Pup Driver Guide

User Guide

DISTECH CONTROLS™

Innovative Solutions for Greener Buildings

EC-Net^{AX} PUP Driver Guide

Copyright © 2010 Distech Controls, Inc.
All rights reserved.
4005-B Matte Boulevard
Brossard
Quebec
J4Y 2P4
Canada

Confidentiality Notice

The information contained in this document is confidential information of Distech Controls, Inc. 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 Distech Controls 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, Distech Controls 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, Brossard, Quebec.

Trademark Notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft and Windows are registered trademarks, and Windows NT, Windows 2000, Windows XP Professional, and Internet Explorer are trademarks of Microsoft Corporation. Java and other Java-based names are trademarks of Sun Microsystems Inc. and refer to Sun's family of Java-branded technologies. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Niagara Framework and Niagara^{AX} Framework registered trademarks of Tridium, Inc. Distech Controls, the Distech Controls logo, and Innovative Solutions for Greener Buildings are trademarks of Distech Controls, Inc. All other product names and services mentioned in this publication that is known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

Copyright and Patent Notice

This document may be copied by parties who are authorized to distribute Distech Controls products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Distech Controls Inc.
©, Distech Controls Inc. 2010. All rights reserved. The product(s) described herein may be covered by one or more U.S or foreign patents.

TABLE OF CONTENTS

Preface iii

Document Change Log iii

Compatibility and Installation. 1–1

Compatibility 1–1

EC-Net^{AX} platform compatibility 1–1

PUP Specifications 1–1

Mapping of PUP Attribute Types to Proxy Extensions 1–3

License requirements 1–5

Installation 1–5

PUP Driver Quick Start 2–1

Add and configure the PupNetwork 2–1

Add the PupNetwork 2–1

To add a PupNetwork in the station 2–1

Configure PUP network communication parameters 2–1

To set the PUP network communications parameters 2–1

Add PUP devices 2–2

Using online Discover to add PupDevices 2–2

To discover PUP Devices 2–2

To add discovered PUP Devices 2–2

Create PUP proxy points 2–3

To add PUP proxy points 2–3

EC-Net^{AX} PUP Concepts 3–5

About PUP Architecture 3–5

Palette for PUP driver 3–5

About the PUP Network 3–6

Common PupNetwork slots 3–6

PupNetwork status notes 3–6

PupNetwork monitor notes 3–6

PupNetwork tuning policy notes 3–7

PupNetwork poll scheduler notes 3–7

PupNetwork message handling properties 3–7

PupNetwork serial port config properties 3–7

PUP-specific network slots 3–7

Token Pass Config 3–8

PupNetwork action 3–10

Pup Device Manager 3–10

About PUP Device Discovery Config 3–10

Pup Device Manager discover notes 3–11

PupDevice 3–12

PupDevice properties 3–12

Pup Region Manager 3–13

Pup Point Manager 3–15

About PUP Point Discovery Config 3–16

Pup Point Manager discover notes 3–17

PUP proxy points 3–18

PupProxyExt properties 3–18

PUP BooleanWritable notes 3–19

NumericPoints with “CV” attribute 3–19

StringPoint actions for time or date attributes 3–20

PupProxyExt actions 3–20

Modifying DeviceTypes.xml 3–21

XML syntax 3–21

Example: HX1 and GPC1 3–22

PUP (aapup) Plugin Guides 4–23

Plugin Guides Summary 4–23

aapup-PupDeviceManager 4–23

aapup-PupPointManager 4–24

aapup-PupRegionManager 4–25

PUP (aapup) Component Guides 5–27

Component Guides Summary 5–27

aapup-PupBooleanProxyExt 5–27

aapup-PupDevice 5–27

aapup-PupDeviceFolder 5–28

aapup-PupEnumProxyExt 5–28

aapup-PupNetwork 5–28

aapup-PupNumericProxyExt 5–28

aapup-PupPeerListFolder 5–28

aapup-PupPointDeviceExt 5–28

aapup-PupPointFolder 5–28

aapup-PupStringProxyExt 5–29

Preface

This documents usage of the American Auto-Matrix® Public Unitary Protocol (PUP)™ driver for the NiagaraAX framework.

The following main sections are in this document:

- "Compatibility and Installation" on page 1-1
  Explains PUP devices/protocols supported, as well as the EC-Net^{AX} platform, software, and licensing requirements.
- "PUP Driver Quick Start" on page 2-1
  Provides several quick procedures for online station configuration to add a PupNetwork, PupDevices, and PUP proxy points.
- "EC-Net^{AX} PUP Concepts" on page 3-5
  Provides concepts behind the PUP driver, including all major components and views, including various screenshots. Includes sections on "special" proxy points and the "DeviceTypes.xml" file.
- "PUP (aapup) Plugin Guides" on page 4-23
  Provides brief summaries of the different PUP manager views, each with links back to the more detailed concepts section. Entries are used in EC-Net^{AX} context-sensitive help "On View".
- "PUP (aapup) Component Guides" on page 5-27
  Provides brief summaries of the different PUP components, most with links back to the more detailed concepts section. Entries are used in EC-Net^{AX} context-sensitive help "Guide On Target".

Document Change Log

Updates (changes/additions) to this *EC-Net^{AX} PUP Driver Guide* document are listed below.

- Updated: November 2011
  Changed document to reference the *EC-Net^{AX} Drivers Guide*, a new document created from sections formerly in the *EC-Net^{AX} User Guide*.
- Updated: July 27, 2007
  Completely reworked as a "single-source" document, replacing the previous PDF-only *EC-Net^{AX} PUP Driver User Guide*. Includes standard chapters/sections common to most EC-Net^{AX} driver documents.
- Revised: May 31, 2006
  Added new section "Modifying DeviceTypes.xml" and this document change log. Also, some additional Notes were added in a few of the property descriptions for the PupNetwork (for example "Token Recovery", "Token Recovery Timeout").
- Initial document: September 20, 2005
  Initial publication as PDF-only document.

CHAPTER 1

Compatibility and Installation

Currently, this section has the following main subsections:

- Compatibility
- License requirements
- Installation

Compatibility

The EC-Net^{AX} PUP driver has the following compatibility criteria:

- EC-NetAX platform compatibility
- PUP Specifications
- Mapping of PUP Attribute Types to Proxy Extensions

EC-Net^{AX} platform compatibility

The PUP driver will function on all EC-Net^{AX} platforms that support serial communications.

PUP Specifications

The PUP driver is compliant with PUP version 8.35 protocol specifications, as published in the document *PUP version 8.35 Public Unitary Protocol Guidelines*[1]dated July 2005.

Table 1-1 details support for PUP protocol 8.35 commands/responses.

Table 1-1 *PUP protocol 8.35 command/response support*

Command Description	Code	Notes
Synchronize Time and Date	0x00	Broadcast and Directed time sync available as actions on the PupNetwork and PupDevice respectively
Read Attribute	0x01	Used to poll for proxy ext data from device
Write Attribute	0x02	Used to write proxy ext data to device
Pass Token	0x03	Token passing configured from PupNetwork
Say “Hello”	0x04	Used for device ping and network learn
Read Region Data	0x05	Supports SPL upload/download manager
Write Region Data	0x06	Supports SPL upload/download manager
Create Named Region	0x07	Supports SPL upload/download manager
Lookup Named Region	0x08	Supports SPL upload/download manager
Report Exception Message	0x09	Used for native alarm support for AX
Exchange Data with Virtual Terminal	0x0A	Not Supported
Retransmit Virtual Printout	0x0B	Not Supported
Open/Close Virtual Terminal	0x0C	Not Supported

1. “Pup Version 8.35 Public Unitary Protocol”, July 2005, American Auto-Matrix, One Technology Lane, Export, Pennsylvania 15632

Command Description	Code	Notes
Acknowledge Transaction	0x0D	Used for native alarm support for AX
Change Operation Mode	0x0E	Not Supported
Declare Exception Message	0x0F	Not Supported
Read Address	0x10	Not Supported
Write Address	0x11	Not Supported
Free Region	0x1D	Supports SPL upload/download manager
Write Text Attribute	0x1E	Supports PupStringProxyExt in devices that support this command
Write Zone Attribute	0x1F	Not Supported
Flash Upgrade	0x20	Not Supported
Reserved	0x21	Not Supported
Reserved	0x22	Not Supported
Read Channels	0x23	Supports AX point learns in devices that support this command
Large Flash Upgrade	0x24	Not Supported
Error Response	0x80	Supported
General Acknowledge Response	0x81	Supported
Numeric Data Response	0x82	Supported
Text Data Response	0x83	Supported (applies only to PupStringProxyExt)
Region Data Response	0x84	Supports SPL upload/download manager
Exception Message Response	0x85	Supported for native alarm support for AX
Virtual Printout Response	0x86	Not Supported
Region Name Response	0x87	Supported for native alarm support for AX
Read Channels Response	0x88	Supported for point learns from devices that support this command.
Large Flash Ack	0x89	Not Supported

Mapping of PUP Attribute Types to Proxy Extensions

Table 1-2 lists the data types that can be modeled as points by this driver, and the type of proxy extensions that they can be mapped to.

- “R” under an extension type means the proxy extension can be an extension on a “read-only” control point (for example a “NumericPoint”).
- “RW” under an extension type means the proxy extension can be an extension on a either a “read-only” or a “read-write” control point (for example a “NumericWritable”).
- “—” under a proxy extension type means the attribute type cannot be mapped to that type.

Table 1-2 *PUP attribute type to PupProxyExt type*

PUP Attribute Type	Format	PupNumericProxyExt	PupBooleanProxyExt	PupEnumProxyExt	PupStringProxyExt
0xFF	Signed 10 digit	RW	RW	RW	—
0xFE	Unsigned 10 digit	RW	RW	RW	—
0xFD	Signed 9.1 digit	RW	RW	RW	—
0xFC	Unsigned 9.1 digit	RW	RW	RW	—
0xFB	Signed 8.2 digit	RW	RW	RW	—
0xFA	Unsigned 8.2 digit	RW	RW	RW	—
0xF9	Signed 7.3 digit	RW	RW	RW	—
0xF8	Unsigned 7.3 digit	RW	RW	RW	—
0xF7	Signed 6.4 digit	RW	RW	RW	—
0xF6	Unsigned 6.4 digit	RW	RW	RW	—
0xF5	Signed 5.5 digit	RW	RW	RW	—
0xF4	Unsigned 5.5 digit	RW	RW	RW	—
0xF3	Signed 4.6 digit	RW	RW	RW	—
0xF2	Unsigned 4.6 digit	RW	RW	RW	—
0xF1	Signed 3.7 digit	RW	RW	RW	—
0xF0	Unsigned 3.7 digit	RW	RW	RW	—
0xEF	Signed 2.8 digit	RW	RW	RW	—
0xEE	Unsigned 2.8 digit	RW	RW	RW	—
0xED	Signed 1.9 digit	RW	RW	RW	—
0xEC	Unsigned 1.9 digit	RW	RW	RW	—
0xEB	Signed .10 digit	RW	RW	RW	—
0xEA	Unsigned .10 digit	RW	RW	RW	—
0xE9	Channel map	RW	RW	RW	—
0xE8	Bitmap of text	RW	RW	RW	—
0xE7	BCD Hours/Minutes/Seconds	—	—	—	R
0xE6	BCD Hours/Minutes	—	—	—	R
0xE5	Packed BCD	RW	RW	RW	—
0xE4	BCD Date (Y/M/D)	—	—	—	R
0xE3	Binary Date	—	—	—	R
0xE2	Reserved	—	—	—	—
0xE1	Reserved	—	—	—	—
0xE0	IEEE 754 32-bit floating point	RW	RW	RW	—
0xDF	ANSI X34	—	—	—	RW
0x7F	Bogus Sample	—	—	—	R
0x08	Packed Date	—	—	—	R
0x07	Boolean	—	RW	—	—
0x06	ASCII-Z	—	—	—	R
0x05	DOS Day-of-week	—	—	—	R
0x04	DOS Time	—	—	—	R
0x03	Segment:Offset	—	—	—	R
0x02	Hex Double	RW	RW	RW	—

PUP Attribute Type	Format	PupNumericProxyExt	PupBooleanProxyExt	PupEnumProxyExt	PupStringProxyExt
0x01	Hex Word	RW	RW	RW	—
0x00	Hex Byte	RW	RW	RW	—

License requirements

To use the EC-Net^{AX} PUP driver, you must have a target EC-Net^{AX} host (EC-BOS^{AX}) that is licensed with the "`aapup`" feature, as well as the "`serial`" feature. In addition, the "`aapup`" feature may have other device limits or proxy point limits.

Installation

From your PC, use the EC-Net^{AX} Pro 3.*n.nn* installed with the "installation tool" option (checkbox "This instance of EC-Net^{AX} Pro will be used as an installation tool"). This option installs the needed distribution files (.*dist* files) for commissioning various models of remote EC-BOS^{AX} platforms. The dist files are located under your EC-Net^{AX} install directory under a "sw" subdirectory. For more details, see "About your software database" in the *Platform Guide*.

Apart from installing the 3.*n.nn* version of the EC-Net^{AX} distribution in the EC-BOS^{AX}, make sure to also install the *aapup* module too, plus any modules shown as dependencies. For more details, see "About the Commissioning Wizard" in the *EC-BOS^{AX} Install and Startup Guide*.

Following this, the remote EC-BOS^{AX} is now ready for PUP software integration, as described in the rest of this document. See the next section "PUP Driver Quick Start" for a series of task-based procedures.

CHAPTER 2

PUP Driver Quick Start

This section provides a collection of procedures to use the EC-Net^{AX} PUP driver to build an PupNetwork with proxy points. Like other EC-Net^{AX} drivers, you can do most configuration from special "manager" views and property sheets using EC-Net^{AX}.

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

These are the main quick start subsections:

- Add and configure the PupNetwork
- Add PUP devices
- Create PUP proxy points

Add and configure the PupNetwork

- Add the PupNetwork
- Configure PUP network communication parameters

Add the PupNetwork

The PupNetwork is the top-level PUP component in the station.

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

Step 3 Scroll to select "`Pup Network`," number to add: `1` and click **`OK`**.

This brings up a dialog to name the network.

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

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

After you Configure PUP network communication parameters, status should change to "`{ok}`".

Configure PUP network communication parameters

In the PupNetwork property sheet you must define several parameters for communications.

To set the PUP network communications parameters

To set the communications parameters for a PupNetwork:

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

The **`Property Sheet`** appears.

Step 2 Expand the **`Serial Port Config`** slot.

Set the properties for the EC-BOS^{AX} serial port used, where defaults are:

- Port Name: `none` — Enter the EC-BOS^{AX} port being used, like COM2 or COM3.
- Baud Rate: `Baud9600` — Or choose different from selection list.
- Data Bits: `Data Bits8` — Or choose different from selection list.
- Stop Bits: `Stop Bit1` — Or choose different from selection list.
- Parity: `none` — Or choose different from selection list.

- Flow Control Mode: `none` — Or choose different using checkbox.

Note: *You must determine the setup of the PUP serial network to correctly set the baud rate, data bits, stop bits, parity, and flow control settings.*

Step 3 Set the **`Unit Number`** property value to assign an address to the EC-Net^{AX} network node. The PupNetwork behaves as a master device on the PUP network, and as such must have an address in order to send and receive messages from the network.

Step 4 Expand the **`Token Pass Config`** slot and edit properties as needed. For more details, see "Token Pass Config" on page 3-8.

Step 5 Click the **`Save`** button.

For further details on the PupNetwork, see "About the PUP Network" on page 3-6.

Add PUP devices

After adding a PupNetwork, you can use the network's default "PupDeviceManager" view to add PUP devices.

Using online Discover to add PupDevices

If the EC-BOS^{AX} is connected to the PUP network, this is the easiest way to accurately populate the station with the necessary PupDevice objects. Use the following procedures:

- To discover PUP Devices
- To add discovered PUP Devices

To discover PUP Devices

Perform this task to discover PUP devices.

Step 1 Double-click the **`PupNetwork`** to bring up the **`Pup Device Manager`**.

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

A popup dialog appears. By default, discovery occurs for all possible PUP devices, by address range (unit numbers), from 0 to 65534. Typically, you make changes before initiating the discovery.

Note: *To learn what devices are on the wire, it is recommended that you narrow the Start Address and Stop Address range, because the learn process has to loop through and attempt to talk to each and every address in the range. Attempting communications to 65535 devices would take a long time!*

The "Search By" item defaults to "search using unit number," but may be changed to a "back door" search using the American Auto-Matrix method of serial number access. For more details, see "About PUP Device Discovery Config" on page 3-10

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 PUP 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 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 procedure: "To add discovered PUP Devices".

To add discovered PUP Devices

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

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 PupDevice object before it is added in the EC-Net^{AX} station, to tweak display name, enabled state, and/or address of selected devices.

- For PUP-specific details, see "Pup Device Manager discover notes" on page 3-11.
- For general details, see the Device Manager section "Add" in the *User Guide*.

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

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

For further details, see "PupDevice" on page 3-12.

Create PUP proxy points

As with device objects in other drivers, each PupDevice 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 "**`Pup Point Manager`**"). You use it to add PUP proxy points under any PUP device.

For general information, see the "About the Point Manager" section in the *User Guide*.

Note: *Like the point managers in many other drivers, the* **`Pup Point Manager`** *offers a "Learn mode" with a* **`Discover`** *button and pane.*

To add PUP proxy points

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

Step 1 In the Nav tree expand the PupDevice and double-click the Points component, or in the Pup Device Manager, double-click the `Points` icon in the row for the device you wish to create proxy points.

This brings up the **`Pup Point Manager`**.

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

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

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

A popup dialog appears. By default, discovery occurs for all possible pre-defined channels (max to min) in the defined channel list file. If needed, you can make changes before initiating the discovery. For more details, see "About PUP Point Discovery Config" on page 3-15.

Click OK to initiate the discovery job, which when finished displays all found attributes in the specified channel range in the top pane.

Step 4 In the discovered pane of the Pup Point Manager, as needed expand channel rows showing a "+" to see additional attributes apart from the first (default) attribute. Each row is a proxy point candidate, which you can select individually or in multiples.

Step 5 Double-click an item to add as a proxy point, or select multiple items and click **`Add`**.

The **`Add`** dialog appears, in which you select a point "Name" and "Type", and typically review the other fields and change if needed.

Note: *The "Attribute Type" determines which proxy "Type" you can select. For a list of valid mappings of Attribute Type to ProxyExt type, see "Mapping of PUP Attribute Types to Proxy Extensions" on page 1-3.*

For additional Pup Point Manager details, see "Pup Point Manager" on page 3-15.

Step 6 Click **`OK`**.

This adds the PUP proxy point(s) to the database, where they are visible in the database pane of the view, showing the current values of the attributes.

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

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

CHAPTER 3

EC-Net^{AX} PUP Concepts

This section provides conceptual details on the EC-Net^{AX} PUP driver and its components, including views. These are the main subsections:

- About PUP Architecture
- About the PUP Network
- Pup Device Manager
- PupDevice
- Pup Region Manager
- Pup Point Manager
- PUP proxy points
- Modifying DeviceTypes.xml

About PUP Architecture

The PUP driver uses the standard EC-Net^{AX} network architecture, similar to other serial-based polling drivers. See "About Network architecture" in the *Drivers Guide* for more details. For example, real-time data is modeled using PUP proxy points, which reside under PupDevices, which in turn reside under a PupNetwork container in the station's DriverContainer (Drivers).

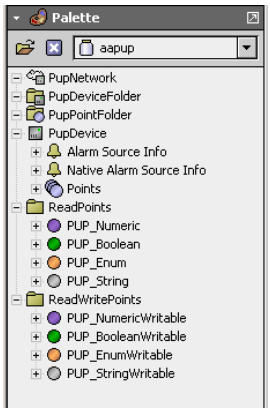
Hierarchically, the component architecture is: network, device, points extension, points. The points extension is the only "device extension" under a PupDevice—meaning there are no schedule or history device extensions. The default "Pup manager" views of the network (and the points extensions under each device) supports online discovery of PUP devices and their data items for proxy points, respectively.

Unique to a PupDevice is its default "Pup Region Manager" view, providing an interface for you to view, upload, and download SPL program regions in the selected PUP device.

Palette for PUP driver

The `aapup` module has a palette, in which you can open to copy components into a station (Figure 3-1).

Figure 3-1 *Components in aapup palette*

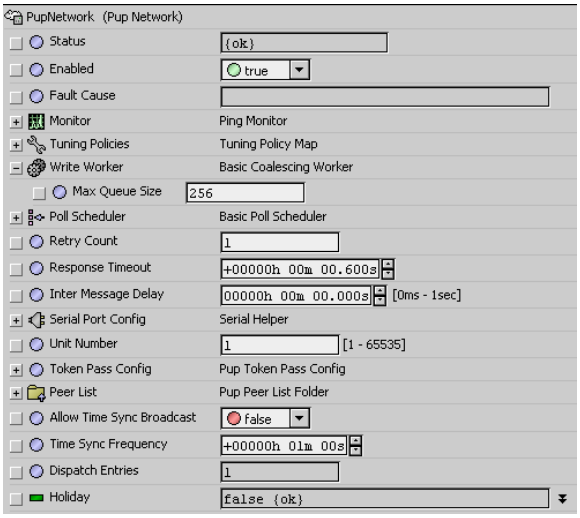


However (as with most EC-Net^{AX} drivers), you *rarely* need to work from the palette. Instead, the various Pup manager views simplify component creation, enforcing proper component hierarchy.

About the PUP Network

The PupNetwork is the top-level component for the PUP driver in a station. Using its property sheet (Figure 3-2), you configure specific settings for accessing other PUP devices on the attached serial network, where the station operates as a "host" PUP device on that network.

Figure 3-2 *Property sheet of PupNetwork*



Once configured, the default "Pup Device Manager" view of the PupNetwork allows online "discovery" to create, edit, and delete child PupDevice components.

The following sections provide more details on PupNetwork properties and slots:

- Common PupNetwork slots
- PUP-specific network slots
- PupNetwork action

Common PupNetwork slots

The PupNetwork component includes the typical collection of slots and properties as most other network components. For general information, See "Common network components" in the *Drivers Guide*. The following sections provide additional details:

- PupNetwork status notes
- PupNetwork monitor notes
- PupNetwork tuning policy notes
- PupNetwork poll scheduler notes
- PupNetwork message handling properties
- PupNetwork serial port config properties

PupNetwork status notes

As with most "fieldbus" drivers, the status of a PupNetwork is either the normal "ok" or less typical "fault" (fault might result from licensing error, or if a non-existent COM port is assigned to Serial Port Config). The Health slot contains historical timestamp properties that record the last network status transitions from ok to any other status. The "Fault Cause" property further explains any fault status.

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

PupNetwork monitor notes

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

PupNetwork tuning policy notes

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

PupNetwork poll scheduler notes

The PupNetwork has the typical Poll Scheduler slot, as described in "About poll components" in the *Drivers Guide*. It enables/disables polling, determines fast/normal/slow poll rates, and maintains statistics about proxy extension polls.

PupNetwork message handling properties

The PupNetwork has several "message handling" network-level properties common among serial drivers, described separately as follows:

- Retry Count — Determines how many retries the communications handler will try to send a message if the initial attempt is unsuccessful. For the PUP protocol, this should normally be set to 1.
- Response Timeout — Specifies the maximum time to wait for a response to a PUP message once sent. If a response is not received before this timeout, the PUP message is resent up to "Retry Count" times, each of which waits for this timeout period.
- Inter Message Delay — The minimum amount of time to wait between receiving a message on the PUP bus, and sending the next request. This gives time for some PUP devices to prepare for receiving messages again. Note that although setting this to a non-zero value has a negative impact on overall throughput, it may be necessary if a "slow-to-turn-around" PUP device is on the network.

PupNetwork serial port config properties

The PupNetwork has a Serial Port Config container with the following properties:

- Status — Either `{ok}` or `{fault}`.
- Port Name — String for the serial port, known to the host platform. For example, `COM1` or `COM4`.
- Baud Rate — Selected from a drop-down list.
- Data Bits — Selectable as 5,6, 7, or 8 bits. For PUP protocol use 8 data bits.
- Stop Bits— Selectable as 1 or 2 bits. For PUP protocol use 1 stop bit.
- Parity — Selectable as None, Odd, Even, Mark, or Space. For PUP protocol use None.
- Flow Control Mode — Do not select any flow control for PUP protocol.

PUP-specific network slots

In addition to Common PupNetwork slots, the PupNetwork contains key PUP configuration properties that enable it to operate on the PUP network, as follows:

- **Unit Number**
  The PupNetwork behaves as a master or peer device on the PUP network, and as such must have an address in order to send and receive directed messages from the network. This is the address.
- **Token Pass Config**
  A container for all properties that govern how the driver handles token passing. See the next section "Token Pass Config".
- **Peer List**
  This is a dynamic list of devices which require the token to be passed to them. Whenever the driver determines it is time to pass the token to a device on the network, it selects the device from this list which has not possessed the token in the longest amount of time (that is, the "oldest").
- **Allow Time Sync Broadcast**
  If set to true, then time sync messages will be broadcast to address 65535 at an interval determined by the next property "Time Sync Frequency." The time broadcast is the local time of the platform the driver is running on, and the "holiday" bit is set according the following property "Holiday."
- **Time Sync Frequency**
  The frequency at which the time sync messages are broadcast, if enabled.
- **Dispatch Entries**
  Diagnostic only.
- **Holiday**
  The time sync message includes a bit for "Holiday." The "Sync Time" action on the PupNetwork

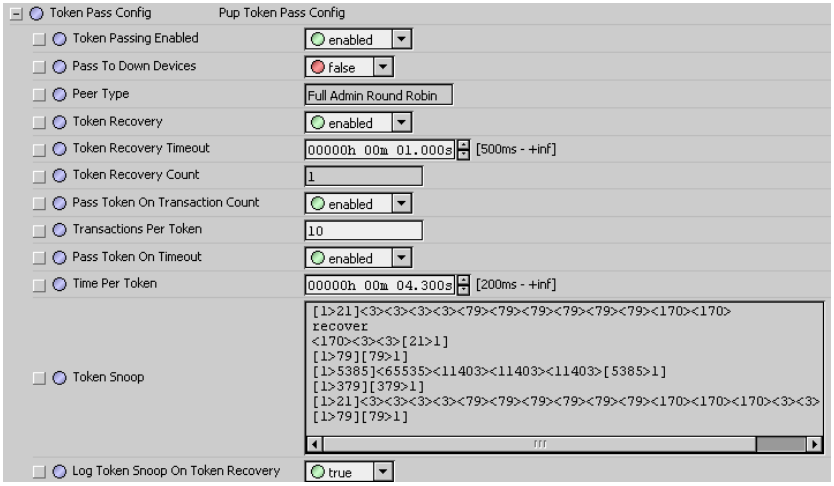
will set or reset this bit in PUP controllers based on the value of this property. This is a linkable property that can be set from a schedule object or some other component.

- **Device Type Files**
  Specifies an ord to the XML-formatted file that is referenced when doing online "Discovers" of PUP devices and proxy points. The file describes device types, channel names, channel search ranges, and so forth. By default, the ord points to the `deviceTypes.xml` file included within the `aapup.jar`.
  If needed, you can unzip and edit/modify this file—and then copy the modified file back onto EC-BOS^{AX}s running the PUP driver (typically under the station directory), where you can point to it with this PupNetwork property. For more details, see "Modifying DeviceTypes.xml" on page 3-21.

Token Pass Config

In the PupNetwork property sheet, click to expand this container to access child properties (Figure 3-3).

Figure 3-3 *Token Pass Config properties of PupNetwork*



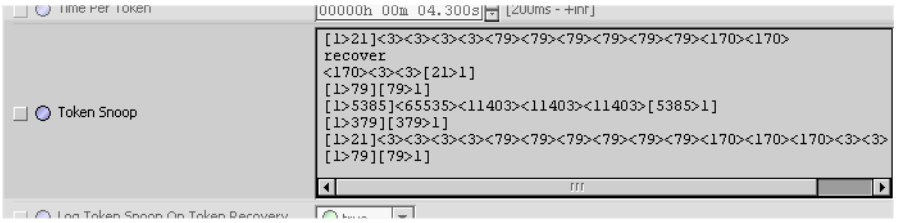
Token Pass Config properties include the following:

- **Token Passing Enabled**
  Select either enabled or disabled from the drop-down menu. If token passing is disabled, then the PUP driver becomes a strict PUP Master, and all devices on the network should be configured as slaves. If token passing is enabled, then tokens are passed based on the values of the remaining Token Pass Config properties, as described below.
- **Pass to Down Devices**
  Normally set to false, this property governs whether or not the driver attempts to pass the token to devices that were previously marked as non-responsive. Since a token pass does not expect an explicit response from the device being passed the token, the token will have to be recovered if the device is still not responsive (see the related properties "Token Recovery", "Token Recovery Timeout" and "Token Recovery Count").
- **Peer Type**
  (Read-only) The PUP driver is always considered a "Full Admin". This cannot be changed.
- **Token Recovery**
  Enables or disables token recovery.
  Note: *If Token Recovery is enabled, verify that no other device on the network is configured as a Full Administrator, with token recovery enabled. To do this, see attributes "TP" and "ER" in the system channel (channel FF00) of every device.*
- **Token Recovery Timeout**
  The amount of time the driver will wait with no detected PUP traffic, before declaring the token dropped. Any valid PUP message detected (sent or received by any device on the network) resets this timeout.
  Typically, this is set to about 1 second. This is the amount of time that no network traffic is detected AFTER PASSING THE TOKEN. As an example, if we pass the token from the EC-BOS^{AX} (EC-Net^{AX} aapup driver) to unit 50, and unit 50 then initiates and talks to other units for 5 second before passing the token back to us, this is OK since we are looking for "idle" time be-

fore saying a token is dead.
As long as unit 50 does not stop talking for more than 1 sec (or whatever period we have this property configured to), we won't attempt to recover the token. Note that it would be OK for unit 50 to pass the token on to someone else—we just look for idle time on the comm line.

- **Token Recovery Count**
  The number of times the token has had to be recovered. This count is reset at startup, or may be reset by invoking the “Reset Token Recovery Count” action (right-click on “Token Pass Config”, and select “Actions”).
- **Pass Token On Transaction Count**
  If enabled, then a count of up to “Transactions Per Token” transactions can be sent before the token is passed to a device on the PUP trunk.
  Note: *The “Pass Token on Timeout”/ “Time Per Token” properties may cause a token pass faster than this if enabled and the timeout expires before this number of messages is reached. In the case where both “Pass Token on Transaction Count” and “Pass Token on Timeout” are enabled, only one of the two conditions must be satisfied to pass the token.*
- **Transactions Per Token**
  The maximum number of transactions that can be transmitted to the bus by the PUP driver, before the token is passed to the next available peer. This count does not include retries (for example, if this property is set to 2, and the “Retry Count” property is set to 3, then up to eight messages `[2*(1+3)=8]` may be sent in an attempt to deliver up to 2 transactions).
- **Pass Token On Timeout**
  If enabled, then transactions may be sent until the timer expires, at which point the token will be passed. This guarantees that the PUP driver does not “monopolize” the token for extended periods of time, even if there are no messages to send.
  Note: *It is recommended you set this to “true” and adjust the “Time Per Token” property to an appropriate value. The reason for this is that if you pass the token only on transaction count, you run the risk if you have no points subscribed, the only transactions you'd be sending are the “ping” messages, and these are typically on the order of minutes, so it can take quite a while for the transaction count to be hit.*
- **Time Per Token**
  The time allowed for token ownership before the token must be passed. The timer is started whenever the token is recovered, or whenever the token is received from a device on the PUP network.
  Note: *If “Pass Token on Transaction Count” and “Pass Token on Timeout” are both enabled, it is recommended to set this value slightly longer than the time it would take to send “Transactions per Token” transactions. In the case where both “Pass Token on Transaction Count” and “Pass Token on Timeout” are enabled, only one of the two conditions must be satisfied to pass the token.*
- **Token Snoop**
  This is a diagnostic display which shows recent activity pertinent to token passing. This display is 8 lines long—usually long enough to diagnose token problems, especially when coupled with the next property. A new snoop line is started every time the token ownership is obtained by the driver (that is, whenever the token is passed from a device on the network to driver’s unit number).

Figure 3-4 *Example Token Snoop*



There are 3 basic types of events shown (see Figure 3-4 above):

- [1>79][79>1]
  Items in square brackets indicate a token pass from one device to another. In this example, device 1 (the driver) passed a token to device 79, and device 79 sent the token back to unit 1.
- [1>5385]<65535><11403><11403><11403><65535>[5385>1]
  Items in angle brackets (like <65535> in this example, represent destination addresses of

messages sent from other devices on the network. In this example, you can see that device 1 passed the token to device 5385, which in turn broadcast a message (to address 65535), sent 3 messages to device address 11403, broadcast a second message (to address 65535), and then sent the token back to address 1.

- If you see any "`!`" embedded in the list, it most likely means someone else on the network is recovering the token. The exclamation point will appear on both sides of the unit number which incorrectly tried to pass a token.
  Token recovery events are denoted by the text "`recover`" in the display.

- **Log Snoop On Token Recovery**
  If set to "true" then every time there is a "recover" event, the "Token Snoop" data is pushed to the Pup Log. This is a valuable tool to diagnose token passing issues.

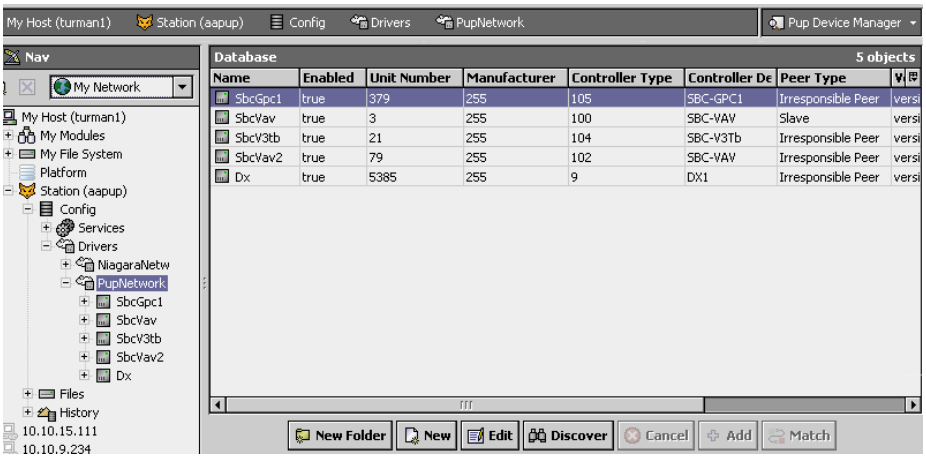
PupNetwork action

The PupNetwork has a single action: **`Sync Time`**. It sends an immediate time sync broadcast message on the PUP network, including the holiday bit setting as specified in the network's "Holiday" property.

Pup Device Manager

The Pup Device Manager (Figure 3-5) is the default view for the PupNetwork, and works similar to other device managers that support online device discovery. See "About the Device Manager" in the *Drivers Guide* for general information.

Figure 3-5 *Pup Device Manager is default view for PupNetwork*



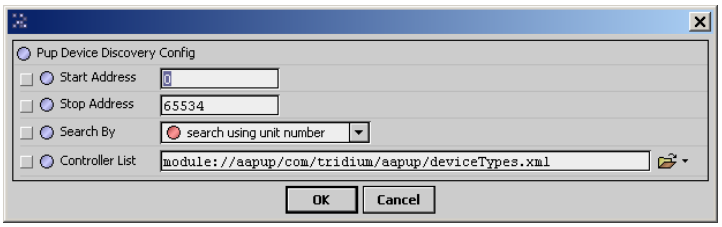
The following sections provide PUP-related details:

- About PUP Device Discovery Config
- Pup Device Manager discover notes

About PUP Device Discovery Config

When you click **`Discover`** in the Pup Device Manager, a discovery config dialog appears, as shown in Figure 3-6.

Figure 3-6 *PUP device discovery config dialog*



Fields in this dialog are explained as follows:

- **Start Address**
  Beginning address range (typically unit number) for PUP device discovery. Default value is 0.
- **Stop Address**

Ending address range (typically unit number) for PUP device discovery. Default value is 65534.

Note: *To learn what devices are on the wire, it is best if you narrow the Start Address and Stop Address range, because the learn process has to loop through and attempt to talk to each and every address in the range. Attempting to talk to 65535 would take quite a long time!*

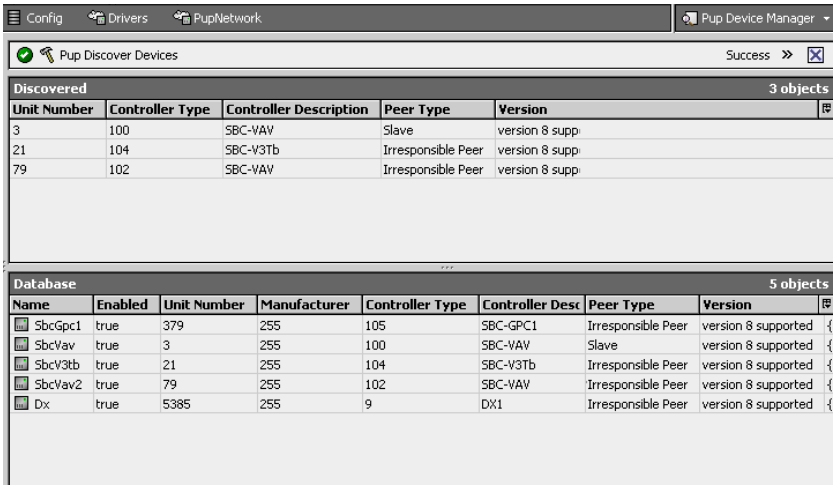
- **Search By**
  Allows selection of either:
  - "search using unit number" — (default) EC-Net^{AX} will search using the specified start and stop address range of unit numbers.
  - "search using serial number" — EC-Net^{AX} will search using the specified start and stop address range as serial numbers (a "back door" search of devices that use the American Auto-Matrix method of serial number to access devices).
- **Controller List**
  Specifies the XML-formatted file to be referenced when doing online "Discovers" of PUP devices, as also specified by the "Device Types File" property of the PupNetwork. The file describes device types, channel names, channel search ranges, and so forth. For more details, see "Modifying DeviceTypes.xml" on page 3-21.

Pup Device Manager discover notes

After clicking **`Discover`**, you can observe the discover device job proceed along the "progress bar" at the top of the device manager view. In addition, a log of addresses searched, along with search results, can be viewed by clicking the small ">>" button to the immediate right of the job bar.

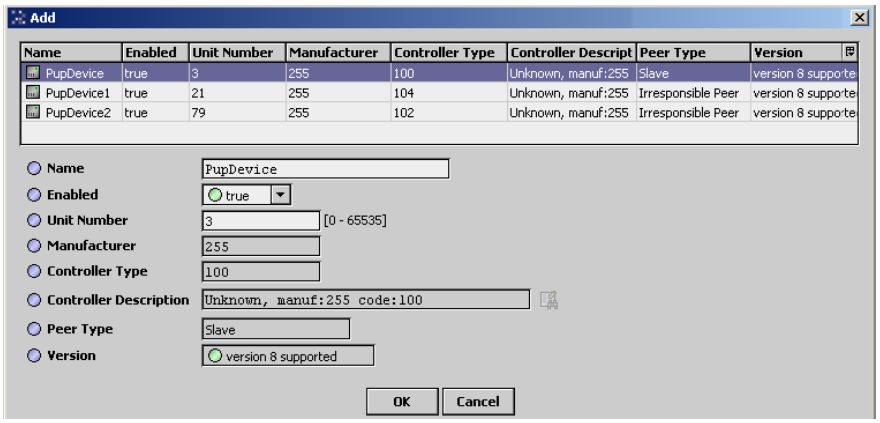
As in other drivers, the "Learn Mode" from a discovery splits the Pup Device Manager into two panes—the top "Discovered" pane and the lower "Database" pane (Figure 3-7).

Figure 3-7 *Learn Mode in Pup Device Manager*



When you click **`Add`** with one or more discovered devices highlighted in the top pane, the "**`Add`**" dialog appears, as shown in Figure 3-8.

Figure 3-8 *Add dialog in Pup Device Manager*



This Add (or Edit) dialog from the Pup Device Manager gives you the opportunity to tweak each device's display name, enabled state, and/or address. Click the **`OK`** button to add the PupDevices to the station database, or click **`Cancel`** to close the dialog without any changes.

You can rerun a device discover as many times as needed. If your PUP network is very large, you may wish to add multiple PupDeviceFolders (using **`New Folder`** button), and run a series of *differently* configured device discovers (Figure 3-6) for each one, using the PupDeviceManager view available with each folder.

PupDevice

A PupDevice provides all configuration parameters necessary for the driver to communicate with a given PUP device. In addition to common components, views, and the Points device extension, a PupDevice contains an "Native Alarm Source Info" container slot (in addition to the standard "Alarm Source Info" container slot). Also unique to a PupDevice is its default "Pup Region Manager" view.

The following subsection provides more details:

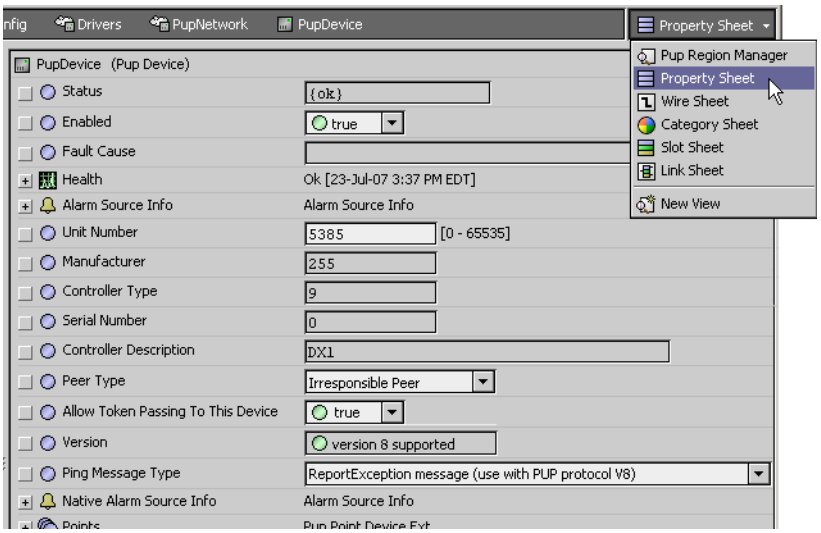
- PupDevice properties

Also see sections "Pup Region Manager" on page 3-13 and "Pup Point Manager" on page 3-15.

PupDevice properties

The PupDevice property sheet is shown in Figure 3-9.

Figure 3-9 *PupDevice property sheet*



These PupDevice properties are described as follows:

- **Status, Enabled, Fault Cause, Health, Alarm Source Info**
  These properties operate the same as for a device object in most drivers. see "Common device

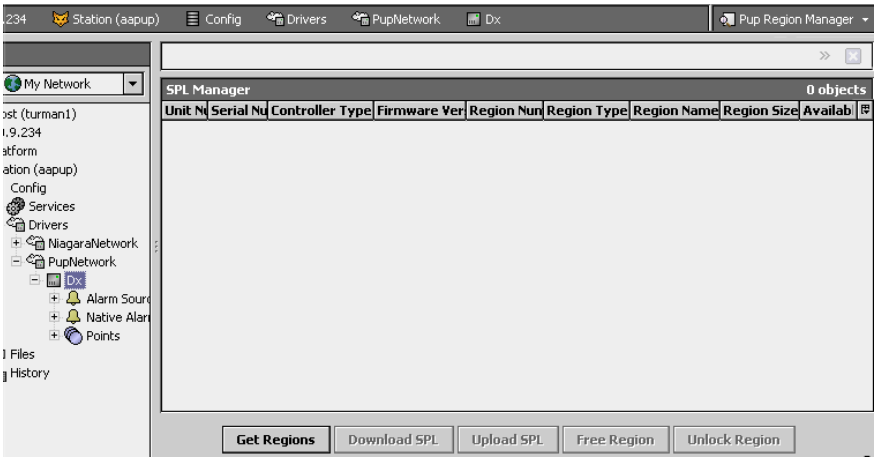
components" in the *Drivers Guide* for general information on these properties.

- **Unit Number**
  The address of target PUP device (the "`ID`" attribute of channel `FF00`).
- **Manufacturer**
  The "`CM`" attribute of channel `FF00` in the device.
- **Controller Type**
  The "`CT`" attribute of channel `FF00` in the device.
- **Serial Number**
  The serial number of the PUP device.
- **Controller Description**
  The common name found in the "Devices Type File" file (property of the network), using the "Manufacturer" and "Controller" as look up parameters.
- **Peer Type**
  The peer type as read from the device's "`TP`" attribute in its system channel. If the "`TP`" attribute does not exist, the Peer Type is assumed to be "`slave`".
- **Allow Token Passing To This Device**
  (Property added in builds 3.1.31, 3.2.17, 3.3.1 or later) Defaults to true. Determines whether or not the device will be added to the PupNetwork's "Peer List" folder, to be included in any token passing.
  This setting has no effect if the Peer Type is "slave". If set to false and the device is already in the network's Peer List folder, it will be removed (and thus excluded from token passing). If set to true and the device is not a "Slave" peer type, the device will be added to the network's Peer List folder (and thus included in token passing).
- **Version**
  The "version 8" bit from the Acknowledge response to the "Say Hello" command.
- **Native Alarm Source Info**
  Configuration data applied to alarms that originate in this PUP device and that are retrieved using the "Report Exception" message. PUP native alarms are normalized and passed to the EC-Net^{AX} alarming subsystems which then display/manage them. Acknowledgments are in turn transmitted back down to the PUP device that originated them.
- **Points**
  The standard Points device extension (container) for all data items (attributes) in this PUP device which need to be polled for data.
- **Allow Time Sync**
  Whether to allow a "directed" (not broadcast) time sync to be transmitted to this device. If enabled by this property, a directed time sync is invoked from an action on the device (right-click the PupDevice, select **`Actions > Sync Time`**). Or, if enabled by this property a time sync also occurs whenever this device's Status property has been {down} and transitions to {ok}.

Pup Region Manager

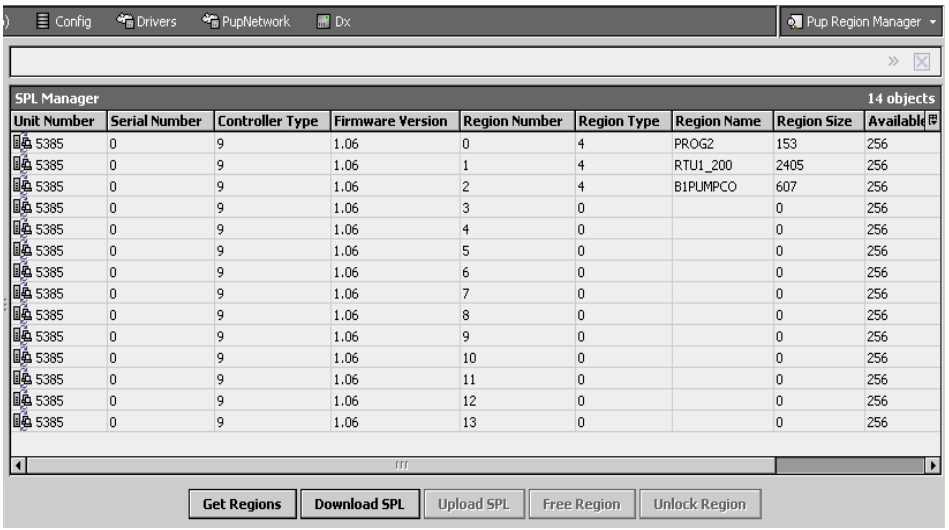
The Pup Region Manager is the default view when you double-click on a PupDevice, and is unique to devices in the PUP driver. This view provides a way to view, upload, and download SPL programs in a PUP device. Figure 3-10 shows an example Pup Region Manager view when first displayed.

Figure 3-10 *Pup Region Manager is default view for PupDevice*



A table-based view, each row represents a unique memory region within a PUP device. Initially, the "**`Get Regions`**" button is the only button active. To fetch a list of regions in a device, you click this button. The view is then populated with a list of regions in the device (if any), as shown in Figure 3-11.

Figure 3-11 *Example Pup Region Manager view after Get Regions*

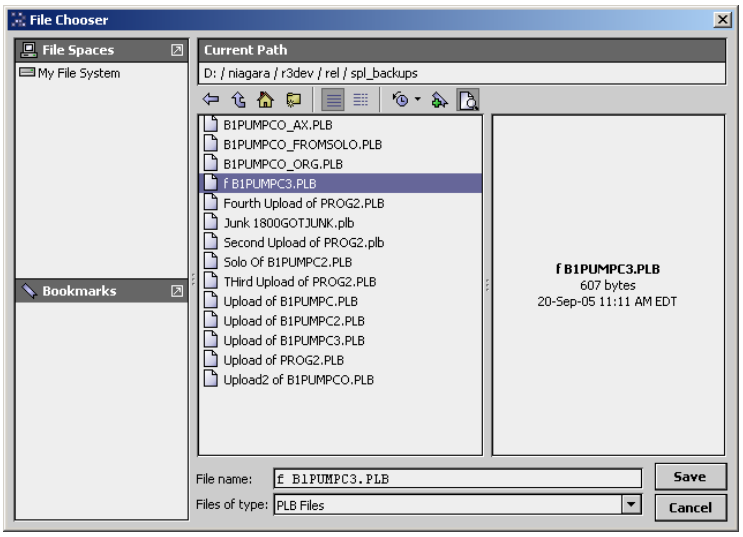


Unit Number	Serial Number	Controller Type	Firmware Version	Region Number	Region Type	Region Name	Region Size	Available
5385	0	9	1.06	0	4	PROG2	153	256
5385	0	9	1.06	1	4	RTU1_200	2405	256
5385	0	9	1.06	2	4	B1PUMPCO	607	256
5385	0	9	1.06	3	0		0	256
5385	0	9	1.06	4	0		0	256
5385	0	9	1.06	5	0		0	256
5385	0	9	1.06	6	0		0	256
5385	0	9	1.06	7	0		0	256
5385	0	9	1.06	8	0		0	256
5385	0	9	1.06	9	0		0	256
5385	0	9	1.06	10	0		0	256
5385	0	9	1.06	11	0		0	256
5385	0	9	1.06	12	0		0	256
5385	0	9	1.06	13	0		0	256

Once the regions in the controller are determined, additional buttons become active or inactive depending on your selection of rows in the table, and the values of the region parameters:

- If there are "free" regions (that is, a region without a "Region Name"), then the **`Download SPL`** button will be active. To download an SPL program to the controller, click this button.
  A File Chooser dialog similar to Figure 3-12 appears:

Figure 3-12 *Dialog from "Download SPL" in Pup Region Manager*



Select a program from the list or navigate to a file under the "My File System" file space to select a program. When you click "**Save**," the program will be downloaded to the next free region.

- The "**Upload SPL**" button of the Pup Region Manager is enabled whenever a region with a non-blank "Region Name" is selected. This allows the transfer of SPL files from the controller to the file space on your EC-Net^{AX} Pro client PC. Again, a file chooser dialog box like the one shown in Figure 3-12 is displayed to allow selection of an existing file to overwrite, or allows entering of a new file name to create a new file. Navigation is limited to your client-side file space.
- The "**Free Region**" is used to remove a SPL program from controller memory and free up the space of other programs to be downloaded. Select a named region out of the list, and click "Free Region." A confirmation dialog as shown in Figure 3-13 appears:

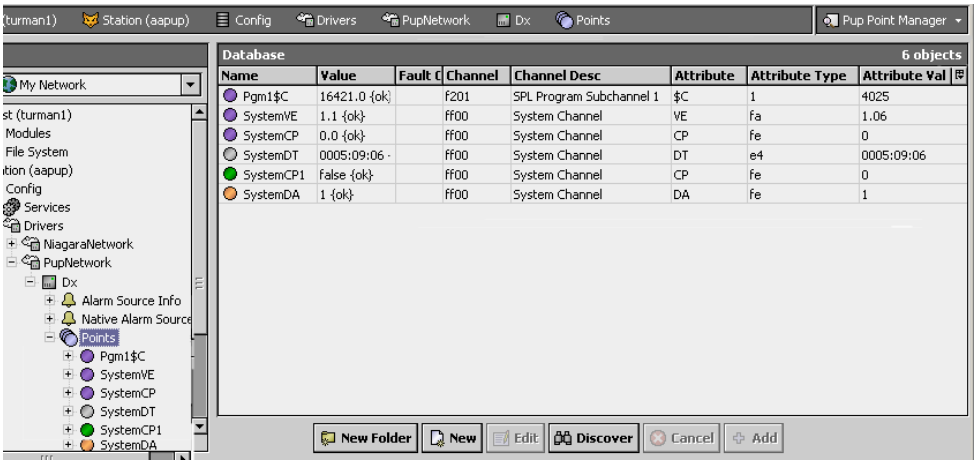
Figure 3-13 *Confirmation dialog for "Free Region" in Pup Region Manager*

Click "**Yes**" or "**No**" or as needed.

Pup Point Manager

The Pup Point Manager (Figure 3-14) is the default view for the "**Points**" device extension for any PupDevice, and works similar to other point managers that support online device discovery. See "About the Point Manager" in the *Drivers Guide* for general information.

Figure 3-14 *Pup Point Manager is default view for Points under PupDevice*



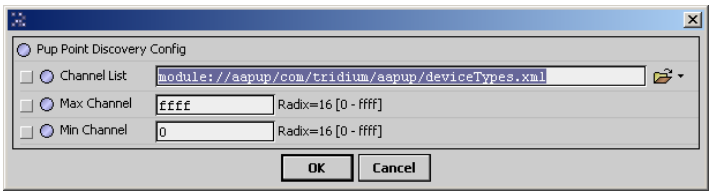
The following sections provide PUP-related details:

- About PUP Point Discovery Config
- Pup Point Manager discover notes

About PUP Point Discovery Config

When you click **`Discover`** in the Pup Point Manager, a discovery config dialog appears, as shown in Figure 3-15.

Figure 3-15 *PUP point discovery config dialog*



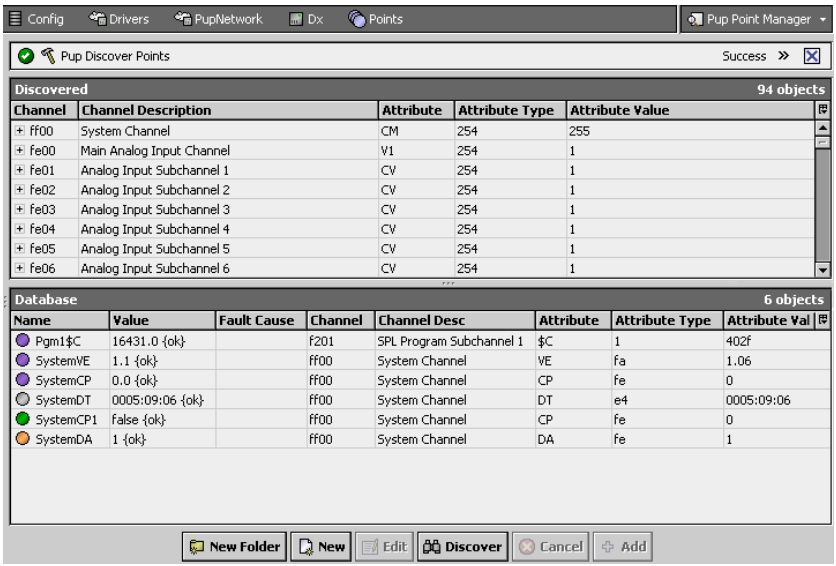
Fields in this dialog are explained, as follows:

- **Channel List**
  Specifies the ord to the XML-formatted file to be referenced when doing online "Discovers" of PUP points, also specified by the "Device Types File" property of the PupNetwork. This file describes known device types and the channels that they contain.
  By default, the ord points to the `deviceTypes.xml` file included within the `aapup.jar`. For more details, see "Modifying DeviceTypes.xml" on page 3-21.
- **Max Channel**
  Maximum channel in the device searched during the points learn. Default value is `ffff`.
- **Min Channel**
  Beginning channel in the device searched during the points learn. Default value is `0`.

Pup Point Manager discover notes

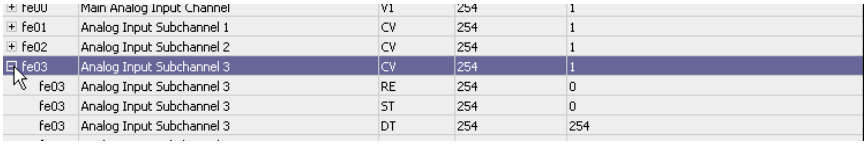
After clicking **OK** in the PUP point discovery config dialog, the discovery “Learn Mode” splits the Pup Point Manager into two panes—the top “Discovered” pane and the lower “Database” pane (Figure 3-16).

Figure 3-16 *Learn Mode in Pup Point Manager*



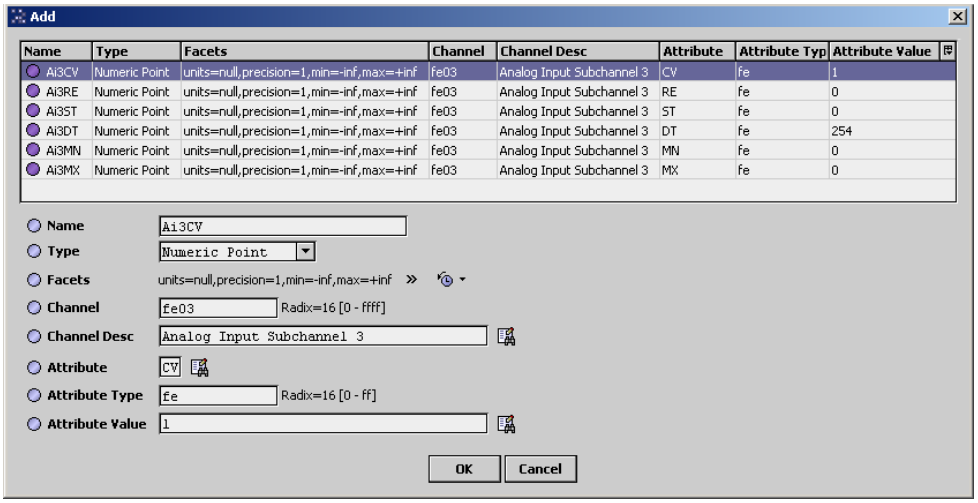
Note that the first (default) attribute of each channel is listed, with a “+” mark in front of the channel number if there are additional attributes in that channel. To access the additional attributes o a channel, click on the “+” to expand, as shown in Figure 3-17.

Figure 3-17 *Expanding discovered channel in Pup Point Manager*



Single or multiple attributes can be added as control points with PupProxyExt extensions by selecting the discovered row(s) in the top pane, and clicking **Add**. Doing so will cause the “**Add**” dialog box to appear, as shown in Figure 3-18.

Figure 3-18 *Add dialog in Pup Point Manager*



The **Add** dialog from the Pup Point Manager lets you edit points individually or in a batch.

Note: *The "Attribute Type" determines which proxy point "Type" can be created. For a list of valid mappings, see "Mapping of PUP Attribute Types to Proxy Extensions" on page 1-3.*

Once the point(s) are satisfactorily edited in the **`Add`** dialog, click the **`OK`** button to create the PUP proxy points corresponding to the channel attributes. For more details, see the next section "PUP proxy points".

You can rerun a point discover as many times as needed. If there are many channels and attributes, you may wish to add multiple PupPointFolders (using **`New Folder`** button), and run a series of *differently* configured point discovers (Figure 3-15) for each one, using the Pup Point Manager view available with each folder.

PUP proxy points

PUP proxy points are similar to other driver's proxy points, as "point-level" components in the EC-Net^{AX} architecture. See "About proxy points" in the *Drivers Guide* for general information.

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

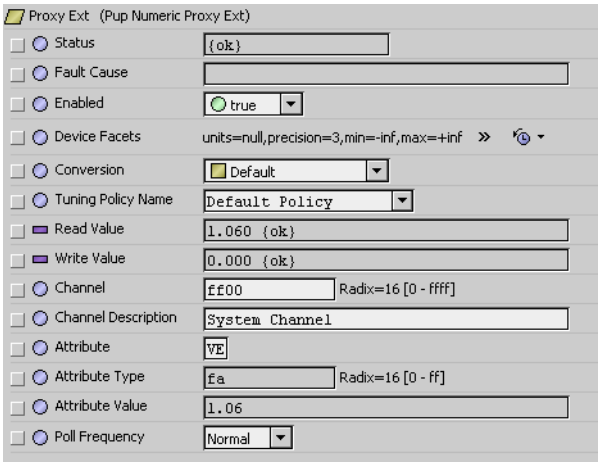
- PupProxyExt properties
- PupProxyExt actions
- NumericPoints with "CV" attribute
- StringPoint actions for time or date attributes

PupProxyExt properties

All PupProxyExt types (PupNumericProxyExt, PupBooleanProxyExt, PupEnumProxyExt, and PupStringProxyExt) share the same set of configuration properties. Any one of the PupProxyExt types is a proxy for one piece of data of interest in a PUP controller. This single piece of data is defined by a combination of the channel and the attribute in that channel.

Figure 3-19 shows the property sheet of an example PupProxyExt.

Figure 3-19 *Property sheet for a PupNumericProxyExt*



In addition to typical ProxyExt properties, the proxy extension in any PUP proxy point includes these additional properties:

- **Channel**
  A hexadecimal number representing the controller channel where the attribute resides.
- **Channel Description**
  A text description of the channel, generally created when the points are learned, using the channel list in the `deviceTypes.xml` file (or a user-selected equivalent file).
- **Attribute**
  The PUP conversion type. This tells the driver how the data returned in the poll message is to be interpreted. This entry must match the published data type for the attribute.
- **Attribute Value**
  (read only) The attribute's value.
- **Poll Frequency**
  The frequency to poll this point in the controller. Choices are either Normal, Fast, or Slow. Poll rates for normal, fast, and slow are determined by the "Poll Scheduler" property of the PupNet-

work.

Note: *Any BooleanWritable PUP proxy point has two additional properties in its proxy extension. See the next section "PUP BooleanWritable notes". Also, there are special considerations for some Numeric Points, depending on attributes assigned. See "NumericPoints with "CV" attribute" on page 3-19.*

PUP BooleanWritable notes

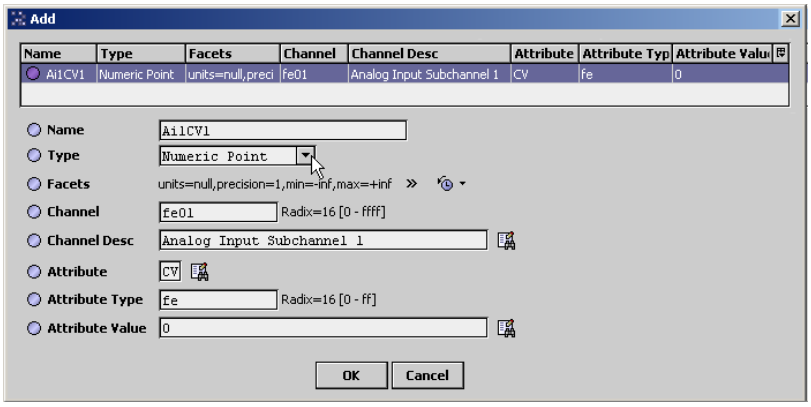
In addition all other PupProxyExt properties, the following two properties apply to the PupBoolean-ProxyExt in a BooleanWritable PUP proxy point:

- **Value to Write If True**
  The value written to the PUP attribute whenever a write of the value "true" is required. Normally, this should be set to "1".
- **Value to Write If False**
  The value written to the PUP attribute whenever a write of the value "false" is required. Normally, this should be set to "0".

NumericPoints with "CV" attribute

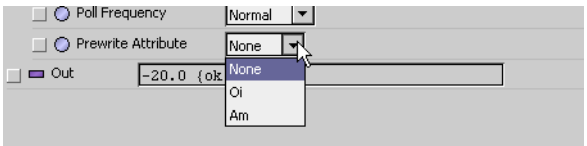
PUP proxy points that are type Numeric Point (read-only) configured with an Attribute property of "`CV`" have a special behavior. Figure 3-20 shows an **Add** dialog in the Pup Point Manager configured to create a Numeric Point with the CV attribute.

Figure 3-20 *Adding NumericPoint with CV Attribute*



In this case, the special behavior is the addition of a command (action) to set an override value to the point. However, this requires some additional configuration to perform correctly. In particular, the "Prewrite Attribute" property of the proxy extension must be set to the appropriate value, as shown being done in Figure 3-21.

Figure 3-21 *Prewrite Attribute value being selected*



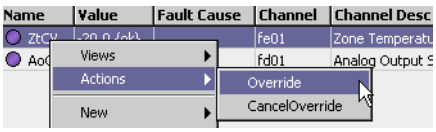
This property is invisible until an attribute value of "`CV`" is entered, whereupon it is listed in the property sheet of the proxy extension. The value of Prewrite Attribute defaults to "`None`". If the value is changed to "`Oi`" or "`Am`", then a dynamic action is added to the proxy point to allow setting of the CV value.

These are the basic rules to be able to use the set action:

1. The PUP proxy point Type must be **`Numeric Point`** (Numeric Writable *does not work*).
2. The "Attribute" property must be set to "`CV`". This makes the "Prewrite Attribute property visible in the PupNumericProxyExt.
3. The "Prewrite Attribute" property must be set to "`Oi`" or "`Am`" depending on whether the OI attribute or the AM attribute must be written before the CV value can be written. This varies from channel-to-channel, and/or device-to-device.

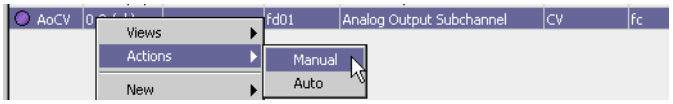
4. If the “Prewrite Attribute” is “`Oi`” or “`Am`”, than the appropriate pair of actions is added to the proxy point, as follows:
   - If the Prewrite Attribute is set to “`Oi`”, then the actions “Override” and “CancelOverride” are added to the proxy point, as shown in Figure 3-22.

Figure 3-22 *Override actions from “Oi” value for Prewrite Attribute*



   - Invoking the “Override” action sets the “`OI`” attribute in the PUP controller to 1, and writes the user-entered value to the attribute “`CV`”.
   - Invoking the “CancelOverride” action resets the “`OI`” attribute in the PUP controller to 0.
   - If the Prewrite Attribute is set to “`Am`”, then the actions “Manual” and “Auto” are added to the proxy point, as shown in Figure 3-23.

Figure 3-23 *Manual/auto actions from “Am” value for Prewrite Attribute*



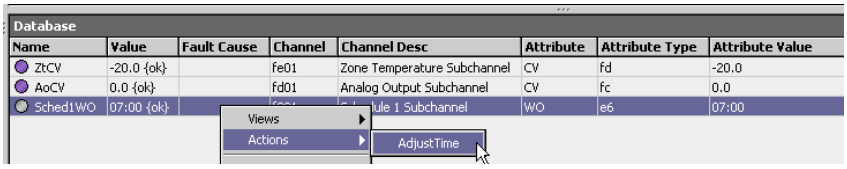
   - Invoking the “Manual” action sets the “`AM`” attribute in the PUP controller to 1, followed by a write of the user-entered value to the attribute “`CV`”.
   - Invoking the Auto action resets the “`AM`” attribute in the PUP controller to 0.

StringPoint actions for time or date attributes

PUP proxy points configured with an Attribute property that corresponds to a *time* or *date* must be created on a **`String Point`** type control point. In this case, there is an additional command (action) available on the proxy point that allows the user to set time or date values.

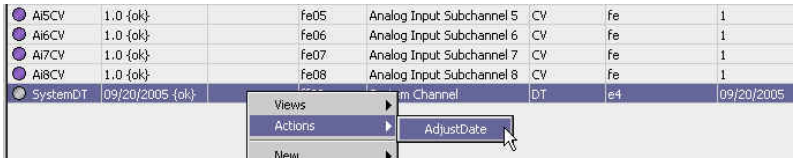
- If the Attribute Type property is “`e6`” or “`e7`”, then an “Adjust Time” action is added to the proxy point, as shown in Figure 3-24.

Figure 3-24 *Adjust Time action on StringPoint with “e6” or “e7” Attribute Type*



- If the Attribute Type property is “`e3`” or “`e4`”, then an “Adjust Date” action is added to the proxy point, as shown in Figure 3-25.

Figure 3-25 *Adjust Date action on StringPoint with “e3” or “e4” Attribute Type*



PupProxyExt actions

For any of the PUP proxy points, its Pup*Type*ProxyExt provides two actions. Note that these actions are on the ProxyExt itself, and not the parent control point.

These actions are briefly described as follows:

- **Force Read**
  Results in an immediate poll of the attribute in the PUP device’s channel.
- **Force Write**
  Forces an attempt to write from EC-Net^{AX} to the attribute in the PUP device’s channel.

Modifying DeviceTypes.xml

The `DeviceTypes.xml` file is included in the `aapup.jar`, which you locally extract using WinZip or some other zip file extraction program. Once extracted, you can use a text editor to modify it.

You can then save and rename it, and copy it back to the EC-BOS^{AX} (using the File Transfer Client view when platform connected to that EC-BOS^{AX}). Typically, you copy the modified file under the `stations` folder of the target EC-BOS^{AX}. Then you can reference it (by a file ord) when configuring the PupNetwork or doing a discover.

Note: *A modified file is recommended to be copied to a different location, rather than be re-jar'ed back into the `aapup.jar`, as that module file is replaced whenever the `aapup` module is updated.*

Modifying device types would be useful, for instance, to change the Channel Names, channel search ranges, add/remove channels, add new device types, and so forth. The ord to this file is found in the "Device Types File" property of the PupNetwork, or in the popup (config discovery) dialogs when doing a discovery in the Pup Device Manager or Pup Point Manager.

See the sections "XML syntax" and "Example: HX1 and GPC1" for more details.

XML syntax

The `DeviceTypes.xml` file used by the PUP driver is in XML format, so some knowledge of xml encoding is required, but not too difficult to follow. In general, do not edit the first line. Comment lines (each starts with "`<!--`" and ends with "`-->`") may be edited, but have no effect on the parsing of the file.

Lines within the `<deviceTypes>` section can be edited to modify the device and point learn processes.

Each `<device>` entry in the `<deviceTypes>` section should always have the following items:

- `controllerType` — a text description.
- `ct` — integer for the controller type, corresponds to the ct attribute.
- `cm` — integer for the controller manufacturer, corresponds to the cm attribute.
- `snc` — serial number channel, the channel that contains the serial number.
- `sna` — serial number attribute, the attribute that contains the serial number.
- `rdChCmd` — does the controller support the "read channel" command? This is a feature of later generation controllers that allows the point discovery process to determine dynamically the list of channels in the device. If set to true, channel entries in this file for the device will not be used and may be omitted.
- `<channel address>` — one entry for each channel that is desired to be discovered in the device. To skip any channel during a learn, then remove the corresponding line from the device. Additional channels may be added as needed, or descriptions/prefixes may be modified as desired, using the following syntax:
  - `channel address` — the Hex integer address of a channel.
  - `desc` — the long description of a channel, used in learn displays and for naming point folders.
  - `prefix` — the short description of a channel, used as a default prefix for point names created during the learn.

See the next "Example: HX1 and GPC1" section for two example `<device>` entries.

Example: HX1 and GPC1

The following lines represent a “snippet” of the DeviceTypes.xml file.

```
<deviceTypes>
 <device   controllerType="HX1"        ct="0"   cm="255"  snc="8000" sna="SN" rdChCmd="false">
   <channel address="FF00" desc="System Channel"                                  prefix="system"/>
   <channel address="FF01" desc="System Subchannel"                              prefix="system2"/>
   <channel address="FE00" desc="Main Analog Input Channel"                         prefix="ai"/>
   <channel address="FE01" desc="Analog Input Subchannel 1"                         prefix="ai1"/>
   <channel address="FE02" desc="Analog Input Subchannel 2"                         prefix="ai2"/>
   <channel address="FE03" desc="Analog Input Subchannel 3"                         prefix="ai3"/>
   <channel address="FE04" desc="Analog Input Subchannel 4"                         prefix="ai4"/>
   <channel address="FE05" desc="Analog Input Subchannel 5"                         prefix="ai5"/>
   <channel address="FD00" desc="Main Analog Outputs Channel"                       prefix="ao"/>
   <channel address="FD01" desc="Analog Output Subchannel 1"                        prefix="ao1"/>
   <channel address="FD02" desc="Analog Output Subchannel 2"                        prefix="ao2"/>
   <channel address="FC00" desc="Binary Inputs Channel"                             prefix="bi"/>
   <channel address="FB00" desc="Main Digital Outputs Channel"                      prefix="do"/>
   <channel address="FB01" desc="Digital Outputs Subchannel 1"                      prefix="do1"/>
   <channel address="FB02" desc="Digital Outputs Subchannel 2"                      prefix="do2"/>
   <channel address="FB03" desc="Digital Outputs Subchannel 3"                      prefix="do3"/>
   <channel address="FB04" desc="Digital Outputs Subchannel 4"                      prefix="do4"/>
   <channel address="FA00" desc="Setpoints Channel"                                 prefix="sp"/>
   <channel address="FA01" desc="Economizer Setpoints Subchannel"                   prefix="econ"/>
   <channel address="FA02" desc="Cooling Reset Subchannel"                       prefix="coolReset"/>
   <channel address="FA03" desc="Heating Reset Subchannel"                       prefix="heatReset"/>
   <channel address="FA04" desc="Calculated Setpoints Subchannel"                   prefix="calcSp"/>
   <channel address="FA05" desc="VVT Setpoints Subchannel"                           prefix="vvtSp"/>
   <channel address="F900" desc="Main Schedule Channel"                          prefix="mainSched"/>
   <channel address="F901" desc="Schedule 1 Subchannel"                             prefix="sched1"/>
   <channel address="F902" desc="Schedule 2 Subchannel"                             prefix="sched2"/>
   <channel address="F903" desc="Schedule 3 Subchannel"                             prefix="sched3"/>
   <channel address="F904" desc="Optimized start Subchannel"                       prefix="optStart"/>
   <channel address="8000" desc="Factory"                                           prefix="factory"/>
   <channel address="5F00" desc="Packaged Heat Pump Subchannel 1"                   prefix="htPmp1"/>
   <channel address="5F01" desc="Packaged Heat Pump Subchannel 2"                   prefix="htPmp2"/>
 </device>
<device   controllerType="SBC-GPC1"   ct="105"   cm="255"   snc="FF00" sna="SN" rdChCmd="true"/>
</deviceTypes>
```

CHAPTER 4

PUP (aapup) 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 EC-Net^{AX} Pro menu, or press F1 while the view is open.

Summary information is provided here about the different PUP views.

Plugin Guides Summary

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

- PupDeviceManager
- PupPointManager
- PupRegionManager

aapup-PupDeviceManager

The **`Pup Device Manager`** is the default view of a PupNetwork. Use the Pup Device Manager to add, edit, and access PUP device components (PupDevices). As with many other drivers, after setting network configuration properties, you can use an online "Discover" to learn available (PUP) devices. For general information, see "About the Device Manager" in the *Drivers Guide*.

Added devices appear in the Database table. For more details, see "Pup Device Manager" on page 3-10.

Database table

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

- **Name**
  Name of the device-level component that represents the PUP device.
- **Enabled**
  Indicates whether the device component is enabled (true) or disabled (false).
- **Unit Number**
  Reflects the address of PUP device (the "ID" attribute of channel FF00).
- **Manufacturer**
  Reflects the "CM" attribute of channel FF00 in the PUP device.
- **Controller Type**
  Reflects the "CT" attribute of channel FF00 in the PUP device.
- **Controller Description**
  The common name looked up the "Devices Types File" file, using "Manufacturer" and "Controller Type" as look up parameters.
- **Peer Type**
  Reflects the peer type as read from the device's "TP" attribute of the system channel. If the "TP" attribute does not exist, the Peer Type is assumed to be "slave."
- **Version**
  The "version 8" bit from the Acknowledge response to the "Say Hello" command.
- **Exts**
  (Build 3.0.104, 3.1.28, 3.2.2 and higher only) Provides shortcut access to default manager views for the component's device extensions—in this case, "Points" (PupPointManager).

- **Status**
  Reflects the status of the device, typically “ok” unless “disabled” or “down.”

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

- **Path**
  Station path of the device-level component, relative to the root.
- **Type**
  Device-level component type, currently only PupDevice.

aapup-PupPointManager

Use the Pup Point Manager to add, edit, and access PUP proxy points under the Points extension of a selected PupDevice, or in a PupPointFolder. The PupPointManager is the default view on both these components. To view, *double-click* the Points extension or PupPoint-Folder, or right-click and select **`Views > Pup Point Manager`**.

As in some other point managers, there is a Discovered table (if in Learn mode) and a Database table. Discovery produces an intermediate popup dialog, in which you can enter a range of channels to limit discovery, and/or point to an alternate “Channel List” file.

For more details, see “Pup Point Manager” on page 3-15.

Discovered table

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

- **Channel**
  The PUP channel with one or more attributes, each as a row.
- **Channel Description**
  Text descriptor for the channel, sourced from the “Channel List” file.
- **Attribute**
  Two-character text field that determines the attribute for this channel.
- **Attribute Type**
  The PUP conversion type, which must match the published data type for the attribute.
- **Attribute Value**
  Value of the attribute upon discovery.

Database table

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

- **Name**
  EC-Net^{AX} name of the proxy point.
- **Value**
  Current last polled value of the attribute, reflecting status and facets.
- **Fault Cause**
  String describing the cause of the proxy point status fault, if any.
- **Channel**
  PUP controller channel (in hexadecimal format) where the attribute resides.
- **Attribute Type**
  The PUP conversion type (in hexadecimal format).
- **Attribute Value**
  Native value of the attribute.

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

- **Path**
  Station path of the proxy point component, relative to the root.
- **Type**
  EC-Net^{AX} type of component, as either a Pup Point Folder (for a folder) or a type of control point if an PUP proxy point (for example, Boolean Point, Boolean Writable, Numeric Point, and so on).
- **Facets**
  Reflect the facets in use by the proxy point.
- **Enabled**
  Reflects whether proxy point is enabled (true) or disabled (false).
- **Read Value**
  Reflects current read value in point’s Pup*Type*ProxyExt.
- **Write Value**

Reflects current write value (if any) in point's Pup*Type*ProxyExt.

aapup-PupRegionManager

The Pup Region Manager is the default view of any PupDevice under the PupNetwork. To view, *double-click* the device, or right-click and select **`Views > Pup Region Manager`**. Use the Pup Region Manager to view, upload, or download SPL program regions in a PUP device. A table-based view, each row represents a unique memory region within the device.

Although not a standard view in the driver architecture, it is similar to other manager views in that is a table-based view, with online discovery ("Get Regions"). For more details, see "Pup Region Manager" on page 3-13.

CHAPTER 5

PUP (aapup) Component Guides

These component guides provides summary help on PUP components.

Component Guides Summary

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

- PupBooleanProxyExt
- PupDevice
- PupDeviceFolder
- PupEnumProxyExt
- PupNetwork
- PupNumericProxyExt
- PupPeerListFolder
- PupPointDeviceExt
- PupPointFolder
- PupStringProxyExt
- PupDeviceFolder

aapup-PupBooleanProxyExt

PupBooleanProxyExt is the proxy extension for either a PUP_Boolean Point (read-only) proxy point or a PUP_BooleanWritable proxy point. Like any Pup*Type*ProxyExt, it represents a single piece of data defined by a combination of the channel and the attribute in the channel, and has standard proxy extension properties such as Status and Enabled, among others (see "ProxyExt properties" in the *Drivers Guide* for related details). For additional details, see "PUP proxy points" on page 3-18.

aapup-PupDevice

PupDevice is the "device-level" component in a PupNetwork, and represents a specific PUP device. It contains all configuration parameters necessary for the driver to communicate with that device. A PupDevice has a Points device extension (PupPointDeviceExt) that contains all proxy points for polling.

A PupDevice also contains *two* "AlarmSourceInfo" container slots:

- Alarm Source Info — To configure the alarm routing and formatting of "device up/down" alarms from the PupNetwork's monitor (ping) process.
- Native Alarm Source Info — To configure the alarm routing and formatting of alarms that originate in the PUP device, and are retrieved using the "Report Exceptions" message.

A PupDevice has the standard device component properties such as status and enabled (see "Common device components" in the *Drivers Guide* for general information). The default view for a PupDevice is the PupRegionManager. For more PupDevice details, see "PupDevice" on page 3-12.

Actions available on a PupDevice are as follows:

- Ping — Sends a ping monitor request to verify device "health. "
- Sync Time — Sends a directed time sync message to this device, providing that the device's "Allow Time Sync" property is set to true.

- Read Device Values — Uploads device-specific information from the PUP device.

aapup-PupDeviceFolder

PupDeviceFolder is the PUP driver implementation of a folder under a PupNetwork. Usage is optional. Each PupDeviceFolder has its own PupDeviceManager view.

You can use the **`New Folder`** button in the PupDeviceManager view to add a PupDeviceFolder. It is also available in the **`aapup`** palette.

aapup-PupEnumProxyExt

PupEnumProxyExt is the proxy extension for either a PUP_Enum Point (read-only) proxy point or a PUP_EnumWritable proxy point. Like any Pup*Type*ProxyExt, it represents a single piece of data defined by a combination of the channel and the attribute in the channel, and has standard proxy extension properties such as Status and Enabled, among others (see "ProxyExt properties" in the *Drivers Guide* for related details). For additional details, see "PUP proxy points" on page 3-18.

aapup-PupNetwork

PupNetwork is the top-level component for the PUP driver in a station. It provides configuration parameters necessary for the driver to communicate with a network of PUP devices, and is effectively a "host" PUP device in a network of PUP devices.

The PupNetwork component has the typical collection of slots and properties as most other network components. For details, See "Common network components" in the *Drivers Guide*. In addition, the PupNetwork has properties unique to operation as a master or peer in a PUP system. For more details, see "About the PUP Network" on page 3-6.

aapup-PupNumericProxyExt

PupNumericProxyExt is the proxy extension for either a PUP_Numeric Point (read-only) proxy point or a PUP_NumericWritable proxy point. Like any Pup*Type*ProxyExt, it represents a single piece of data defined by a combination of the channel and the attribute in the channel, and has standard proxy extension properties such as Status and Enabled, among others (see "ProxyExt properties" in the *Drivers Guide* for related details). For additional details, see "PUP proxy points" on page 3-18.

aapup-PupPeerListFolder

PupPeerListFolder (default name `Peer List`) is a frozen slot on a PupNetwork. It contains a dynamic list of devices which require the token to be passed to them. Whenever the driver determines it is time to pass the token to a device on the network, it selects the device from this list which has not possessed the token in the longest amount of time (in other words, the "oldest").

aapup-PupPointDeviceExt

PupPointDeviceExt (default name `Points`) is the container for PUP proxy points under a PupDevice. Proxy points represent PUP attributes of channels in the device that need to be polled for data. It operates as in most other drivers; see "About the Points extension" in the *Drivers Guide* for general information. The default and primary view for the Points extension is the PupPointManager.

aapup-PupPointFolder

PupPointFolder is an optional container for PUP proxy points. You can use the **`New Folder`** button in the PupPointManager view to add a PupPointFolder. It is also available in the **`aapup`** palette. Each PupPointFolder has its own PupPointManager view.

aapup-PupStringProxyExt

PupStringProxyExt is the proxy extension for either a PUP_String Point (read-only) proxy point or a PUP_StringWritable proxy point. Like any Pup*Type*ProxyExt, it represents a single piece of data defined by a combination of the channel and the attribute in the channel, and has standard proxy extension properties such as Status and Enabled, among others (see "ProxyExt properties" in the *Drivers Guide* for related details). For additional details, see "PUP proxy points" on page 3-18.

Distech Controls Inc.
4005-B Boul. Matte
Brossard, QC J4Y 2P4
Tel: 1-800-404-0043
Tel: 1-450-444-9898
www.distech-controls.com

05DI-UGPUPDR-10