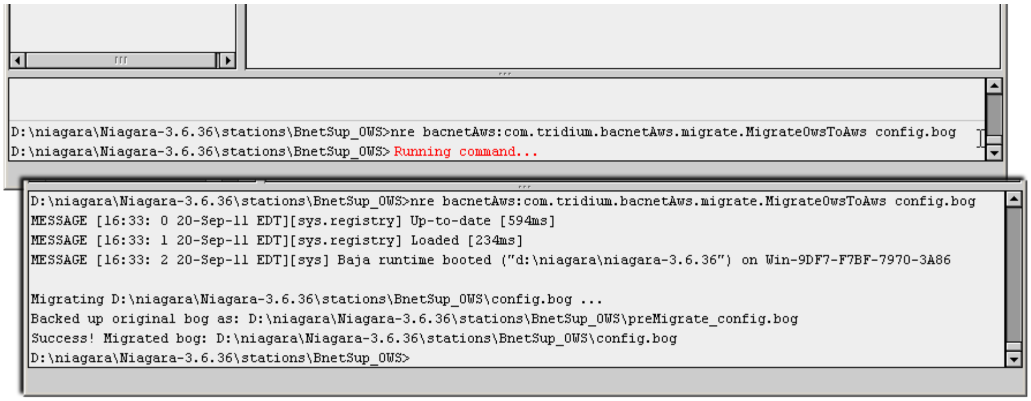
Note: *You can skip the step above, but it makes it easier to issue the command. Otherwise, you must include the relative path for the target* `config.bog` *file in the utility command.*

Step 6 From the stations folder for the BACnet OWS Supervisor, enter the following command:

```
nre bacnetAws:com.tridium.bacnetAws.migrate.MigrateOwsToAws config.bog
```

This utility starts running, and after several seconds should complete with console results similar to those shown in Figure A-28.

Figure A-28 *Example converting BOG file for station BnetSup_OWS to one compatible as AWS Supervisor*



```
D:\niagara\Niagara-3.6.36\stations\BnetSup_OWS>nre bacnetAws:com.tridium.bacnetAws.migrate.MigrateOwsToAws config.bog
D:\niagara\Niagara-3.6.36\stations\BnetSup_OWS> Running command...
```

```
D:\niagara\Niagara-3.6.36\stations\BnetSup_OWS>nre bacnetAws:com.tridium.bacnetAws.migrate.MigrateOwsToAws config.bog
MESSAGE [16:33: 0 20-Sep-11 EDT][sys.registry] Up-to-date [594ms]
MESSAGE [16:33: 1 20-Sep-11 EDT][sys.registry] Loaded [234ms]
MESSAGE [16:33: 2 20-Sep-11 EDT][sys] Baja runtime booted ("d:\niagara\niagara-3.6.36") on Win-9DF7-F7BF-7970-3A86

Migrating D:\niagara\Niagara-3.6.36\stations\BnetSup_OWS\config.bog ...
Backed up original bog as: D:\niagara\Niagara-3.6.36\stations\BnetSup_OWS\preMigrate_config.bog
Success! Migrated bog: D:\niagara\Niagara-3.6.36\stations\BnetSup_OWS\config.bog
D:\niagara\Niagara-3.6.36\stations\BnetSup_OWS>
```

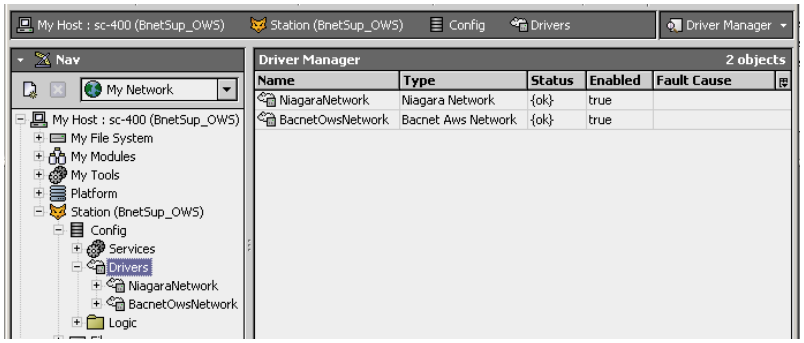
Results with “Success” mean you can start and run the converted **`Bacnet OWS Supervisor`** station as a **`BACnet AWS Supervisor`** station.

Notes on a converted BACnet OWS Supervisor station

The following notes apply to a **`BACnet OWS Supervisor`** station that has been converted to a **`BACnet AWS Supervisor`** station:

- Following the conversion, components like the BacnetOwsNetwork and BacnetDevice (from the `bacnetOws` and `bacnet` modules) are changed to the BacnetAwsNetwork or BacnetAwsDevice, and so on. However, *names* of all components *remain unchanged*. This means if you used default names, like “BacnetOwsNetwork” for the top network component, it remains that way. See Figure A-29.

Figure A-29 *Converted AWS Supervisor with BacnetAwsNetwork named “BacnetOwsNetwork”*



Driver Manager — 2 objects

Name	Type	Status	Enabled	Fault Cause
NiagaraNetwork	Niagara Network	{ok}	true	
BacnetOwsNetwork	Bacnet Aws Network	{ok}	true	

To reduce confusion, you should *rename* such components as appropriate. Note that the default views for such changed components will automatically be the correct ones—for example, the **`Bacnet Aws Device Manager`**, the **`Bacnet Aws Config Manager`**, and so forth.

APPENDIX B

BACnet Supervisor

Note: *Concurrent with this document update in the AX-3.6 maintenance release timeframe, the "***BACnet Supervisor***" based on the* bacnetws *module is being* superseded *in favor of two new Supervisor products, each of which are BTL (BACnet Testing Laboratories) listed. An offline utility allows migration of a* **BACnet Supervisor** *database to one compatible as a* **BACnet AWS Supervisor** *or* **BACnet OWS Supervisor***. See "BACnet AWS and OWS Supervisors" on page A-1 for further details.*

This appendix describes the **BACnet Supervisor***, which* is not BTL listed*. Sale of this product* stops *with the release of the BACnet AWS and BACnet OWS Supervisors. The* bacnetws *module will continue to be available through the 3.7 release of NiagaraAX, with the following caveats:*

- Bug fixes will not be implemented, barring some extraordinary issue. Therefore, the resolution path for virtually any problem is to upgrade a BACnet Supervisor to either a BACnet AWS Supervisor or BACnet OWS Supervisor.
- New features and enhancements will become available only in the BACnet AWS and BACnet OWS Supervisor products.

For these reasons, it is strongly recommended to migrate away from bacnetws*-based products, either in usage or in development, at the earliest opportunity.*

In a few cases where a **BACnet Supervisor** *was implemented using BACnet server functions (exporting BACnet objects), a license exception may allow migration to a* **BACnet AWS Supervisor** *with this same ability. However (at the time of this document), this would invalidate its BTL listing. Although BTL testing and listing of server operation may be added at some future time, the currently completed testing and listing applies only to BACnet client operation.*

This appendix describes the features and functionality specific to a BACnet Supervisor, and has the following main sections:

- BACnet Supervisor overview
- Bacnet Ws Device Manager
- LocalBacnetWsDevice additions
- BacnetWsDevice additions

BACnet Supervisor overview

The BACnet Supervisor uses much of the same software as the regular BACnet driver. A few extensions are made to allow enhanced operations. Instead of a BacnetNetwork, you use a **BacnetWsNetwork** (from the bacnetws module) as the root container for BACnet operations.

Using its **Bacnet Ws Device Manager** view, you add **BacnetWsDevices** instead of BacnetDevices. Because the BacnetWsNetwork is also a BacnetNetwork, you can also manually copy regular BacnetDevices into the network—however, in AX-3.4 or earlier, you will not be able to perform the supervisory-related functions on them.

Note: *Starting in AX-3.5, device type defaults to BacnetDevice, which does allow supervisory-related functions.*

In AX-3.4 and earlier, the distinction between a BacnetDevice and BacnetWsDevice is purely a Niagara difference—this simply identifies what properties the Niagara representation will have, and what functions may be done with the device. If a target BacnetWsDevice cannot support a specific supervisory request, it will return the appropriate error, meaning that the action will fail with an understandable message dialog.

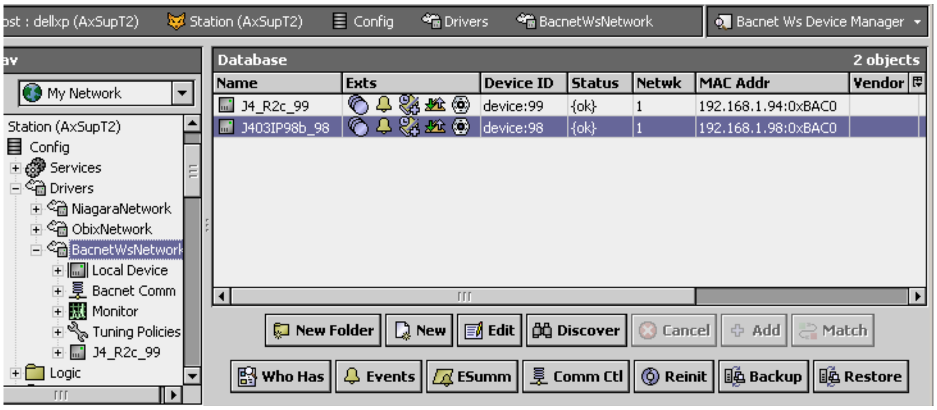
To use the BACnet Supervisor, you need to have a couple of things in place:

- The Supervisor PC must have both the bacnet and bacnetws modules in its modules folder (also applies for any other Workbench PC with which you access the BACnet Supervisor).
- The Supervisor PC must have both the bacnet feature and bacnetws feature in its Tridium license. Note that license may also have limits on the number of devices, points, and so forth.

Bacnet Ws Device Manager

The Bacnet Ws Device Manager (Figure B-1) is the default view for the BacnetWsNetwork, and provides all the same features as the regular Bacnet Device Manager.

Figure B-1 *Bacnet Ws Device Manager is default view for BacnetWsNetwork*



Name	Exts	Device ID	Status	Netwk	MAC Addr	Vendor
J4_R2c_99		device:99	{ok}	1	192.168.1.94:0xBAC0	
J403IP98b_98		device:98	{ok}	1	192.168.1.98:0xBAC0	

You discover BACnet devices in this view the same as in the **`Bacnet Device Manager`** view. For details, see "About Bacnet Device Find Parameters" on page 3-27.

When doing this, the **`Add`** dialog provides the same parameters for adding BacnetWsDevices as for BacnetDevices. See "BacnetDevice properties" on page 3-31.

This device manager view of this BACnet Supervisor network provides several additional buttons in the button bar. These correspond to additional functions that can be performed on BacnetWsDevices.

These additional buttons available in the Bacnet Ws Device Manager are:

- **`Events`** — See "Event Information" on page B-2.
- **`ESumm`** — See "Enrollment Summary" on page B-3.
- **`Comm Ctl`** — See "Communication Control" on page B-4.
- **`Reinit`** — See "Reinitialize Device" on page B-5.
- ◷ — See "Time Synchronization" on page B-5.
- **`Backup`** — (AX-3.2 and later only) See "Backup Device" on page B-6.
- **`Restore`** — (AX-3.2 and later only) See "Restore Device" on page B-7.

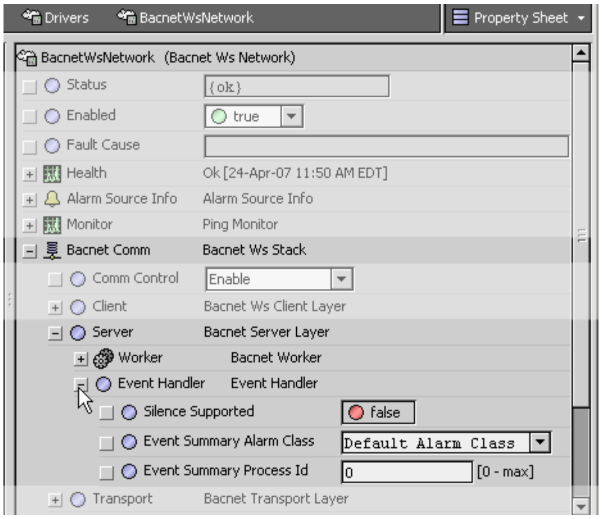
Event Information

Events This button in the Bacnet Ws Device Manager retrieves event information from the selected device, requesting all outstanding events from the BACnet device. The purpose of this feature is that if Niagara somehow missed receiving an event, you can still retrieve and acknowledge this event using the Event Information service. For example, the Niagara station may not have been running at the time the original event occurred.

If the GetEventInformation service is supported, this will be used to retrieve events. The events retrieved in this manner contain enough information to generate a valid acknowledgment to the remote device. These event summaries will be propagated to the Niagara Alarm Service through the alarm class assigned in the eventSummaryAlarmClass property of the Event Handler.

Event Handler configuration is contained within the Server layer configuration, as shown in Figure B-2.

Figure B-2 *Event Summary Alarm Class*



BacnetWsNetwork (Bacnet Ws Network)

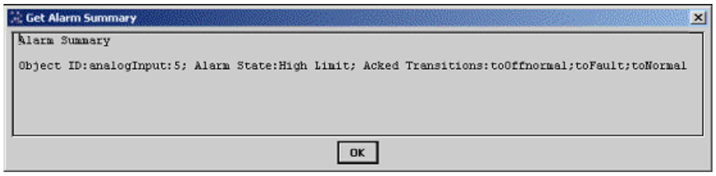
- Status: {ok}
- Enabled: true
- Fault Cause:
- Health: Ok [24-Apr-07 11:50 AM EDT]
- Alarm Source Info: Alarm Source Info
- Monitor: Ping Monitor
- Bacnet Comm: Bacnet Ws Stack
  - Comm Control: Enable
  - Client: Bacnet Ws Client Layer
  - Server: Bacnet Server Layer
    - Worker: Bacnet Worker
    - Event Handler: Event Handler
      - Silence Supported: false
      - Event Summary Alarm Class: Default Alarm Class
      - Event Summary Process Id: 0 [0 - max]
  - Transport: Bacnet Transport Layer

If the GetEventInformation service is not supported, but the GetAlarmSummary service is supported, the events will be retrieved using GetAlarmSummary. The GetAlarmSummary service retrieves only events with a Notify_Type of "`alarm`", so events with a Notify_Type of "`event`" are not retrieved.

Note: *The GetAlarmSummary service has been* deprecated *by the BACnet Committee, and manufacturers should no longer be making devices that use it. Typically, this is seen only in older devices.*

Event summaries retrieved using GetAlarmSummary do not have enough information to generate a valid acknowledgment, so they are not propagated to the Niagara Alarm Service. In this case, a dialog box is presented to the user, such as shown in Figure B-3.

Figure B-3 *Alarm Summary dialog, showing a single object in alarm*

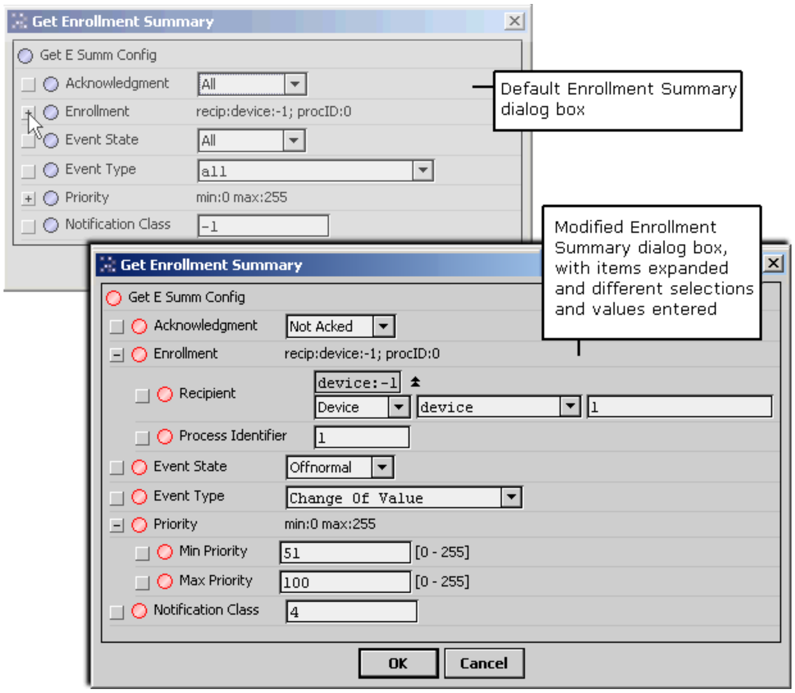


```
Alarm Summary
Object ID:analogInput:5; Alarm State:High Limit; Acked Transitions:toOffnormal;toFault;toNormal
```

Enrollment Summary

ESumm This button in the Bacnet Ws Device Manager retrieves a summary of all objects within the device that match a set of filter criteria that you define. When you click the Enrollment Summary button, a filter configuration dialog box is displayed, as shown in Figure B-4.

Figure B-4 *Get Enrollment Summary request configuration dialog*



Get Enrollment Summary

○ Get E Summ Config
- □ ○ Acknowledgment: All
- ⊞ ○ Enrollment: recip:device:-1; procID:0
- □ ○ Event State: All
- □ ○ Event Type: all
- ⊞ ○ Priority: min:0 max:255
- □ ○ Notification Class: -1

Default Enrollment Summary dialog box

Get Enrollment Summary

○ Get E Summ Config
- □ ○ Acknowledgment: Not Acked
- ⊟ ○ Enrollment: recip:device:-1; procID:0
  - □ ○ Recipient: device:-1 / Device / device / 1
  - □ ○ Process Identifier: 1
- □ ○ Event State: Offnormal
- □ ○ Event Type: Change Of Value
- ⊟ ○ Priority: min:0 max:255
  - □ ○ Min Priority: 51 [0 - 255]
  - □ ○ Max Priority: 100 [0 - 255]
- □ ○ Notification Class: 4

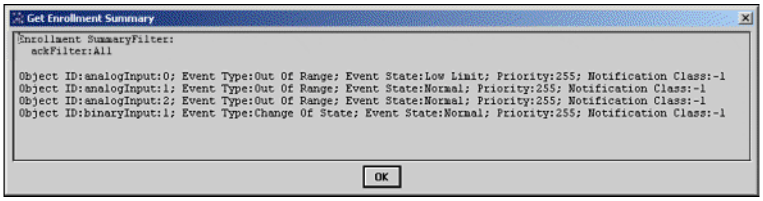
Modified Enrollment Summary dialog box, with items expanded and different selections and values entered

Here you configure the various options to define the specific types of objects you wish to locate, and then click **OK**. The device is queried for a list of objects that meet the specified filter criteria, and the result is displayed in a dialog box, as shown in Figure B-5.

Filter options include the following:

- **Acknowledgement**
  Filter on points that are acknowledged, or unacknowledged, or leave the filter open to all points.
- **Enrollment**
  Filter on only event mechanisms that have a specific recipient, definable either by device id or by device address.
- **Event State**
  Specify a particular event state, to restrict the request to include only points that are currently in a specific event state.
- **Event Type**
  Restrict the filter to objects that use a particular event type algorithm.
- **Priority**
  Choose to only include objects where the priority of the last transition is within certain bounds.
- **Notification Class**
  Include only objects and event mechanisms where the notification class used to route the event notifications is equal to a specific number.

Figure B-5 *Example Get Enrollment Summary response dialog*



Get Enrollment Summary

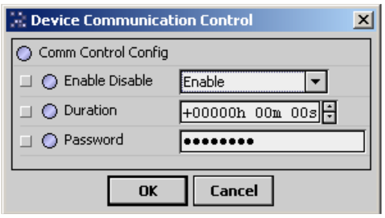
```
Enrollment SummaryFilter:
  ackFilter:All

Object ID:analogInput:0; Event Type:Out Of Range; Event State:Low Limit; Priority:255; Notification Class:-1
Object ID:analogInput:1; Event Type:Out Of Range; Event State:Normal; Priority:255; Notification Class:-1
Object ID:analogInput:2; Event Type:Out Of Range; Event State:Normal; Priority:255; Notification Class:-1
Object ID:binaryInput:1; Event Type:Change Of State; Event State:Normal; Priority:255; Notification Class:-1
```

Communication Control

Comm Ctl This button in the Bacnet Ws Device Manager allows you to control a device's ability to generate traffic on the network. This could be useful in a diagnostic situation, where you want to temporarily eliminate all traffic except from a single device. Or you may have a faulty device that is sending extraneous data, and you wish to "silence it" until you can identify the cause of the problem. When you click this button, a dialog box appears which allows you to configure the request, as shown in Figure B-6.

Figure B-6 *Device Communication Control configuration dialog*



○ Comm Control Config

□ ○ Enable Disable: Enable

□ ○ Duration: +00000h 00m 00s

□ ○ Password: ••••••••

Configuration options include the following:

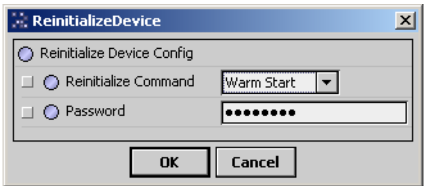
- **Enable Disable**
  You can choose to Enable or Disable the device from sending any BACnet requests. You can also choose to “Disable Initiation,” which allows the device to respond to BACnet requests, while being prevented from initiating any requests of its own. Note that the sending of a single I-Am request is allowed if a Who-Is request that matches the device’s device id is received.
- **Duration**
  You can configure the length of time for which communications will be disabled, after which the communications will automatically be re-enabled for the device.
- **Password**
  Some devices may require a password to disable their communications. If a device requires a password, you may enter it here.

When you click **OK**, a request to control the communication is sent to the device. The result is displayed in a dialog box, with either a success message, or a message including the error returned by the device.

Reinitialize Device

Reinit This button in the Bacnet Ws Device Manager allows you to restart a device. In the popup dialog (Figure B-7), you may choose either “Cold Start” or “Warm Start.”

Figure B-7 *Reinitialize Device configuration dialog*



○ Reinitialize Device Config

□ ○ Reinitialize Command: Warm Start

□ ○ Password: ••••••••

Note: The specific meaning of terms Cold Start and Warm Start is left up to the device’s manufacturer to define, so make sure you understand exactly what each procedure entails for the device in question.

As shown above, this dialog has two fields:

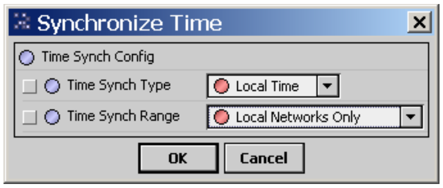
- **Reinitialize Command**
  Select either Warm Start (default) or Cold Start (see Note: above).
- **Password**
  Enter the password required by the device to invoke this command. Some devices may not require a password, so if you do not enter anything, no password is sent to the device.

When you click **OK**, the reinitialize request is sent to the device. The result is displayed in a dialog box, as either a success or failure.

Time Synchronization

This button in the Bacnet Ws Device Manager is available only on the tool bar and the menu bar. It allows you to send a one-time time synchronization message. When you click it, you are first asked to confirm this action, as it will send the current station time to all controllers on the network. If you choose Yes, you are presented with a dialog, shown in Figure B-8.

Figure B-8 *Time Synchronization configuration dialog*



Time Synch Config

Time Synch Type: Local Time

Time Synch Range: Local Networks Only

As shown above, this dialog has two fields:

- **Time Synch Type**
  Select either Local Time (default), or UTC Time.
- **Time Synch Range**
  Select either Local Networks Only (default), or All Connected Networks.

When you click OK, a time synchronization request is then broadcast to the network. This is a one-time message. If you wish to schedule periodic time synchronization, you can configure this in the Local-Bacnet Device.

Backup Device

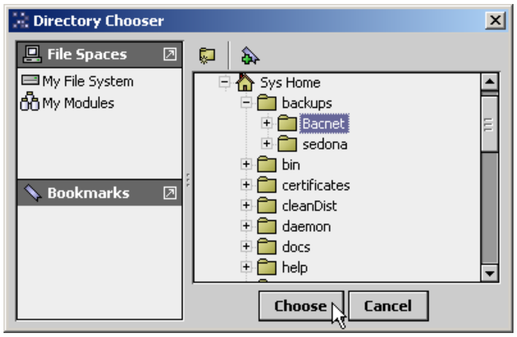
Backup (AX-3.2 and later only) This button in the Bacnet Ws Device Manager allows you to backup the selected device's configuration, as one or more "restorable" files on the BACnet Supervisor PC. If needed, you can use the Restore function later to reinstall this backup.

Note: *The target device must support the BACnet DM-BR-B BIBB, as part of the B-BC device profile conformance. Both the BACnet specification 135-2008 and addendum 135-2008n are supported.*

In addition, the device may require a password before it initiates a backup or a restore. Otherwise, a backup or restore job will immediately fail, showing the associated reason in the job log details. For example, you may see "Unrecognized Service" or "Security: Password Failure".

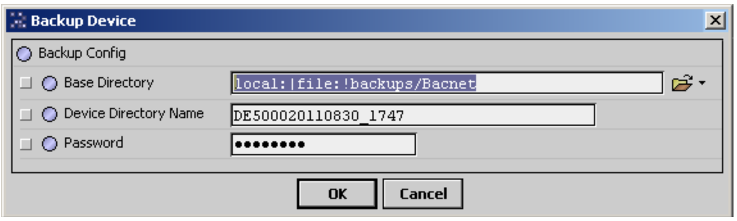
When you click `Backup`, the standard `Directory Chooser` appears, in which you specify the target directory for the backup file(s). If needed, use the "new folder" control, as was done in the example shown in Figure B-9 to make a "`Bacnet`" folder under the system "`backups`" folder.

Figure B-9 *Directory Chooser dialog to specify backup directory*



After you choose the target directory, the `Backup Device` dialog appears, as shown in Figure B-10.

Figure B-10 *Backup Device configuration dialog*



Backup Config

Base Directory: local:|file:!backups/Bacnet

Device Directory Name: DE500020110830_1747

Password: ••••••••

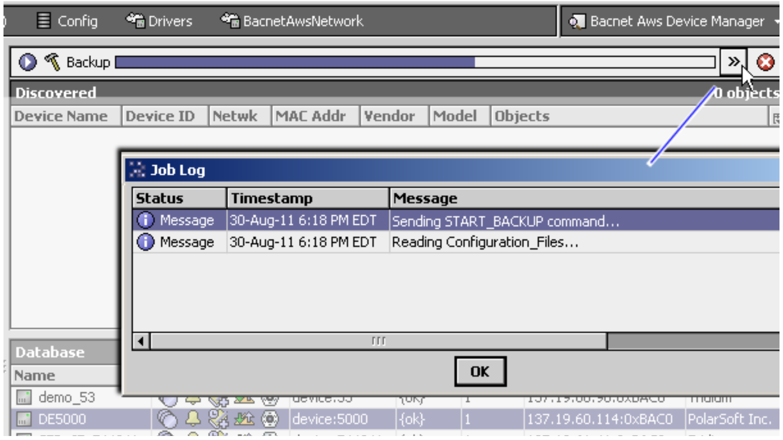
Configuration fields in this dialog include the following:

- **Base Directory**
  Reflects the ord for the local target directory, as previously chosen in the `Directory Chooser`. If needed, you can modify it using file ord syntax.

- **Device Directory Name**
  This specifies the subdirectory that will be made under the base directory for the device's backup file(s). This defaults to *<BacnetWsDeviceName><YYYYMMDD_HHMM>*, where the last portion is a timestamp that reflects when the backup was initiated. If desired, you can modify.
- **Password**
  Some devices may require a password to initiate a backup and/or a restore. If a device requires a password, you may enter it here.

When you click **OK**, the backup request is sent to the device. A Backup job is started, and as shown in Figure B-11 you can click on the job log control near the top of the manager to see the progress of the backup.

Figure B-11 *Backup job started, Job Log details dialog*



Status	Timestamp	Message
Message	30-Aug-11 6:18 PM EDT	Sending START_BACKUP command...
Message	30-Aug-11 6:18 PM EDT	Reading Configuration_Files...

When the job completes, it will post a success or failed status. A successful backup has the specified device subdirectory under the target base directory, containing one or more backup files for the device.

Note: *An invalid backup job typically fails immediately. However, a valid backup job may take several minutes to complete, depending on the implementation in the target BACnet device. Typically, the device executes some "preparation routine" first, before assembling and sending the backup files. In the case of AX-3.2 or higher stations running the Bacnet driver, there is a set server routine as well as associated properties, found in the LocalBacnetDevice of the BacnetNetwork. For more details, see "Local Device backup and restore properties" on page 4-80.*

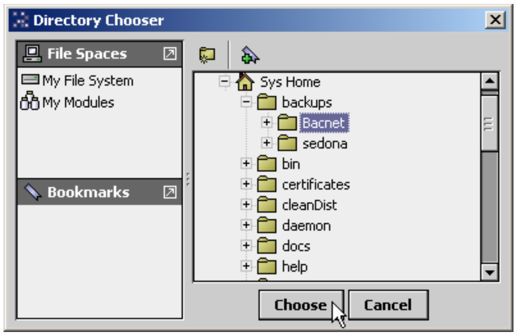
Restore Device

Restore (AX-3.2 and later only) This button in the Bacnet Ws Device Manager allows you to restore a previous backup to the selected device, where a backup is one or more "restorable" files saved on the BACnet Supervisor PC. See "Backup Device" on page B-6.

Note: *The target device must support the BACnet DM-BR-B BIBB, as part of the B-BC device profile conformance. In addition, you may require a device password to have it initiate a backup or a restore. Otherwise, a backup or restore job will immediately fail, showing the associated reason in the job log details. For example, you may see "Unrecognized Service" or "Security: Password Failure".*

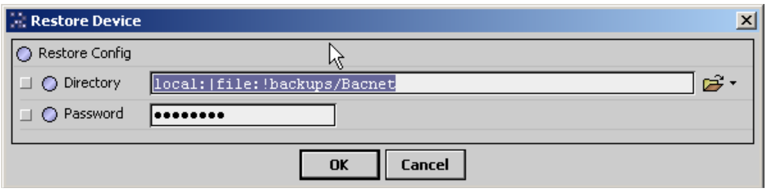
When you click **Restore**, the standard **Directory Chooser** appears, in which you navigate to the directory that contains the backup file(s), as shown in Figure B-12.

Figure B-12 Directory Chooser dialog to specify directory with backup file(s)



After you choose the source directory, the **`Restore Device`** dialog appears, as shown in Figure B-13.

Figure B-13 Restore Device configuration dialog



○ Restore Config
□ ○ Directory `local:|file:!backups/Bacnet`
□ ○ Password ••••••••

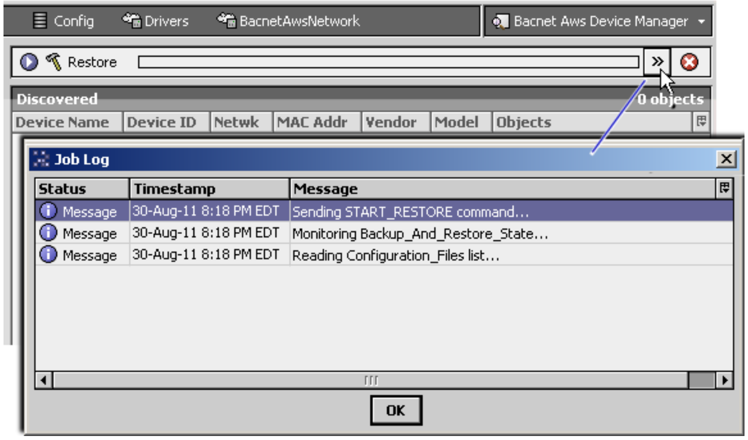
Configuration fields in this dialog include the following:

- **Directory**
  Reflects the ord for the local source directory, as previously chosen in the **`Directory Chooser`**. If needed, you can modify it using file ord syntax.
- **Password**

  Some devices may require a password to initiate a backup and/or a restore. If a device requires a password, you may enter it here.

When you click **`OK`**, the restore backup request is sent to the device. A Restore job is started, and as shown in Figure B-14 you can click on the job log control near the top of the manager to see the job progress.

Figure B-14 Restore job started, Job Log details dialog



Status	Timestamp	Message
Message	30-Aug-11 8:18 PM EDT	Sending START_RESTORE command…
Message	30-Aug-11 8:18 PM EDT	Monitoring Backup_And_Restore_State…
Message	30-Aug-11 8:18 PM EDT	Reading Configuration_Files list…

When the job completes, it will post a success or failed status. Successful restores will install the backup file(s) found in the source directory, and typically re-initialize (reboot) the device.

Note: *An invalid restore job typically fails immediately. However, a valid restore job may take several minutes to complete, depending on the implementation in the target BACnet device. Typically, the device executes some "preparation routine" first, before accepting the backup files. In the case of AX-3.2 or later stations running the Bacnet driver, there is a set routine as well as associated properties, found in the LocalBacnetDevice of the BacnetNetwork. For details, see "Local Device backup and restore properties" on page 4-80.*

LocalBacnetWsDevice additions

The LocalBacnetWsDevice (**`Local Device`**) component represents the configuration of the NiagaraAX station as it appears as a BACnet device on the BACnet network. It includes all the same child slots, properties, and actions as the LocalBacnetDevice component in the Bacnet driver. For details, see related sections in "Niagara Bacnet Server Operation" on page 4-59.

Each LocalBacnetWsDevice also has *additional* properties relating to operation as the BACnet time-synch master, to send periodic time synch messages to defined BACnet recipient devices.

Note: *Starting in AX-3.5, this time synch master functionality was added to the standard BACnet driver (BacnetNetwork), using the same collection of related properties. For details, see "Time synchronization properties" on page 3-22.*

BacnetWsDevice additions

Each BacnetWsDevice component added using the Bacnet Ws Device Manager contains the same set of child slots, properties, and actions as a BacnetDevice component. For details, see related sections in "Niagara Bacnet Client Concepts" on page 3-11.

Each BacnetWsDevice also has the following available additional actions:

- **Comm Control**
  Also available from the Bacnet Ws Device Manager. See "Communication Control" on page B-4.
- **Reinitialize Device**
  Also available from the Bacnet Ws Device Manager. See "Reinitialize Device" on page B-5.
- **Get Enrollment Summary**
  Also available from the Bacnet Ws Device Manager. See "Enrollment Summary" on page B-3.
- **Get Event Information**
  Also available from the Bacnet Ws Device Manager. See "Event Information" on page B-2.

Note: *Starting in AX-3.5, the default device under a* **`BacnetWsNetwork`** *is the standard* **`BacnetDevice`**, *which does not offer the additional actions listed above. However, those same actions* are available *in buttons in the* **`BacnetWs Device Manager`** *view, with any device selected. See "Bacnet Ws Device Manager" on page B-2.*

APPENDIX C

Internetworks and BACnet/IP

The BACnet standard defines an “internetwork” as a set of two or more (BACnet) networks connected by (BACnet) routers. This appendix explains topics relating to BACnet internetworks and Niagara station configuration. Also provided are BBMD-related details that apply only to a BACnet/IP network.

The following main topics are included:

BACnet Network Numbers

In all NiagaraAX releases of the BACnet driver, support is provided for more than one link layer type, using multiple “network ports” under the Bacnet network’s BacnetComm, Network child container. BACnet/IP is the typical and default network port (IpPort) included in a new Bacnet network, with other port types available (EthernetPort, MstpPort) in the `bacnet` palette. The MstpPort, for support of a BACnet MS/TP device trunk, requires a station hosted by a QNX-based JACE.

Each port added must reference a different BACnet network number across a job’s “internetwork”. The range for BACnet network numbers is 1—65535. Enter this number in each port component’s property sheet, in the “Network Number” field (by default, this value is “-1”, which means undefined/inactive).

If installing a Niagara station on an existing BACnet internetwork, where one or more BACnet routers already exist, you need to know the assigned network number(s), and enter (as appropriate) in the network port(s) under the BacnetComm, Network container. Find the existing network number information in the configuration setup of the BACnet routers.

Note a station’s Bacnet network is not limited to only BACnet networks defined in its own network ports—as it automatically “learns” other remote networks, from global BACnet “I-Am” messages received from other devices (and routers) on remote networks. The station maintains a table of known BACnet networks under the network’s BacnetComm, Network child, in a “Router Table” component.

By default, the station automatically performs BACnet router functions across different BACnet networks. Often this is the desired configuration, especially when there are MS/TP trunks that attach to RS-485 ports on a JACE controller. However, multiple BACnet routes to the same network segment results in message flood issues, and is considered a misconfiguration. If this is detected, the station automatically disables BACnet routing by setting the “Routing Enabled” property (found in the Bacnet network’s BacnetComm, Network slot) to false. For related details, see the next section “Internetwork = BACnet Routers” on page C-2.

Internetwork = BACnet Routers

As previously mentioned, the Bacnet network maintains a table of BACnet routers, in the BacnetComm, Network component. Included is the ability to modify, add, or remove entries representing BACnet routers. See "About Bacnet Comm: Network: Router Table" on page 3-14.

Typically you should not need to edit this table, which is automatically populated when you perform a Discover from the Bacnet network, or perform a "Who Has" command.

Internetwork Rules

When implementing a BACnet internetwork there are several rules that must be followed:

- Rule: There must be only *one* message path between any two BACnet devices on an internetwork. No communication "loops" are allowed.
  - Example: You should not configure multiple BACnet devices on the same LAN with both BACnet/IP and BACnet/Ethernet enabled—note this includes Niagara stations (hosts) as well.
- Rule: A BACnet router must exist between two different BACnet networks (different network numbers) for BACnet messages to pass between the devices on the two networks. This applies to any link-layer types.
  - Concept: A BACnet router can be a single-purpose device, or an "application layer" device (controller) that also includes router functionality, such as a station running in a Niagara host.
    Note: *With recent changes to the BACnet Specification,* all *devices communicating on the BACnet network* must *have an application layer entity, including a BACnet Device object.*

    By default, if a Niagara station is configured for multiple link-layers, it automatically acts as a router between those 2 (or more) BACnet networks to which it is directly attached.
    For example, if an EthernetPort and 2 MstpPorts are all enabled in a JACE station, it operates as a BACnet router between those 3 networks. If Ethernet, IP, and a MSTP port are enabled, the station operates as a BACnet-to-BACnet/IP router between those 3 networks, and so forth.
    Note that if a second IpPort is added, bound to a different UDP port, the station routes BACnet messages between those networks, just like other data link layers.
    However, note that a Niagara station does not act as a "plain IP router".
- Rule: Within any given internetwork, each BACnet network must have unique network number, from 1 to 65534.
  - Concept: Each link-layer (network port) that is enabled in the Bacnet network's BacnetComm, Network container corresponds to a specific BACnet network, specified in the "Network Number" property.
    – If establishing a new BACnet internetwork, you can specify whatever network number for each port. For example, BACnet/IP (IpPort) = 1, BACnet/Ethernet (EthernetPort) = 2, and (if a QNX-based JACE), MstpPort (1) = 3, MstpPort (2) = 4, and so on.
    – If adding a JACE on an existing BACnet internetwork, you should specify the established BACnet/IP network number and/or BACnet/Ethernet network number currently in use. If a JACE with one or more enabled MstpPorts, you must also specify a previously unused network number for each one.
- Rule: Every BACnet object, including the Device object in each BACnet device, must have a unique numeric Object_Identifier. In the specific case of the Device object, this Object_Identifier must be unique, internetwork-wide. Valid values for this identifier range from 0 to 4194302.
  - Concept: BACnet devices respond to a Who-Is broadcast messge with an I-am message that includes each Device object's unique numeric Object_Identifier. If you receive duplicate I-am messages (multiples showing same device, by number), it means either that more than one device has been assigned that same identifier (number), or that a message loop exists.

BACnet Router Functions

A full listing of all BACnet router functions is extensive—see the BACnet specification for complete details on routers. From a general perspective, BACnet routers are required to pass BACnet messages between different BACnet networks. This applies to all directed messages as well as broadcast messages.

Usually, a BACnet router is used to join networks of different media/link-layer types, for example a router with RS-485 MS/TP port(s) and an Ethernet-BACnet/IP port. For router examples in diagrams, see "Example Internetwork Diagrams" on page C-8.

BACnet/IP and BBMDs

NiagaraAX BACnet/IP support includes BBMD-hosting capability. If a station is configured for BBMD operation, you can review (and if necessary, modify) the broadcast distribution table common to all BBMDs on the B/IP network, as well as the foreign device table used in the local Niagara station.

The following sections discuss BBMD-related topics:

- About BBMDs
- Foreign Device Table

About BBMDs

Use of BBMDs (BACnet broadcast management device) is the answer to a BACnet/IP network that needs to span across multiple IP subnets. Each IP subnet with BACnet/IP devices requires one (and only one) BBMD.

A BBMD may be a device operating solely as a BBMD, or more typically, include BBMD functions in addition to other application/controller duties. This is the case with a NiagaraAX station running a Bacnet network with its IpPort, Link container's "Ip Device Type" property set to "Bbmd" (instead of the default "Standard", or "Foreign Device").

Note: *Often an installation with BBMDs is not an "internetwork," but have only one BACnet/IP network. If so, Niagara stations on different subnets should specify the same network number for their associated IpPort.*

Why and What a BBMD Does

A BBMD is used to support delivery of BACnet broadcast messages, such as "Who-Is" and "Who-Has." As a rule, globally broadcast messages are inherently blocked by standard IP routers, which are used to connect separate IP subnets. Note this issue does not exist for any "directed message" between devices on different subnets, such as a common ReadProperty request—which is not a broadcast message.

The BBMD resolves this issue by acting as a broadcast manager for its subnet, working in coordination with other "peer" BBMDs. Each other IP subnet that contains BACnet/IP devices has one BBMD. Each BBMD stores a table with the IP address and distribution mask of all BBMDs, itself included. This is called the BBMD's "BDT" (Broadcast Distribution Table), and is identical[1] in each BBMD for that entire BACnet/IP network.

When a global broadcast message is sent, it is automatically received by all devices on its local subnet, including the BBMD. (Local devices reply as needed to the broadcast device, without BBMD involvement.) The local BBMD forwards the broadcast message to the other subnets, using one of two methods (as defined for each remote subnet):

One-Hop The BBMD sends the message using a "directed broadcast," whereby the IP router for the destination subnet broadcasts the message on its local subnet. The router must support directed broadcasts, otherwise the Two-Hop method must be used. One-hop is unusual, as IP routers rarely pass directed broadcasts off or onto the local subnet.

Two-Hop The BBMD sends the message to its peer BBMD on the remote subnet, whereby that BMMD broadcasts the message on its local subnet.

Note that *replies* to BACnet broadcast messages may or may not require BBMD involvement. Occasionally, these are directed messages back to the particular BACnet/IP device that generated the message. However, replies to a "Who-Is" broadcast are often broadcast "I-Am" messages, as is the case with Niagara.

Foreign Device Table

In a case of an IP subnet where only a few (or perhaps just one) BACnet/IP device exists, a local BBMD may be considered excessive for BACnet broadcast message support. An alternative for that subnet is for each BACnet/IP device to register as "foreign device" with a particular BBMD on a remote subnet. Once registered, the device is added to that BBMD's FDT (Foreign Device Table). It then becomes that BBMD's responsibility to deliver global BACnet broadcast messages to that remote device.

[1] The BDT in each BBMD is required to be identical in the original specification. However, Addendum 135-2008o (not currently implemented in Niagara) relaxes this requirement, in order to allow BACnet/IP communication in the Network Address Translation (NAT) environment.

Because this scheme is sometimes used for BACnet/IP devices that are not "permanent," it was designed with a mandatory "registration lifetime" feature. When any BACnet/IP device registers as a foreign device with a BBMD, it must specify its "Time-to-Live" value, in seconds. It is then expected to re-register within this period, else the BBMD will remove (purge) it from its FDT. This prevents unneccesary broadcast delivery attempts to "part-time participants."

The FDT in any BBMD reflects the current list of its registered foreign devices, along with each device's Time-to-Live value and calculated purge time. If the NiagaraAX station is configured as a BBMD, you can see all entries in its FDT. See "Foreign Device Table notes" on page C-5.

Note: *The "foreign device" term implies no stigma—it is purely BBMD-centric. The most expedient configuration for a Niagara station may well be as a foreign device, providing its local IP subnet has no BBMD and other BACnet/IP devices (if any) are currently each registered as foreign device with a remote BBMD.*

Unlike a BBMD's broadcast distribution table, which is identical in each BBMD, the foreign device table in each BBMD is unique to that BBMD.

Niagara BBMD and Foreign Device configuration tips

The following BBMD and Foreign Device configuration tips are grouped into two categories:

- Existing BACnet/IP Network
- New BACnet IP Network

Existing BACnet/IP Network

If you have an existing BACnet/IP network that spans IP subnets, chances are that it has at least one existing BBMD. Unless the host for the Niagara station is located on that same IP subnet as an existing BBMD[1], you must choose one of the following:

- Configure As BBMD
- Configure As Foreign Device

Configure As BBMD

Typically this is best if the local subnet is receiving other new BACnet/IP devices, and/or existing BACnet/IP devices on this subnet are not currently supported by a BBMD. You should determine if the IP router for the local subnet supports "directed broadcasting."

Configure Niagara station as BBMD

Step 1 Expand the Bacnet network's **`BacnetComm`** container to reveal **`Network`**, **`IpPort`**, **`Link`**, opening the property sheet for the **`Link`** component.

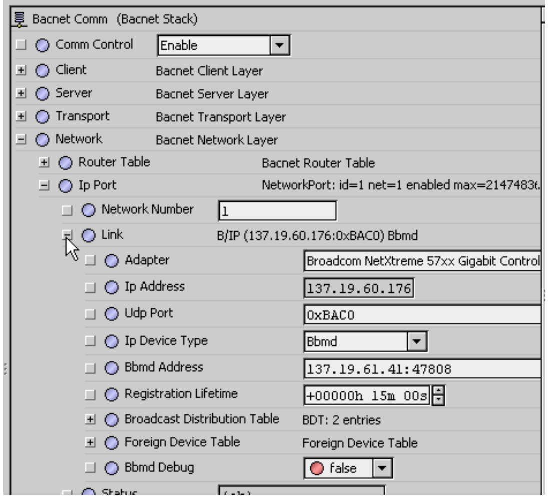
Step 2 In the "Ip Device Type" property, select "Bbmd" from the drop-down list.

Step 3 To have Niagara read from another BBMD, in the "Bbmd Address" property, enter the address of a known BBMD on a remote subnet, using its full BACnet/IP MAC address, that is *`IP_Address`*`:`*`UDP_port`*

For example, for a BBMD at `137.19.61.41` using the standard UDP port 0xBAC0 (decimal 47808), either entry format is accepted: `137:19.61.41:47808` (or) `137.19.61.41:0xBAC0`

[1] If installing the Niagara host on an IP subnet with an existing BBMD for that BACnet/IP network, you are "done". In the Link property sheet of the Ip Port (BacnetNetwork, BacnetComm, IpPort, Link), leave the "Ip Device Type" property set to "Standard".

Figure C-1 *BBMD operation configured in BacnetComm, Network, IpPort, Link container*



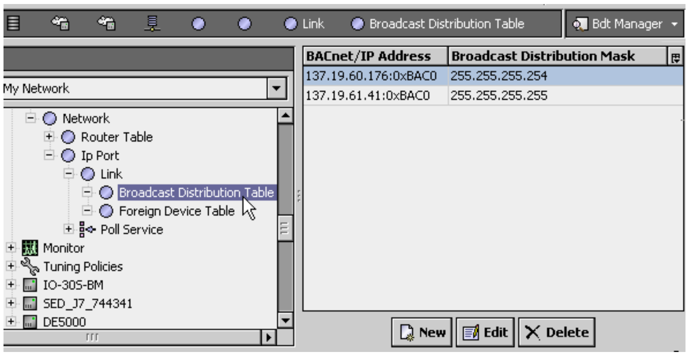
Bacnet Comm (Bacnet Stack)
- Comm Control: Enable
- Client: Bacnet Client Layer
- Server: Bacnet Server Layer
- Transport: Bacnet Transport Layer
- Network: Bacnet Network Layer
  - Router Table: Bacnet Router Table
  - Ip Port: NetworkPort: id=1 net=1 enabled max=2147483
    - Network Number: 1
    - Link: B/IP (137.19.60.176:0xBAC0) Bbmd
      - Adapter: Broadcom NetXtreme 57xx Gigabit Control
      - Ip Address: 137.19.60.176
      - Udp Port: 0xBAC0
      - Ip Device Type: Bbmd
      - Bbmd Address: 137.19.61.41:47808
      - Registration Lifetime: +00000h 15m 00s
      - Broadcast Distribution Table: BDT: 2 entries
      - Foreign Device Table: Foreign Device Table
      - Bbmd Debug: false
    - Status

Step 4 Click **`Save`** to enable this configuration. This should update all other BBMDs on the BACnet/IP network with the same BDT information.

Step 5 Under the **`Link`** container, expand the "**`Broadcast Distribution Table`**" slot to see child entries. There should be one "localDevice" entry and at least one "BdtEntry" for the remote BBMD entered.

Note: *You can also double-click the* **`Broadcast Distribution Table`** *component for its default* **`BDT Manager`** *view. As shown in Figure C-2, the entry for the local device is indicated in a pale blue color.*

Figure C-2 *Bdt Manager view on BroadcastDistributionTable under Link component of IpPort*



BACnet/IP Address	Broadcast Distribution Mask
137.19.60.176:0xBAC0	255.255.255.254
137.19.61.41:0xBAC0	255.255.255.255

This view allows you to manually add new BBMD entries if necessary, or edit or delete existing BBMD entries.

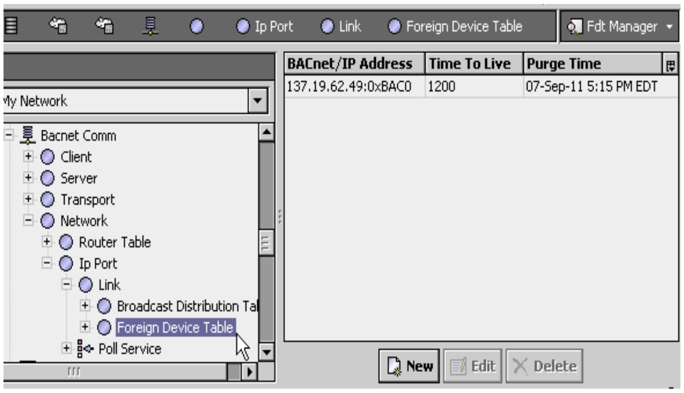
Note: *If the router for the local subnet supports "directed broadcasting," edit the "localDevice" entry's subnet mask (in decimal) in the "Distribution Mask" field. Otherwise, if the router does not support directed broadcasting, or if you are not sure, leave* `255:255:255:255` *as the Distribution Mask.*

Step 6 From the **`Bacnet Device Manager`**, perform a device discover. If the BBMD configuration was successful, you should see devices from other subnets respond.

Foreign Device Table notes If the station is configured as a BBMD, in addition to the **`Broadcast Distribution Table`** component, the IpPort's **`Link`** container also has a **`Foreign Device Table`** component. It accumulates entries for remote BACnet/IP devices that have registered with the station as a foreign device, and includes their registration "time to live" and upcoming purge time.

Double-click the **`Foreign Device Table`** component for its **`Fdt Manager`** view (Figure C-3).

Figure C-3 *Fdt Manager view on Foreign Device Table component under Link component of IpPort*



BACnet/IP Address	Time To Live	Purge Time
137.19.62.49:0xBAC0	1200	07-Sep-11 5:15 PM EDT

As shown in Figure C-3, this view allows you to *manually* add new foreign device entries, or edit or delete existing entries. Manually adding entries is *not typical*, but allows support for devices that cannot register.

If manually adding a device, enter its full BACnet/IP MAC address, that is `IP_Address:UDP_port`, for example: `137:19.64.29:47808` (or) `137.19.64.29:0xBAC0`

Note: *The station automatically purges accumulated FDT entries when the requested lifetime expires. However, any manually-added entries are not purged.*

Configure As Foreign Device

Typically this is best when a remote subnet BBMD exists and the local subnet has only the Niagara station as a BACnet/IP device, or if other BACnet/IP devices on the local subnet are currently registered as foreign devices.

Configure Niagara station as foreign device

Step 1 Expand the Bacnet network's **`BacnetComm`** container to reveal **`Network`**, **`IpPort`**, **`Link`**, opening the property sheet for the **`Link`** component.

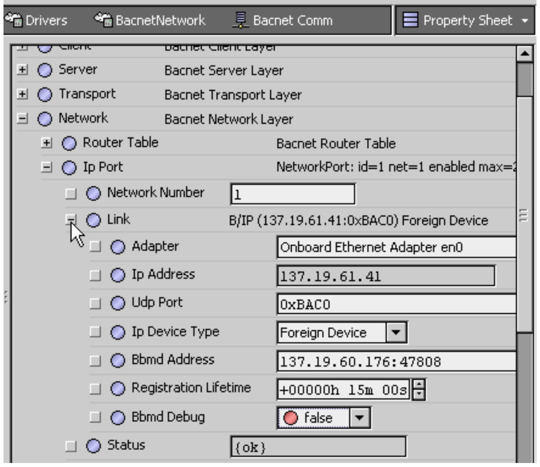
Step 2 In the "Ip Device Type" property, select "Foreign Device" from the drop-down list.

Step 3 In the "Bbmd Address" property, enter the address of a known BBMD on a remote subnet. This is the BBMD that the station will attempt to register with as a foreign device.

Enter using its full BACnet/IP MAC address, which means `IP_Address:UDP_port`

For example, for a BBMD at `137.19.60.176` using the standard UDP port 0xBAC0 (decimal 47808), either entry format is accepted: `137:19.60.176:47808` (or) `137.19.60.176:0xBAC0`

Figure C-4 *Foreign device operation configured in BacnetComm, Network, IpPort, Link container*



Client	Bacnet Client Layer
Server	Bacnet Server Layer
Transport	Bacnet Transport Layer
Network	Bacnet Network Layer
Router Table	Bacnet Router Table
Ip Port	NetworkPort: id=1 net=1 enabled max=:
Network Number	1
Link	B/IP (137.19.61.41:0xBAC0) Foreign Device
Adapter	Onboard Ethernet Adapter en0
Ip Address	137.19.61.41
Udp Port	0xBAC0
Ip Device Type	Foreign Device
Bbmd Address	137.19.60.176:47808
Registration Lifetime	+00000h 15m 00s
Bbmd Debug	false
Status	{ok}

Step 4 In the "Registration Lifetime" property, either accept the default 15 minutes, or enter another value. The station will automatically re-register within this period.

Step 5 Click **`Save`** to enable this configuration. This station attempts to register with the remote BBMD.

Step 6 From the **`Bacnet Device Manager`**, perform a device discover. If the foreign device configuration was successful, you should see devices from other subnets respond.

New BACnet IP Network

If configuring the Niagara station while installing BACnet/IP devices for a new BACnet/IP network, there are probably no existing BBMDs. In this case, you would typically follow "Configure As BBMD" on page C-4. However, you would skip the step to specify a remote BBMD, unless one existed.

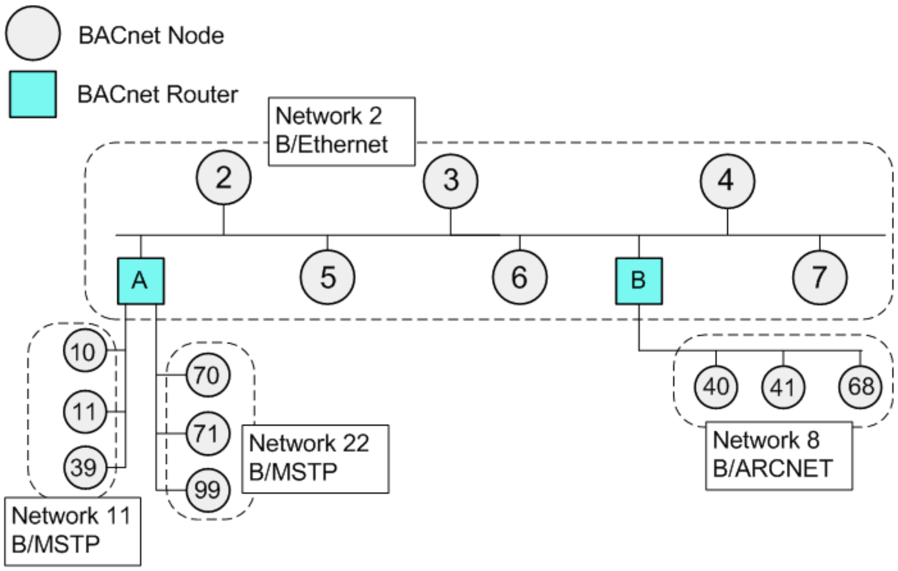
This would allow BACnet/IP devices on remote subnets to register with the station as foreign devices. You would not see other BBMDs in the **`Broadcast Distribution Table`**, but you could see if other devices have registered as foreign devices, in the **`Foreign Device Table`**.

See "Foreign Device Table notes" on page C-5 for related details.

Example Internetwork Diagrams

Figure C-5 shows an internetwork with 2 BACnet routers and 4 BACnet networks.

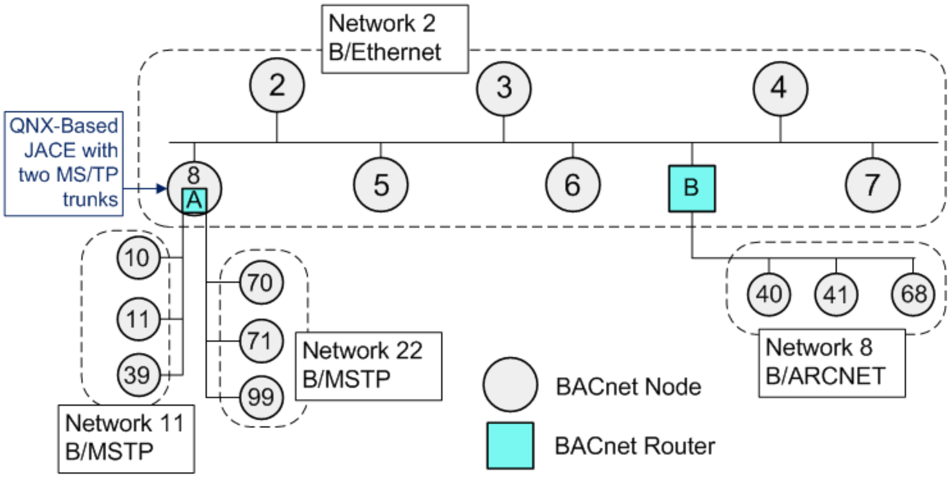
Figure C-5 *Four physical networks, 2 routers*



In the example above, routers A and B join together three different media and link-layer types: BACnet over Ethernet (network 2), BACnet MS/TP (networks 11 and 22), and BACnet ARCnet (network 8).

Note that each BACnet node has a unique Device ID number (shown inside node). Note also that router A could actually be a JACE that is licensed for MS/TP, as shown in Figure C-6.

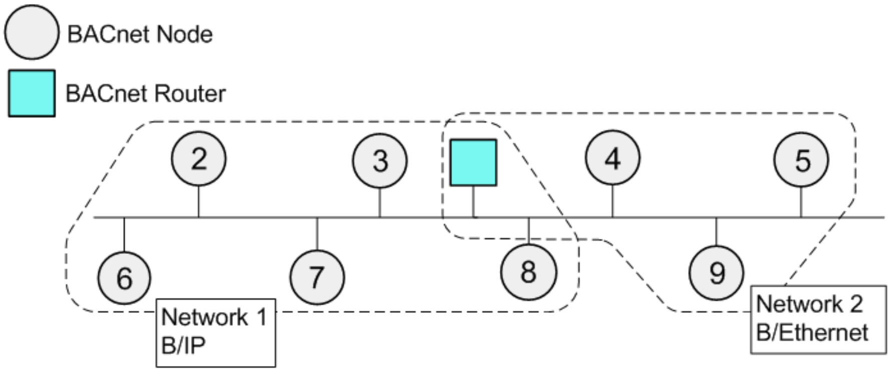
Figure C-6 *BACnet Ethernet to MS/TP router functions provided by station in JACE.*



In this case, the JACE's station provides BACnet routing between the two MS/TP networks and Ethernet network.

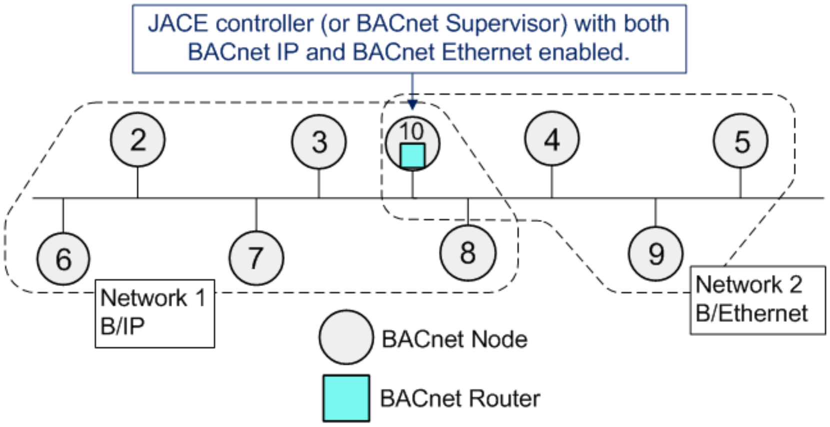
On an Ethernet LAN, there may be two separate BACnet networks sharing the same media—BACnet over Ethernet and BACnet/IP, as shown in Figure C-7.

Figure C-7 *Same physical Ethernet LAN, router joins two BACnet networks.*



Again, note that each BACnet node has a unique Device ID number (shown inside node). Note also that this router function could actually be provided by a Niagara station, as shown inFigure C-8.

Figure C-8 *BACnet/IP to BACnet/Ethernet router function provided by NiagaraAX station*


JACE controller (or BACnet Supervisor) with both BACnet IP and BACnet Ethernet enabled.
2
3
10
4
5
6
7
8
9
Network 1
B/IP
Network 2
B/Ethernet
BACnet Node
BACnet Router

In this case, you must be especially careful that no other router exists between the BACnet Ethernet and BACnet/IP networks—if the Niagara station detects this illegal (loop) configuration, the routing functions performed by the station are stopped.

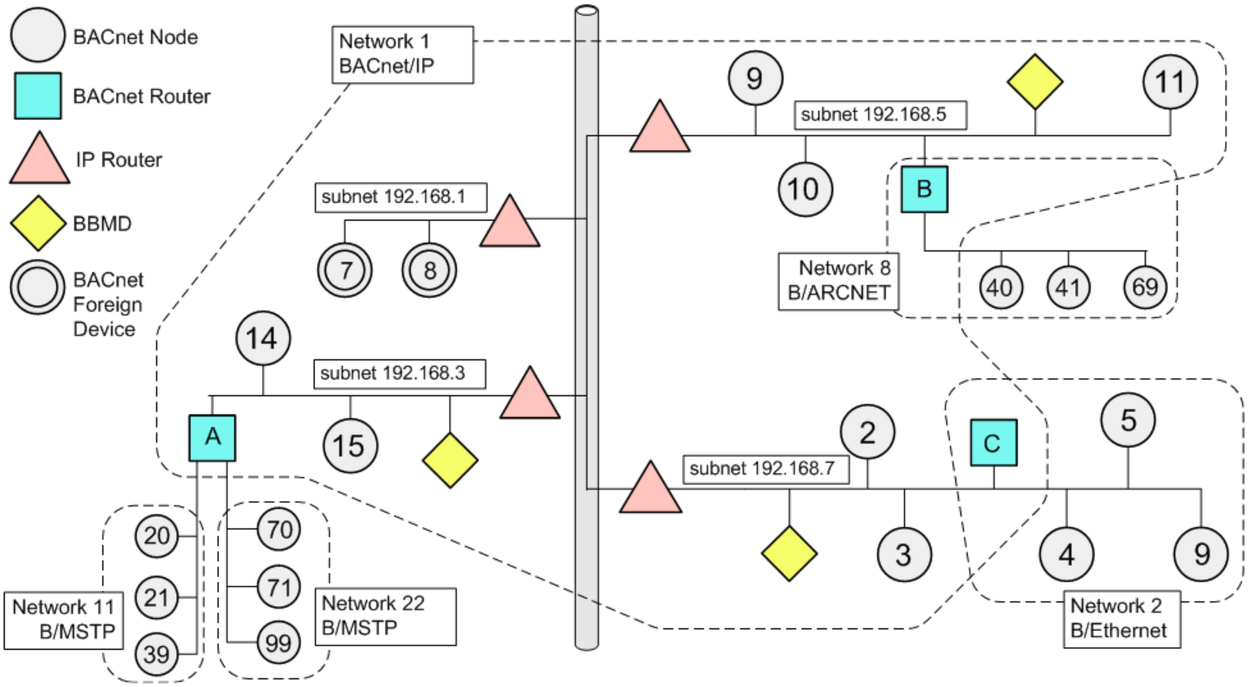
This condition is indicated on the property sheet of the `BacnetComm`, `Network` component, where its “Routing Enabled” property has been automatically set back to false (from true). In such a case, corresponding entries are generated in the stations's LogHistory, noting detected misconfiguration and disabled BACnet router functionality.

BACnet/IP

BACnet/IP networks can be more involved, with devices designated as BBMDs and "foreign devices", plus standard TCP/IP (IP) routers. Note that BBMDs are used to avoid having separate B/IP networks for each subnet—this would require a BACnet/IP router to exist on each subnet (yet even more complexity).

Figure C-9 shows an example internetwork that includes an Ethernet/IP backbone.

Figure C-9 *Internetwork that includes single B/IP network spanning multiple IP subnets, plus other BACnet networks*



The B/IP network shown in Figure C-9 spans multiple IP subnets. Use of BBMDs provides BACnet broadcast message delivery through IP routers

Note that on subnet 192.168.1, there is no BBMD. Both BACnet/IP nodes on that subnet (Device IDs 7, 8) are registered as a "foreign device" with a remote BBMD.

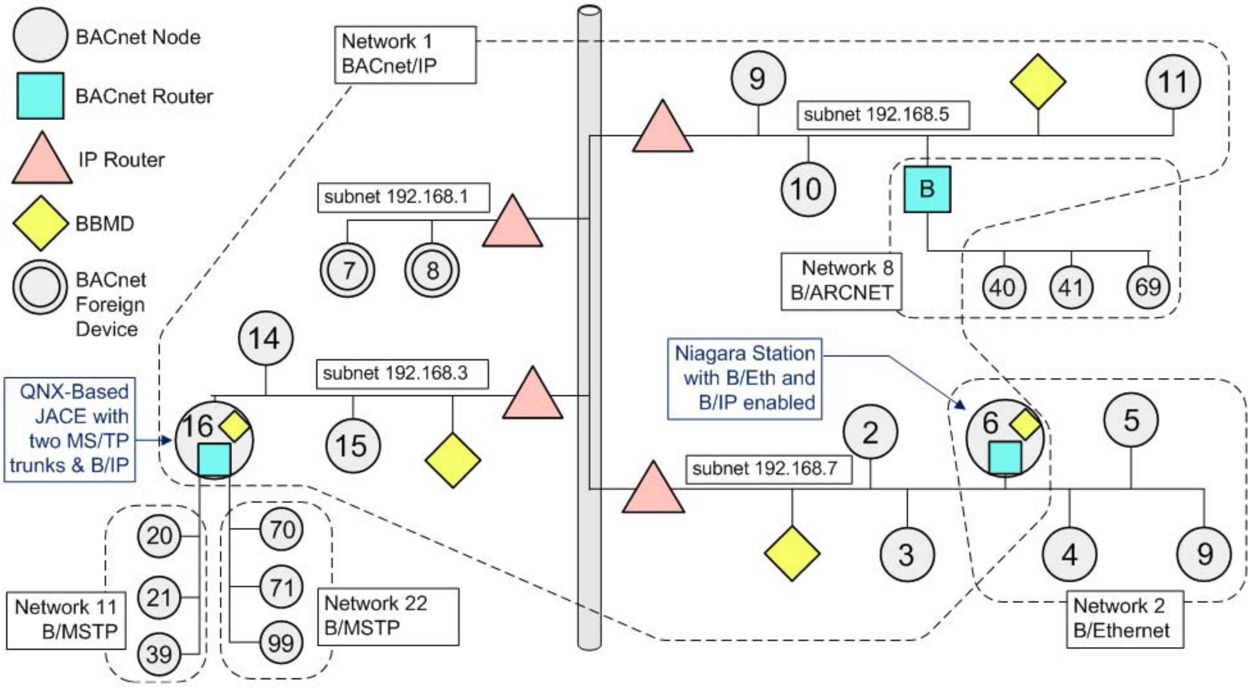
Note also that on the physical Ethernet segment used by subnet 192.168.7, there are both BACnet/IP devices (Device IDs 2 and 3) and BACnet over Ethernet devices (Device IDs 4, 5, and 9). Router "C" provides communications between these two BACnet networks.

With the exception of router "B" (B/IP to B/ARCNET), all BACnet routing and BBMD functions can be performed by Niagara stations.

- On subnet 192.168.3, router A functions (B/IP to MS/TP) and BBMD functions can be performed by a QNX-based JACE with two MSTP trunks and BACnet/IP enabled.
- On subnet 192.168.5, BBMD functions can be performed by any Niagara station with BACnet/IP enabled.
- On subnet 192.168.7, router C functions (B/IP to B/Eth) and BBMD functions can be performed by any Niagara station with both B/Ethernet and BACnet/IP enabled.

Figure C-10 shows an a similar internetwork with these functions provided by Niagara stations, as described above.

Figure C-10 *Similar internetwork with Niagara stations providing BBMD and routing functions*



In the internetwork above, the station operating as BACnet device (node) 6 could be hosted either by either a JACE controller, or a BACnet (OWS or AWS) Supervisor.