ECHELON®

IzoT® BACnet® Developer's Guide

Develop BACnet applications using the FT 6000 EVK and a Series 6000 processor.

078-0507-01C

Contents

Preface

This document describes how to develop applications for the Echelon Series 6000 processors that have LON®, LonTalk®/IP, BACnet®/IP, and BACnet MS/TP interfaces.

Purpose

This document is intended for device manufacturers (OEMs) who wish to create products based on Echelon Series 6000 processors having LON, LonTalk/IP, BACnet/IP, and BACnet MS/TP interfaces.

It details the required software and hardware setup and demonstrates the development process using working examples targeted for the FT 6000 EVB.

Related Documentation

The following table lists related Echelon documentation that can be useful when developing IzoT and LON devices using Series 6000 chips that also support BACnet/IP and BACnet MS/TP. The PDF files are installed in the Echelon IzoT NodeBuilder Software program folder when you install the IzoT NodeBuilder Tool. The latest versions of these manuals are available from the Echelon website at: *www.echelon.com/docs*.

Title	Part Number	Description
FT 6000 EVK Quick Start Guide	078-0506-01D	This manual describes how to install the FT 6000 EVK software, use the IzoT Router, and use the examples.
FT 6000 EVB Hardware Guide	078-0504-01A	This manual describes the hardware for the FT 6000 EVB evaluation boards that are included with the FT 6000 EVK.
FT 6000 EVB Evaluation Board Schematics	012-1460-51A	This document provides the schematic diagrams for the FT 6000 EVB.
IzoT Manual	See *www.echelon.com/docs/izot*	This online manual at *www.echelon.com/docs/izot* describes the IzoT Platform and how to use the IzoT Router, IzoT SDK, and IzoT CPM 4200 Wi-Fi EVK.
Introduction to the LONWORKS Platform	078-0183-01B	This manual provides an introduction to the ISO/IEC 14908-1 (ANSI/EIA/CEA-709.1 and EN 14908-1) Control Networking Protocol, and provides a high-level introduction to LONWORKS networks and the tools and components that are used for developing, installing, operating, and maintaining them.

Title	Part Number	Description
IzoT NodeBuilder User's Guide	078-0516-01A	This manual describes how to develop LON devices and applications using the IzoT NodeBuilder Development Tool.
I/O Model Reference for Smart Transceivers and Neuron Chips	078-0392-01C	This manual provides information about the I/O models used by Echelon's Neuron Chips and Smart Transceivers. It includes hardware and software considerations for each of the I/O models.
Neuron C Programmer's Guide	078-0002-01I	This manual describes how to write programs using the Neuron C Version 2.2 programming language.
Neuron C Reference Guide	078-0140-01G	This manual provides reference info for writing programs using the Neuron C Version 2.2 programming language.
IzoT Resource Editor User's Guide	078-0508-01A	This manual describes how to use the IzoT Resource Editor to create and edit resource file sets and resources such as functional profile templates (*profiles*), network variable (*datapoint*) types, and configuration property (configuration *datapoint*) types.
NodeLoad User's Guide	078-0286-01G	This manual details the usage of the NodeLoad application
NodeUtil User's Guide	078-0438-01B	This manual details the usage of the NodeUtil application

Technical Support

If you have technical questions that are not answered by this document, or by the related documentation, you can obtain technical support via e-mail to *support@echelon.com*.

See *www.echelon.com/support* for more information on Echelon's support services. See *www.echelon.com/training* for online training and for more information on Echelon's training services.

1

Getting Started with BACnet

This chapter provides the information to get started with developing BACnet/IP and BACnet MS/TP applications for the IzoT Series 6000 processors.

Overview of BACnet for the Series 6000 Processors

The BACnet stack for the IzoT Series 6000 processors is a fully BACnet compliant protocol stack that runs with the LonTalk/IP protocol stack in the Series 6000 processors, and shares and maps to the same network variables (datapoints) as the LonTalk/IP stack, all under the developer's control. The BACnet stack for the IzoT Series 6000 processors is called the *Neuron BACnet Stack*.

Compared to the ISO/IEC 14908-1 protocol, BACnet has a very simple data model. *BACnet objects* typically represent a single value. Each BACnet object is embellished with multiple BACnet properties, and in particular the *present value*, which contains the current value of the BACnet object.

Every BACnet device has to contain a Device object which has to have two unique properties, the (device) *object identifier* and (device) *object name*, which must be unique across all devices on a single BACnet internetwork. These are therefore the first two settable parameters of any BACnet stack.

You can implement a *BACnet server* using the Neuron BACnet Stack. As a BACnet server, BACnet objects implemented by your device can be read and written by a *BACnet client* such as a *BACnet workstation*. Your BACnet server can expose scalar and structured network variables, and fields of network variables, to a BACnet client as BACnet objects, where the present value property of each object contains the actual data. Other required BACnet properties are provisioned by the stack.

You must have a free-of-charge vendor ID allocated by ASHRAE to develop devices using the Neuron BACnet Stack. Details on applying for this can be found under *Other Resources* below.

The capability of a BACnet device is shared via the *Protocol Implementation Conformity Statement*, or *PIC statement* (*PICS*). A PIC statement for the Neuron BACnet Stack is available, but you will have generate one for each BACnet device type that you create.

You can create a BACnet device that conforms to an appropriate BACnet device profile, specified by a list of *BACnet Interoperability Building Blocks* or *BIBBs*. BIBBs define sets and groupings of functionality that can be easily compared from device to device, to determine which BACnet features should be interoperable between devices. You can choose the device profile for each device type you develop, and you can publish the conformance to this in a PIC statement that you create for your device. Appropriate device profiles for a Series 6000 based device are the following:

B-SS	-	Smart Sensor
B-SA	-	Smart Actuator
B-ASC	-	Application Specific Controller
B-AAC	-	Advanced Application Controller

BACnet conformance is tested by BTL, or the BACnet Testing Laboratories, under the auspices of BACnet International. The Neuron BACnet Stack has been tested for conformance, but is not BTL approved. You can submit the BACnet products that you develop with the Neuron BACnet Stack to BTL for approval. BTL approval is not required, but is useful for marketing purposes.

The Neuron BACnet Stack supplies the following:

- All required BACnet objects, properties, and services for a simple BACnet device, including the following:
  - B-ASC
  - Device Object
  - Read Property Service
- The following BACnet object types:
  - Analog Input
  - Analog Output
  - Binary Input
  - Binary Output

BACnet supports multiple physical layers, the most popular being BACnet/IP (Ethernet and other IP transports) and BACnet MS/TP (RS-485). In the case of the Neuron BACnet Stack used with the IzoT FT 6050 Smart Transceiver, the physical communications medium is FT (TP/FT-10), as defined by the ISO/IEC 14908-2 standard, with a LonTalk/IP transport layer, or MS/TP (RS-485) with a BACnet MS/TP transport layer. In the case of FT, the Neuron BACnet Stack complies with BACnet/IP and appears to other third party BACnet devices as a BACnet/IP device. In the case of MS/TP, the Neuron BACnet Stack complies with BACnet MS/TP and appears as a BACnet MS/TP device.

When using the FT 6050 Smart Transceiver with BACnet MS/TP, the firmware uses the Series 6000 hardware SCI/SPI port and one additional IO pin as an RS-485 TX enable signal. For cases where the hardware SPI port is used to communicate with a co-processor or other peripheral device, a software library is installed with the IzoT NodeBuilder software that provides a software substitute for the hardware SPI port. See **SwSpi.h** in your LONWORKS **NeuronC\Include** folder.

What's New

This documentation describe how to use Release 1.5 of the Neuron BACnet Stack. This is the second release of the Neuron BACnet Stack, and adds the following key features:

- BACnet MS/TP support in addition to BACnet/IP-FT support.
- Change of Value (COV) support
- Read Property Multiple support
- Dynamic BACnet object type support
- Priority array access
- High-resolution scaling

Data Formatting

LON and IzoT data types typically use SI units and rely on data formatting to present data in other formats, such as imperial units, however, BACnet servers may need to publish data in imperial units directly to the network.

An example of mapping and scaling for imperial units is provided in the BACevb example, which is discussed later in this document.

Software Requirements

The following software is required to use the Neuron BACnet Stack.

From the Echelon IzoT FT 6000 EVK:

- IzoT Net Server
- IzoT Commissioning Tool (OpenLNS CT with the IzoT Net Server installed)
- IzoT NodeBuilder Software – the minimum version to support BACnet MS/TP is 4.30.60, the latest version is available *here*

The installation procedures for the above software are detailed in the *FT 6000 EVK Quick Start Guide*

From ConnectEx:

- IzoT BACnet Browser which can be found *here*,

Hardware Requirements

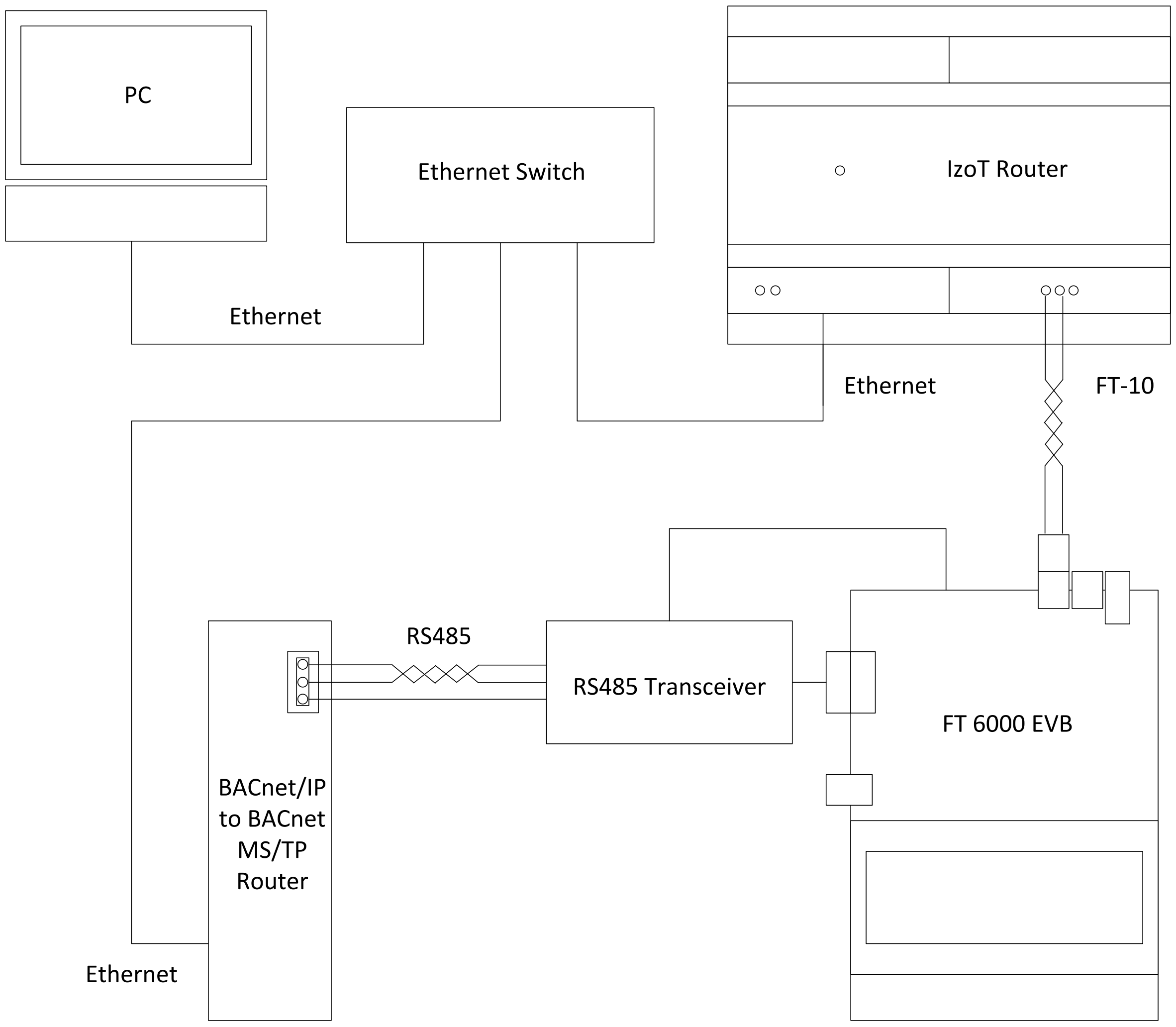
The following hardware is required to develop BACnet/IP applications with the FT 6050 Smart Transceiver:

- IzoT Router (included with the FT 6000 EVK)
- FT 6000 EVB (included with the FT 6000 EVK)
- A computer with the FT 6000 EVK software installed
- Ethernet switch or hub

The following additional hardware is required to develop BACnet MS/TP applications with the FT 6050 Smart Transceiver:

- BACnet/IP to BACnet MS/TP router
- RS-232 to RS-485 level shifter

You can install the hardware as shown in the following diagram to develop BACnet/IP and BACnet MS/TP applications with the FT 6000 EVK.



The IzoT Router includes a BACnet router that is used to route BACnet/IP packets from the Ethernet LAN channel to the FT LON channel. This device also acts as a LonTalk/IP router between the LAN and LON channels. The LonTalk/IP router enables you to load and test applications on the FT 6000 EVB or your custom hardware using the IzoT Commissioning Tool included with the FT 6000 EVK. The IzoT Router is available in models with and without a BACnet router. The model of the IzoT Router included with the FT 6000 EVK includes the BACnet router option. The BACnet router enables you to interface workstations on the BACnet/IP Ethernet channel with your BACnet/IP applications on the LON FT channel.

You can use the FT 6000 EVB to run the examples included with the FT 6000 EVK, and you can also use it for application development. One of the examples is an FT 6000 EVB application that simultaneously provides a LonTalk/IP, BACnet/IP, and BACnet MS/TP interface.

For the BACnet MS/TP connection, a BACnet/IP to MS/TP Router is required. You can use any compliant BACnet/IP to MS/TP router. The Neuron BACnet Stack has been tested with the *BASrouterLX* from Contemporary Controls.

You can use either the FT 6000 EVB DB-9 connector or the EVB I/O pins to attach the FT 6000 EVB to an MS/TP channel.

To use the FT 6000 EVB DB-9 connector for the MS/TP connection you will need:

- An RS-232 to RS-485 level shifter *such as this one* from Hexin The Hexin level shifter is suitable for a connection to only a few devices, but it is self-powered.
- A straight through DB-9 cable *such as this one*.
- A male-to-male null-modem *such as this one*.

To use the FT 6000 EVB I/O pins and an external RS-232 to RS-485 convertor for the MS/TP connection you will need:

- A single board convertor such as the *TEL00070* which provides a simple 4-wire connection to the FT 6000 EVB. The board takes care of enabling and disabling the transmit output of the RS-485 transceiver and therefore does not need a connection to IO9 of the EVB.

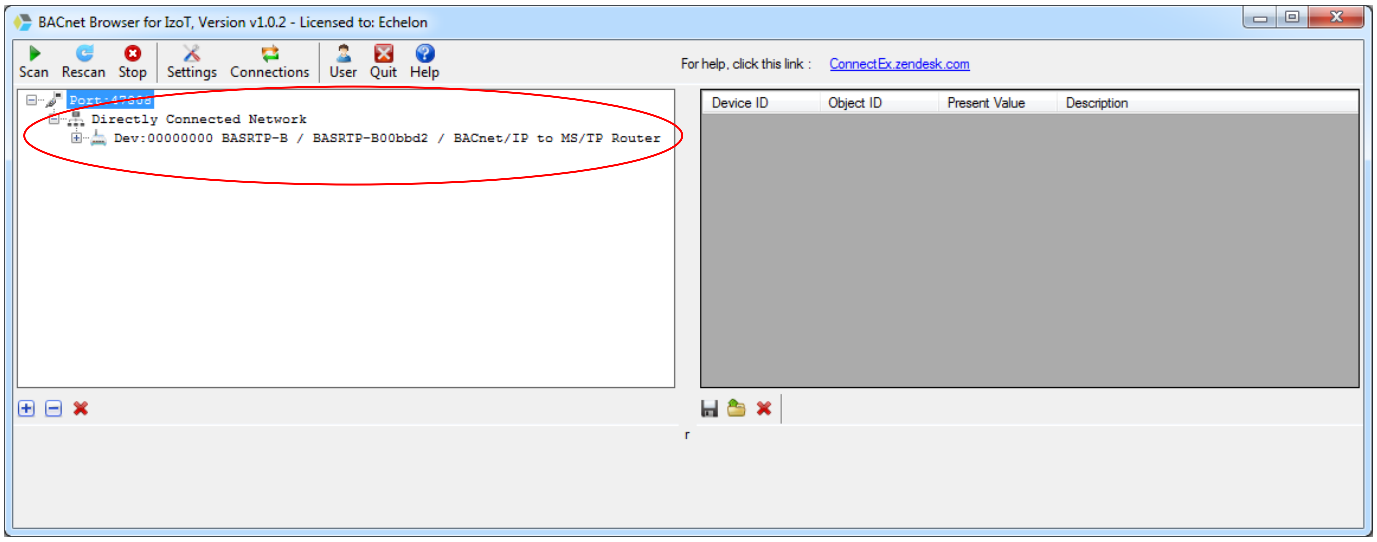
Your production solutions for a BACnet MS/TP interface for the FT 6050 Smart Transceiver require a suitable external 3.3V RS-485 transceiver to the FT 6050, with D (or DI) connected to IO10 and R (or RO) connected to IO8 and DE connected to IO9. The BACnet specification states that the RS-485 transmitter must be de-asserted within 15 bit times of the last bit of the last character (390uS at 38400 baud) which is taken care of by the Neuron BACnet MS/TP stack.

See the *FT 6000 EVK Quick Start Guide* and *FT 6000 EVB Hardware Guide* for information on using and troubleshooting your IzoT Router and EVB with BACnet.

Setting up the BACnet/IP to MS/TP Router

For BACnet MS/TP, the example project sets the EVB to MAC address 47 and the bit rate to 38400 bps by default. The BACnet router must be configured to the same bit rate, and any other free MAC address, <= 127 per BACnet specifications. Instructions on how to configure the router is supplied by the manufacturer. For example, to configure the BASrouter LX from Contemporary Controls follow these steps:

1. Connect to the router using a Web browser. The factory default IPv4 address is **192.168.92.68**. You may have to temporarily change your network adapter IPv4 address accordingly. Once connected to the router, you can change the IPv4 address to something that suits your requirements.
2. Login. The default username and password is **admin** / **admin**.
3. Confirm the BACnet/IP port is set to **47808**.
4. Set the BACnet Network Number to a unique value between 1 and 65535 (unique on your BACnet internetwork; if this is the only BACnet network, then you can use any value in the range).
5. Confirm the Baud Rate is set to 38400.
6. Confirm the MAC address is not 47.
7. Confirm Max Masters >= 47.
8. Change the IP address, IP Subnet, and IP Gateway to match your target IP subnet.
9. Save and restart.
10. If necessary, change your network adaptor IPv4 address back to its original setting.
11. Ensure that you can connect to the router at its new address using a Web browser.
12. Using the IzoT BACnet Browser, confirm that you can discover the router on the IP network as shown below:



```
Port:47808
  Directly Connected Network
    Dev:00000000 BASRTP-B / BASRTP-B00bbd2 / BACnet/IP to MS/TP Router
```

Setting up the EVB

For BACnet/IP-FT only:

Confirm the jumper settings as follows (this should be the factory default):

LCD

JP33 (LCD power) installed
JP31 (IO pins) (all installed) 1-2, 3-4 .. 15-16

For BACnet MS/TP only:

The MS/TP serial connection to the RS485 transceiver uses IO8 for receive, IO9 for transmit enable, and IO10 for transmit (normally used for USB RX, SW1, and USB RX on the EVB).

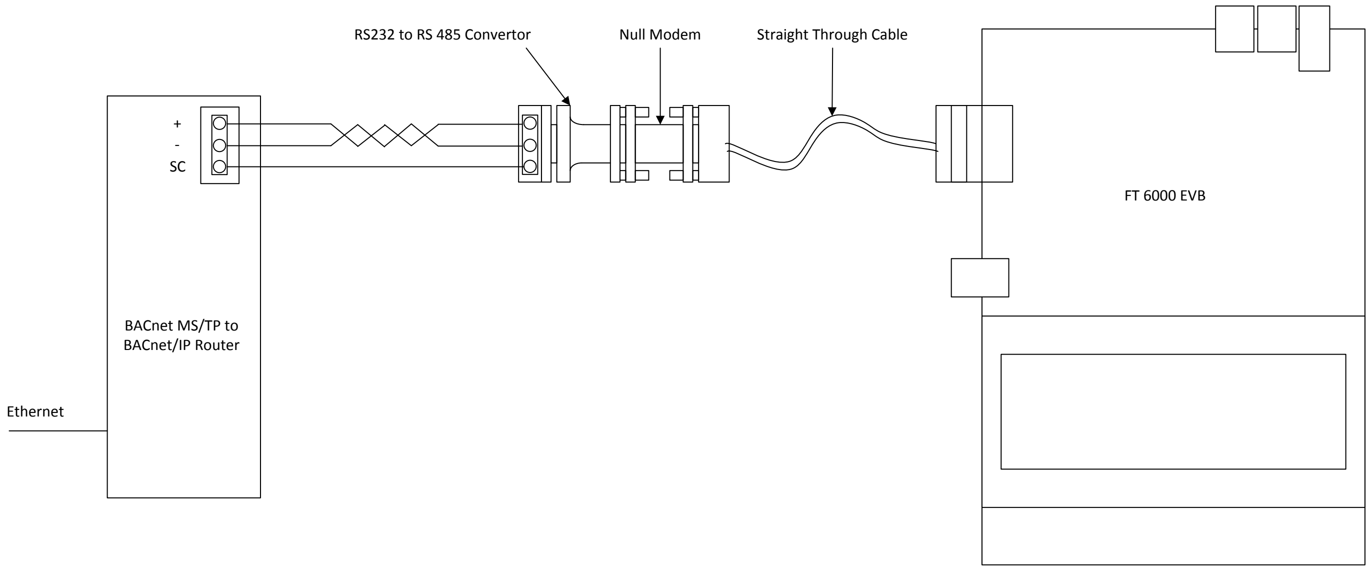
For BACnet/IP-FT and BACnet MS/TP using the EVB DB-9 Connector:

Confirm that the jumper settings are set as follows (note the additional wiring requirements):

JP201	1-2	On	IO8	Rx
	3-4	X	IO4	N/A
	5-6	X	IO1	N/A
	7-8	On	IO10	Tx
JP203	1-2	X		
	3-4	X		
	5-6	Off	**Wire JP203#5 to JP32#8**	
	7-8	X		
JP31	1-2	Off		
	3-4	Off		
	5-6	On		
	7-8	On		
	9-10	On		
	11-12	On		
	13-14	Off		
	15-16	On		
JP32	1-2	X		
	3-4	X		
	5-6	X		
	7-8	Off	**Wire JP32#8 to JP203#5**	
	9-10	X		
	11-12	X		
	13-14	X		
	15-16	X		
JP33	1-2	On		

Connect the FT 6000 EVB to a BACnet MS/TP to BACnet/IP router using a straight-through cable, null modem, and RS-232 to RS-485 converter as shown below.

RS-485 channels are polarity sensitive when used with biased networks such as BACnet MS/TP. All devices must share a common signal ground and the network cabling must be terminated at both ends. The router typically has a terminator enabled by default.



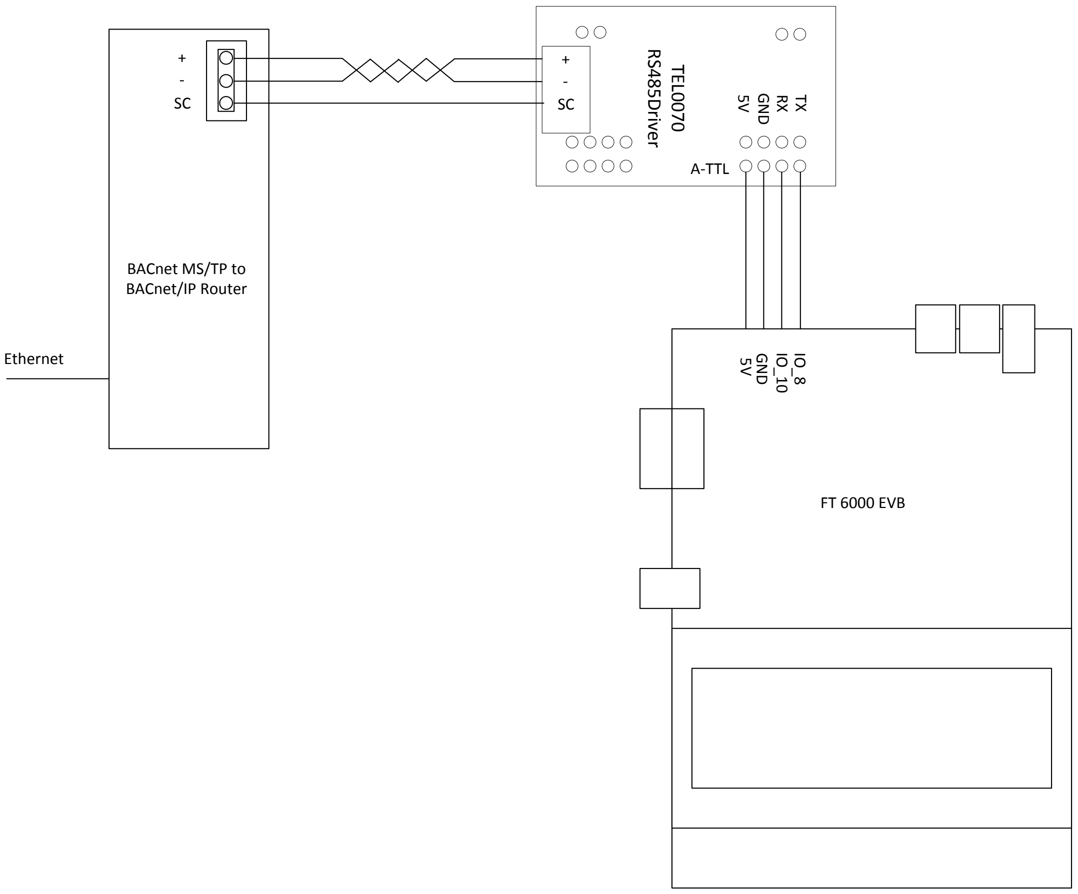
For BACnet/IP-FT and BACnet MS/TP Using the EVB I/O Pins:

Set the EVB jumpers to their factory default settings with the exception of the following:

JP32 pins 7-8   Off (IO_9, SW1)

Connect the EVB, TEL00070, and the BACnet MS/TP network as shown below.

RS-485 channels are polarity sensitive, all devices must share a common signal ground and the network cabling must be terminated at both ends. The router typically has a terminator enabled by default.



The BACnet Example Projects

You can use the example projects to build and download a simple application to an FT 6000 EVB to test your setup. There are two example projects—the *BACevb example* and the *BACsimple example*.

The BACevb example is a NodeBuilder project for the FT 6000 EVB. The application measures and displays space temperature, flashes LEDs, and interacts with the user. The example includes an IzoT CT drawing backup named **BACdemo.zip** which can be found in the LONWORKS **NeuronC\Examples\BAClon\BACevb** folder.

The BACsimple example is a minimal NodeBuilder project to show the smallest viable project using the Neuron BACnet Stack, allowing easier integration into your application.

You can build the example projects with or without BACnet/IP or BACnet MS/TP by commenting or un-commenting **#define BACNET_IP** and **#define BACNET_MSTP** in the **BACopt.h** file for each project. Set up the FT 6000 EVB as described in *Setting up the EVB* above. You must compile and download the software configuration to match the EVB jumper setup in use.

You cannot use the LCD display and light-level sensor on the FT 6000 EVB simultaneously with BACnet MS/TP. An error is generated if these two options are selected simultaneously.

Using the BACevb Project

Setting up a LonTalk/IP Interface

If you have not already created a LonTalk/IP interface for your development computer, follow these steps to create one:

1. Start the **IzoT Network Services LonTalk/IP Interfaces** application from the Start menu under **All Programs → Echelon IzoT Network Services Utilities**.
2. Type **LonTalk/IP** as the name in **LonTalk/IP Interfaces**.
3. Select the appropriate IP network interface for this LonTalk/IP interface to use from the **IP Interface** box: either **Local Area Connection** or **Wireless Network Connection**.
4. Click **Create**.
5. Close the **IzoT Network Services LonTalk/IP Interfaces** application.

Adding Routes for the IzoT Router

6. To allow your development computer to reach the LON FT channel IPv4 subnet, and to allow IzoT CT to complete actions such as Manage → Test on the IzoT Router, add two routes to your computer configuration; you can do this from a Windows command prompt launched with administrator permissions (right click and select **Run as Administrator**).

7. Enter the following commands at the Windows command prompt:

```
route –p add 192.168.11.19.1 mask 255.255.255.255 <IzoT Router LAN host IP address>
route –p add 192.168.8.0 mask 255.255.255.0 <IzoT Router LAN host IP address>
```

You can find the LAN host IP address of your IzoT Router from the relevant DHCP server or using the technique described in the *IzoT FT 6000 EVK Quick Start*. You can also use a smart phone to find your IzoT Router LAN host IP address as described in the *Getting Started→IzoT Router* book at *www.echelon.com/docs/izot*.

Restore and Open the CT Backup Drawing and Database

8. Start the IzoT Commissioning Tool from the Start menu under **All Programs→Echelon OpenLNS CT**, click **Restore** in the Design Manager and restore the drawing from the **BACdemo.zip** backup file, following the instructions when prompted. When the restore has completed successfully, you are prompted to open the drawing, click **Yes**,
9. At the Sever Location dialog, ensure the Server Location is set to **Local**, then click **Next**.
10. At the Network Interface dialog, check **Network Attached** and select **LonTalk/IP** as the Network Interface Name, then click **Next**.
11. At the Logon dialog, click **Next**.
12. At the Onnet/Offnet dialog, ensure **Onnet** is selected, and then click **Next**.
13. At the Plug-in Registration dialog, click **Finish**.
14. Click **No** when prompted to recommission devices. Click **OK** to the subsequent dialog.

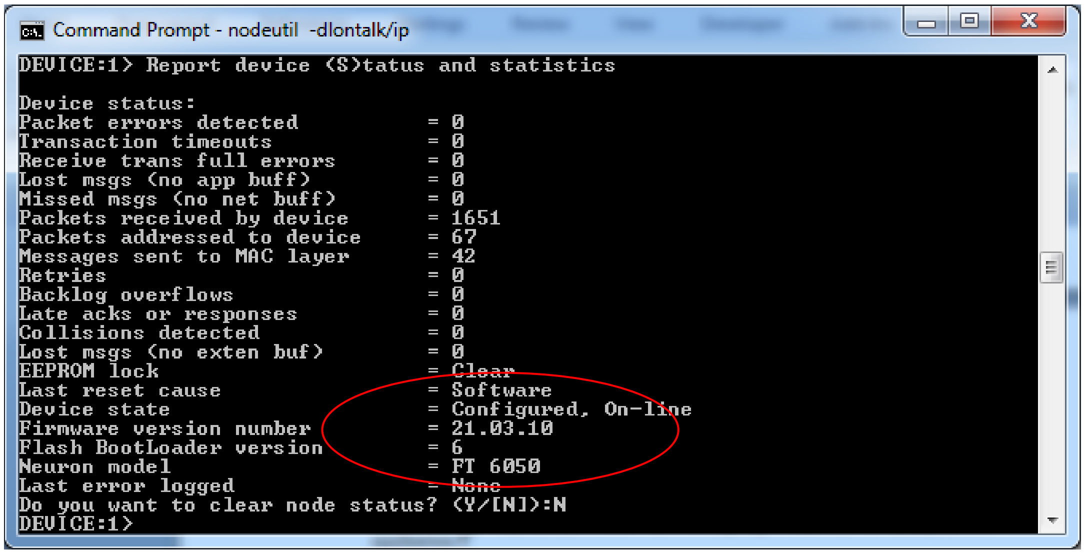
Commission the IzoT Router

15. Commission the IzoT Router by right clicking the router shape, selecting **Commission → Commission**, click **Next** at the **Router Application State** dialogue, click **Finish** at the **New Device Wizard** dialogue and when promoted to press the service pin at the next dialog, press the **Service/Connect** button on the IzoT Router. After the commissioning process completes, confirm that the router shape is now green. The router type is set to repeater to allow NodeUtil to function correctly.

Verify the EVB Flash Bootloader and Firmware Version Numbers

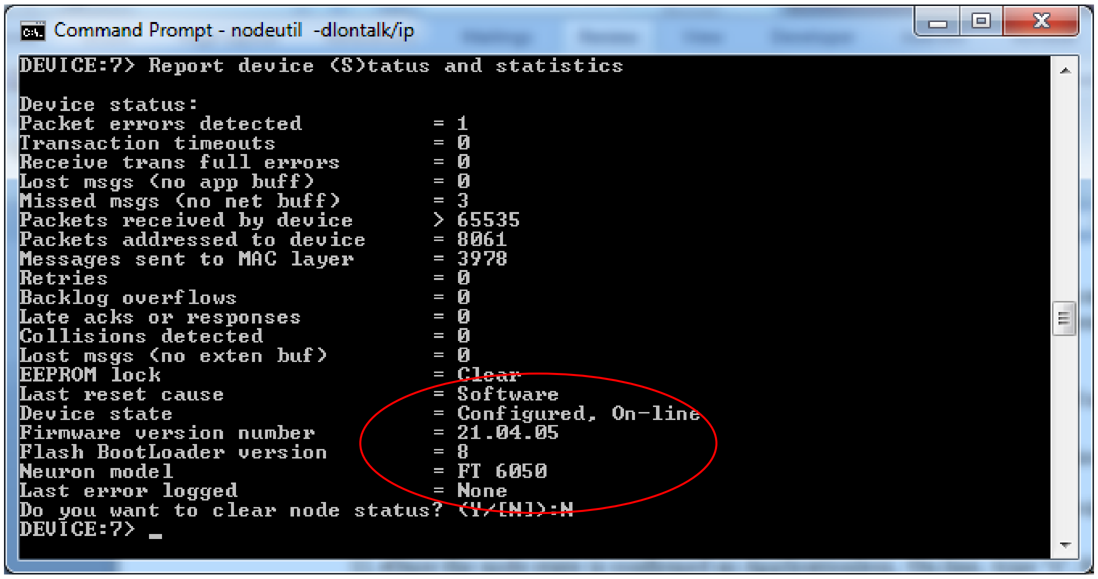
The EVB flash bootloader must be version 8 or greater and the EVB Firmware revision number must be 21.04.05 or greater to support MS/TP. Updated versions of the flash bootloader and firmware are installed with the required version of NodeBuilder as stated in *Software Requirements* above. You can determine the installed versions in the FT 6000 EVB as follows:

16. Open a Windows command prompt.
17. Start NodeUtil by typing: **nodeutil – dlontalk/ip**.
18. With the EVB powered and connected as shown in the *Hardware Requirements* above, press the Service button on the EVB. This will register the EVB as device 1.
19. Go to the device by typing: **g 1** and then pressing **Enter**.
20. Type **s** to get the EVB Status and then **n** when prompted to clear the device status. The following is displayed (although the flash bootloader version and firmware version number may vary):



```
DEVICE:1> Report device (S)tatus and statistics

Device status:
Packet errors detected          = 0
Transaction timeouts            = 0
Receive trans full errors       = 0
Lost msgs (no app buff)         = 0
Missed msgs (no net buff)       = 0
Packets received by device      = 1651
Packets addressed to device     = 67
Messages sent to MAC layer      = 42
Retries                         = 0
Backlog overflows               = 0
Late acks or responses          = 0
Collisions detected             = 0
Lost msgs (no exten buf)        = 0
EEPROM lock                     = Clear
Last reset cause                = Software
Device state                    = Configured, On-line
Firmware version number         = 21.03.10
Flash BootLoader version        = 6
Neuron model                    = FT 6050
Last error logged               = None
Do you want to clear node status? (Y/[N]):N
DEVICE:1>
```

21. If the flash bootloader version is less than 8 or the firmware version number is less than 21.04.04 go to step 22 to update them. Otherwise go to step 27.
22. To update the flash bootloader type **y**.
23. Type *LonWorks***\Images\Ver21\bla6000.ndl**, inserting your LONWORKS path for *LonWorks*, and then press **Enter**. For new installations on 64-bit Windows, the LONWORKS path is **C:\Program Files (x86)\LonWorks**.
24. To update the Firmware, once the device state is confirmed as Applicationless, On-line, type **y**.
25. Type *LonWorks***\Images\Ver21\b6050v4.ndl**, and then press **Enter**.
26. Once the device state is confirmed as Applicationless, On-line, type **s**, followed by **n** when prompted to clear the device status. The following is displayed, confirming the new flash bootloader version and firmware version number (although the versions may be later than illustrated).



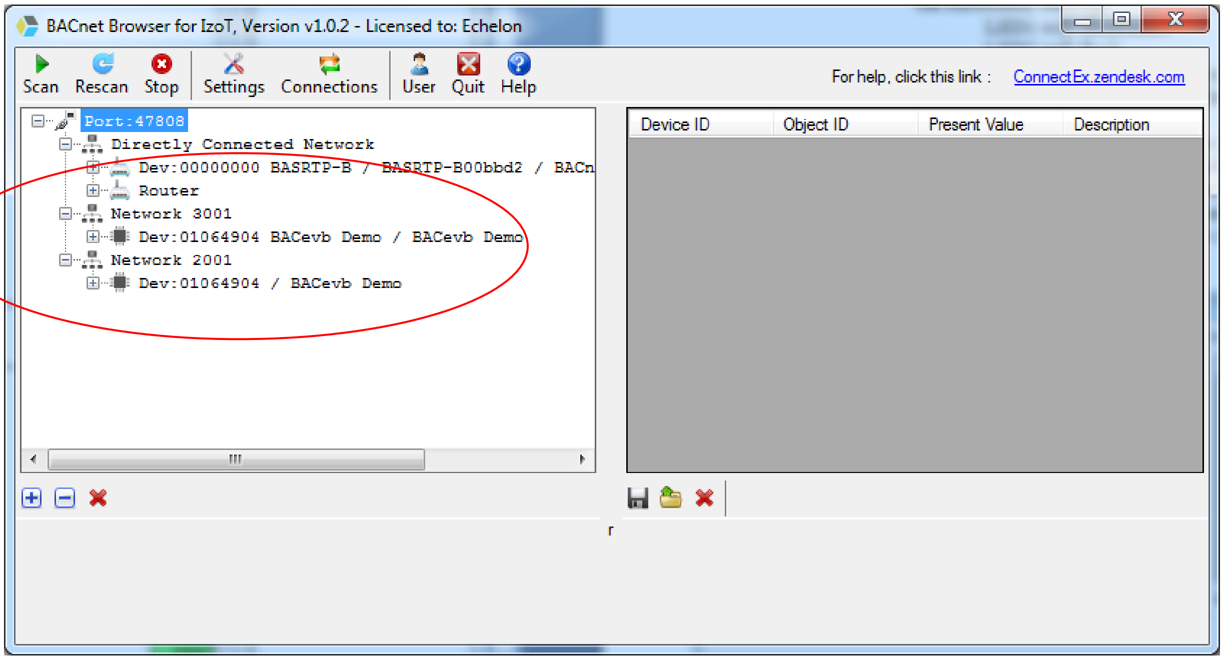
```
DEVICE:?> Report device (S)tatus and statistics

Device status:
Packet errors detected          = 1
Transaction timeouts            = 0
Receive trans full errors       = 0
Lost msgs (no app buff)         = 0
Missed msgs (no net buff)       = 3
Packets received by device      > 65535
Packets addressed to device     = 8061
Messages sent to MAC layer      = 3978
Retries                         = 0
Backlog overflows               = 0
Late acks or responses          = 0
Collisions detected             = 0
Lost msgs (no exten buf)        = 0
EEPROM lock                     = Clear
Last reset cause                = Software
Device state                    = Configured, On-line
Firmware version number         = 21.04.05
Flash BootLoader version        = 8
Neuron model                    = FT 6050
Last error logged               = None
Do you want to clear node status? (Y/[N]):N
DEVICE:?> _
```

27. Close NodeUtil by typing **ee**.

Build, Load and Commission the EVB

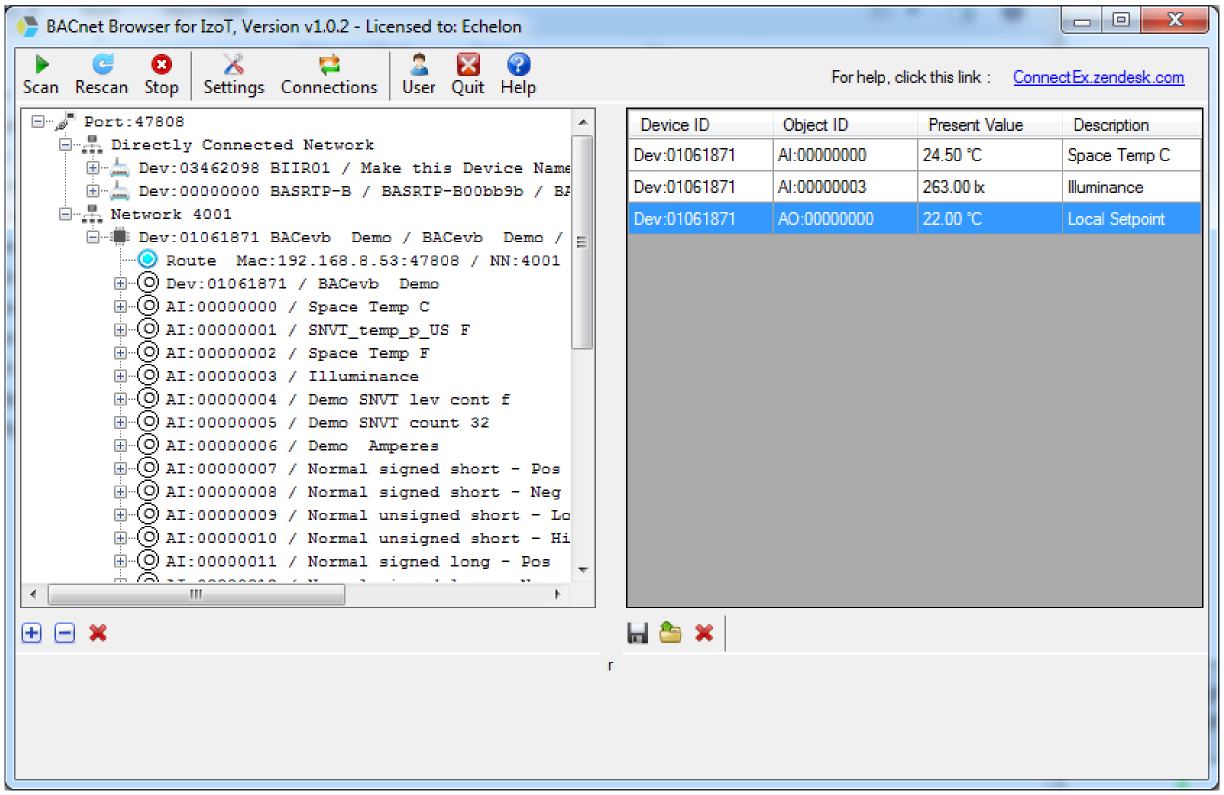
27. From within IzoT CT**,** right click the **FT 6000 EVB** device, select **NodeBuilder→Edit Source**.
28. At the NodeBuilder Project dialogue, check **Open an existing NodeBuilder project.**
29. Click **Next.**
30. At the Select NodeBuilder Project file dialog, click the ellipsis button (…) and traverse to *LonWorks***NeuronC\Examples\BAClon\BACevb** and select **BACdemo.NbPrj**, and then click **Open**.
31. Ensure that **Set as Default Project for this Network** is checked. Click **Finish**.
32. From within **NodeBuilder**, expand **Device Templates** in the Workspace pane.
33. Expand the **BACdemo** device template.
34. Expand **Source Files,** right click **Source Files,** click **Insert,** double click the **BACdemo** folder, type **BACopt.h** in **File Name** ,and then click **Open**.
35. Double click the **BACopt.h** file under **Source Files** and adjust **#define BACNET_IP** and **#define BACNET_MSTP** as required. Comment out **#define USE_EVB_LCD** if you are using using **BACNET_MSTP**.
36. Right click the **BACdemo** template and then click **Clean.**
37. Right click the **BACdemo** template again and click **Build.**
38. Ensure that the build succeeded.
39. From within IzoT CT, right click the **FT 6000 EVB** device and select **Commission→Commission**, ensure **Load Application Image** is checked at the **Application Image** dialogue**,** click **Next**, click **Next** at the **Application State** dialog, click **Finish** at the **New Device Wizard** dialog and when promoted at the next dialogue, press the Service button on the FT 6000 EVB. Confirm that the FT 6000 EVB shape is green.
40. Confirm all is working as expected. If startup is successful, LED1 changes as follows:
    LED1 will illuminate. (If BACnet MS/TP not selected)
    LED1 will go out if SW1 is held down for 1 second. This LED will come on again each time the system restarts. It can be used to check for system crashes. (If BACnet MS/TP not selected)
41. On a development computer attached to the network, run the IzoT BACnet Browser. The browser discovers the BACnet Device on the network, and lists the BACnet Objects contained in it.

42. If BACnet/IP and BACnet MS/TP have both been enabled in **BACopt.h**, then you should be able to simultaneously see the same device via the BACnet/IP and BACnet MS/TP networks as shown below. This is not necessarily a typical operational situation, but is a valid indication of correct functionality during development. If MS/TP is not enabled, then you will only see the BACnet/IP device.


BACnet Browser for IzoT, Version v1.0.2 - Licensed to: Echelon
Scan
Rescan
Stop
Settings
Connections
User
Quit
Help
For help, click this link : ConnectEx.zendesk.com
Device ID
Object ID
Present Value
Description

```
Port:47808
  Directly Connected Network
    Dev:00000000 BASRTP-B / BASRTP-B00bbd2 / BACn
    Router
  Network 3001
    Dev:01064904 BACevb Demo / BACevb Demo
  Network 2001
    Dev:01064904 / BACevb Demo
```

43. The FT 6050's device object identifier and object name will be automatically set based on the IP address of the FT 6050. You can change the device object identifier and object name using a standard BACnet workstation, including the IzoT BACnet Browser.
44. From either the BACnet/IP or BACnet MS/TP device, add **Space Temp C**, **Illuminance**, and **Local Setpoint** to the grid view by right-clicking each in turn and selecting **Add to Grid View** as shown below. Verify that the values are realistic and that you can write the Local Setpoint by right clicking the entry in the grid view, selecting **Change Value**, writing the new value to **Priority 1**, and then clicking **Set**. The Illuminance value will not show valid data if BACnet MS/TP selected.


BACnet Browser for IzoT, Version v1.0.2 - Licensed to: Echelon
Scan
Rescan
Stop
Settings
Connections
User
Quit
Help
For help, click this link : ConnectEx.zendesk.com

```
Port:47808
  Directly Connected Network
    Dev:03462098 BIIR01 / Make this Device Name
    Dev:00000000 BASRTP-B / BASRTP-B00bb9b / BA
  Network 4001
    Dev:01061871 BACevb  Demo / BACevb  Demo /
      Route  Mac:192.168.8.53:47808 / NN:4001
      Dev:01061871 / BACevb  Demo
      AI:00000000 / Space Temp C
      AI:00000001 / SNVT_temp_p_US F
      AI:00000002 / Space Temp F
      AI:00000003 / Illuminance
      AI:00000004 / Demo SNVT lev cont f
      AI:00000005 / Demo SNVT count 32
      AI:00000006 / Demo  Amperes
      AI:00000007 / Normal signed short - Pos
      AI:00000008 / Normal signed short - Neg
      AI:00000009 / Normal unsigned short - Lo
      AI:00000010 / Normal unsigned short - Hi
      AI:00000011 / Normal signed long - Pos
```

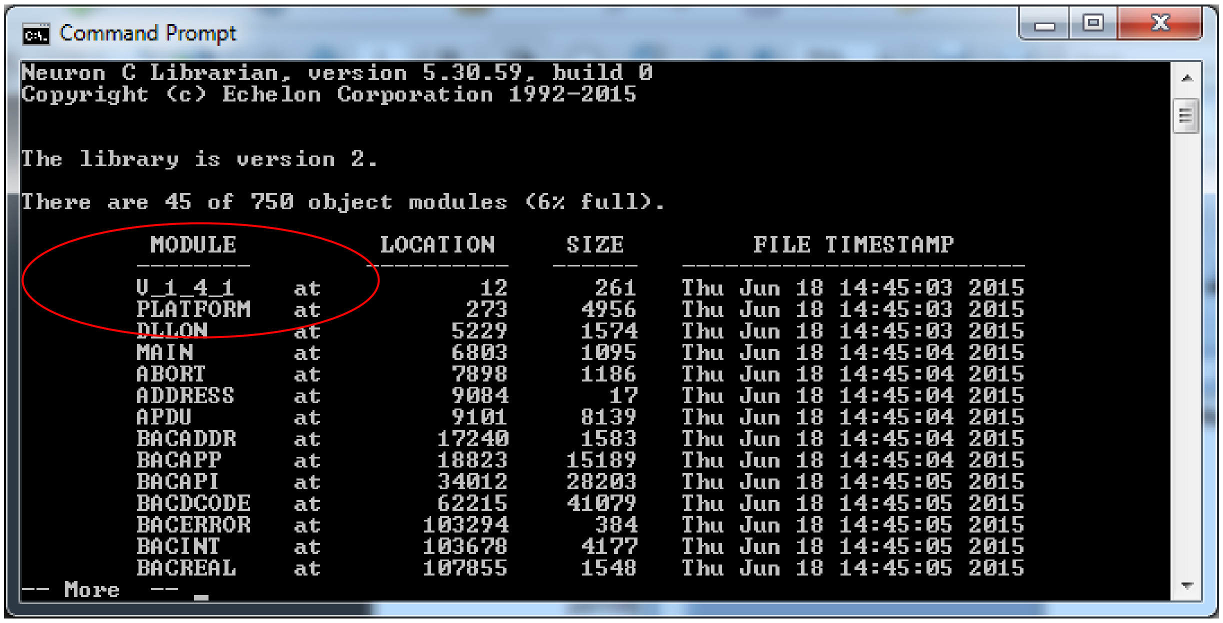
Device ID	Object ID	Present Value	Description
Dev:01061871	AI:00000000	24.50 °C	Space Temp C
Dev:01061871	AI:00000003	263.00 lx	Illuminance
Dev:01061871	AO:00000000	22.00 °C	Local Setpoint

Determining the BACnet Stack Version Number

To determine the version number of the Neuron BACnet Stack, enter the following command in a Windows command prompt after changing to the LONWORKS **Images\Ver21** folder:

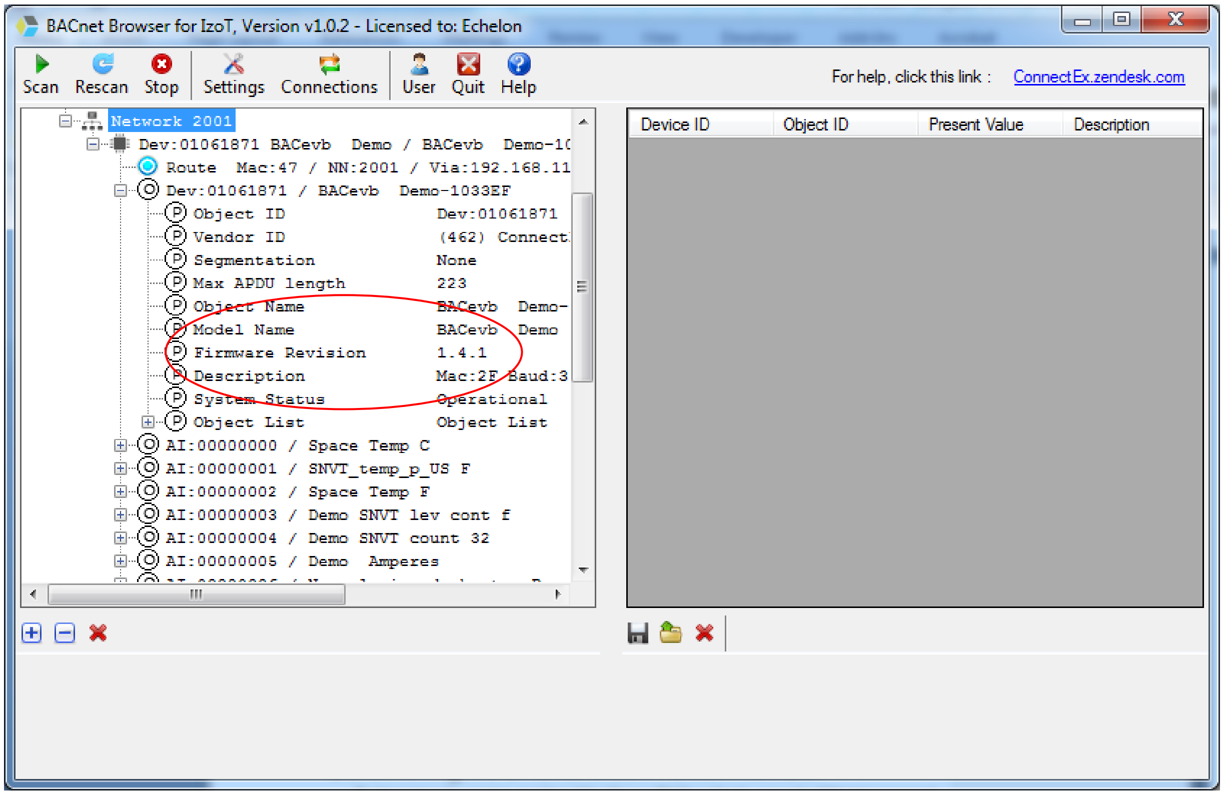
```
nlib -r <libraryname> | more
```

For example **nlib –r baclon.lib | more** may show the following, where the first module is the version number.



```
Neuron C Librarian, version 5.30.59, build 0
Copyright (c) Echelon Corporation 1992-2015

The library is version 2.

There are 45 of 750 object modules (6% full).

              MODULE           LOCATION      SIZE          FILE TIMESTAMP
             --------         ----------    ------    --------------------------
             V_1_4_1     at           12       261    Thu Jun 18 14:45:03 2015
             PLATFORM    at          273      4956    Thu Jun 18 14:45:03 2015
             DLLON       at         5229      1574    Thu Jun 18 14:45:03 2015
             MAIN        at         6803      1095    Thu Jun 18 14:45:04 2015
             ABORT       at         7898      1186    Thu Jun 18 14:45:04 2015
             ADDRESS     at         9084        17    Thu Jun 18 14:45:04 2015
             APDU        at         9101      8139    Thu Jun 18 14:45:04 2015
             BACADDR     at        17240      1583    Thu Jun 18 14:45:04 2015
             BACAPP      at        18823     15189    Thu Jun 18 14:45:04 2015
             BACAPI      at        34012     28203    Thu Jun 18 14:45:05 2015
             BACDCODE    at        62215     41079    Thu Jun 18 14:45:05 2015
             BACERROR    at       103294       384    Thu Jun 18 14:45:05 2015
             BACINT      at       103678      4177    Thu Jun 18 14:45:05 2015
             BACREAL     at       107855      1548    Thu Jun 18 14:45:05 2015
-- More  -- _
```

To see the version during runtime, using a BACnet browser, go to the device, then the device object, and the version number is listed as a property of the device object as shown below. The application version is under the device application developer's control.



```
Network 2001
  Dev:01061871 BACevb  Demo / BACevb  Demo-1(
    Route  Mac:47 / NN:2001 / Via:192.168.11
    Dev:01061871 / BACevb  Demo-1033EF
      Object ID              Dev:01061871
      Vendor ID              (462) Connect
      Segmentation           None
      Max APDU length        223
      Object Name            BACevb  Demo-
      Model Name             BACevb  Demo
      Firmware Revision      1.4.1
      Description            Mac:2F Baud:3
      System Status          Operational
      Object List            Object List
    AI:00000000 / Space Temp C
    AI:00000001 / SNVT_temp_p_US F
    AI:00000002 / Space Temp F
    AI:00000003 / Demo SNVT lev cont f
    AI:00000004 / Demo SNVT count 32
    AI:00000005 / Demo  Amperes
```

2

Using BACnet

This chapter explains BACnet terminology and describes the Neuron BACnet Server (NBS).

BACnet Terminology

The following terms are a brief summary of BACnet terminology.

BACnet Network – A group of BACnet devices on a single network, which may be any of BACnet's physical layer options, namely IP media such as Ethernet or LON FT, and other media such as RS-485 and LonTalk, identified by a *network number*.

BACnet Internetwork – A collection of connected BACnet networks connected via BACnet routers. The resulting devices are all able to communicate with one another. Every BACnet device is required to have a unique BACnet *device instance* and *device object* name to unambiguously distinguish it, and every BACnet network must have a unique network number.

BACnet Device – A controller, operator workstation, or other device, that supports BACnet communications

BACnet Object – A data point, measurement or some other value in a BACnet device

BACnet Object Identifier – An identifying parameter for each object in a BACnet device which is unique on a device basis, and which consists of the *object type* and the object's *instance number*.

BACnet Property – A BACnet object contains multiple *properties*, some optional and some mandatory. For example, properties such as the present value can reflect the value of a physical input such as temperature. The object identifier property of an object uniquely identifies the object.

BACnet Device Instance – The instance number of the object identifier of the Device object that every BACnet device is required to contain. It has to be unique across the whole of the BACnet internetwork.

BACnet Router – A standard BACnet component that allows the interconnection of different BACnet networks. They may effect a physical and logical change between networks of different physical types and different network numbers, or sometimes only a logical change between different BACnet network numbers.

Priority Array – An array of 16 values for some BACnet output objects, (e.g. Analog Output), which allows multiple systems of different priorities to control a single output, by writing to the priority array, with predetermined results. The value of the resulting highest priority array item is transferred to the present value.

Present Value – One of a BACnet object's properties, usually containing a physical input or output value.

Relinquish Default – A value that is transferred to the present value when the priority array does not contain any values at all.

BACnet Priority – Specified when writing a value to the BACnet priority array.

Using the Neuron BACnet Server

You can use the Neuron BACnet Stack to implement a BACnet server on the Series 6000 Neuron core. The Neuron BACnet Server is a BACnet compliant server, and other BACnet devices interact with the Neuron BACnet Server as they would with any other BACnet device.

The Neuron BACnet Server provides a network interface to the BACnet input and output objects implemented by the device application. The description of an input vs. an output is typically different for BACnet and LON devices. For BACnet devices, an input object provides BACnet client access to a datapoint in the device, and an output object provides BACnet client control of a datapoint in the device. For LON devices, the description of a network variable is typically relative to the device itself, so an input network variable can be updated by another device, whereas an output network variable is sent to other devices. For example, for a temperature sensor using BACnet, the temperature sensor is viewed as an Analog Input and displayed and processed accordingly. In a LON example, the common approach is to process the measurement internally and expose the resulting temperature as an analog output network variable of a functional block. The functional block typically has a profile name that identifies the type of value, for example the functional block may be Open Loop Sensor functional block. Similarly, in the case of a setpoint for example, in BACnet systems, setpoints are normally considered Analog Outputs to be written to a device, whereas when using LON devices, setpoints are processed as analog input network variables or configuration properties of functional blocks where the profile identifies the function of the functional block such as a Space Comfort Controller VAV. Another example of LON functional block naming is the set of profiles defined by the LONWORKS IoT resource file set which use BACnet naming conventions.

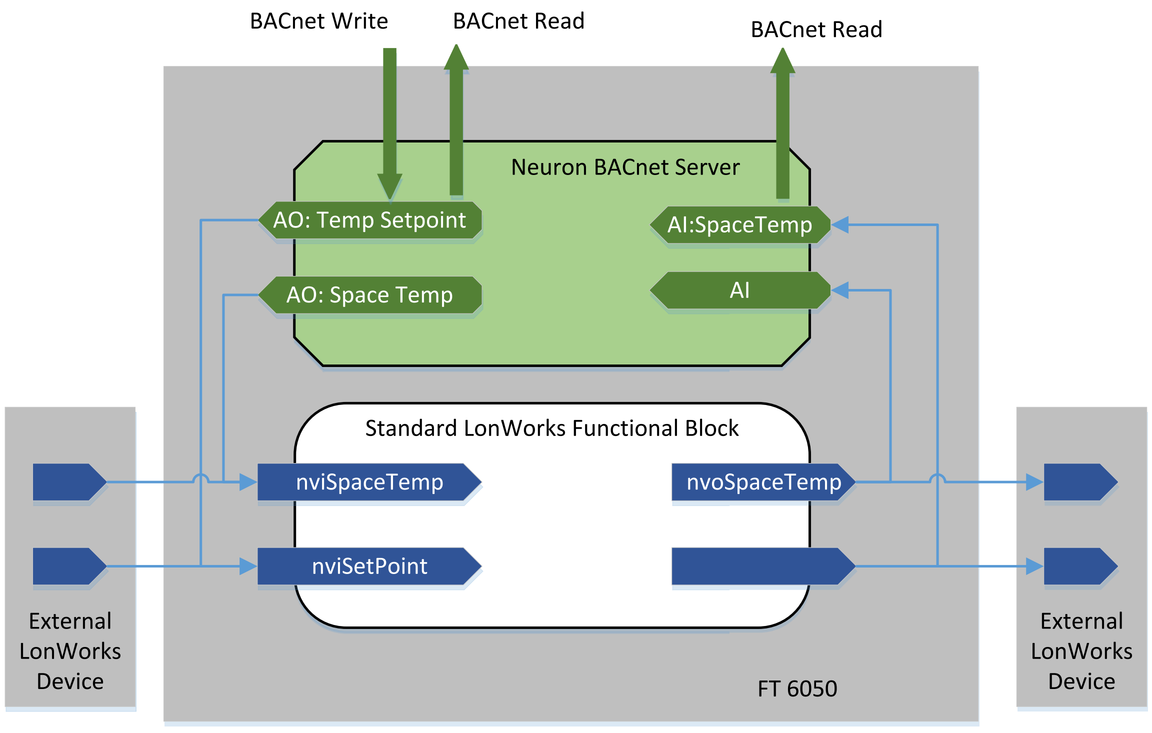


Figure 1. Conceptual Model of a Virtual BACnet Server

Figure 1 shows three LON devices. The central device contains the BACnet interface, embodied as the Neuron BACnet Server (NBS) which exists within the Neuron core. A BACnet client (such as a building management system operator workstation, or another BACnet controller) communicates with the Neuron BACnet Server. The BACnet client can read from and write to BACnet points in a completely BACnet compliant fashion. The Neuron BACnet Server connects and maps these BACnet points internally to the LON network variables.

Typical LON connections can still be bound, simultaneously, to the LON network variables in the LON functional blocks as before, and the LON systems will continue normal LON operation.

You can map a single LON network variable with multiple fields to multiple BACnet objects. For example, in Figure 1 there is a network variable named **nviSpaceTemp** that is normally connected to a suitably typed output network variable from another device. If a BACnet client is required to supply this parameter, then the BACnet client will execute a write to the **AO:TempSetpoint** Analog Output in the Neuron BACnet Server. Internally the Neuron BACnet Stack will map this write to **nviSpaceTemp**.

Similarly, a BACnet client can get the value of a LON output by reading the BACnet point mapped to that network variable output.

BACnet Read Operations

When a BACnet client polls for data, it can request any BACnet property contained within the BACnet object. Most of these properties are rather static (e.g. object identifier, object name) and are seldom polled, sometimes only once, and these types of reads are handled completely within the Neuron BACnet Server.

Reads for live data normally are a BACnet read for the present value property of the BACnet object, e.g. an Analog Input, instance 1. This results in the Neuron BACnet Server interface accessing the appropriate live data in the LON network variable associated with the BACnet object.

Write Operation Resolution

BACnet write operations are more complex within the BACnet protocol than the LonTalk protocol. The BACnet protocol specifies how multiple clients can write to the present value of a BACnet object at different priorities, and specifies a default value called the *relinquish default* to be used when there is no active value. The Neuron BACnet Server leverages this capability to support multiple BACnet and LON clients. This is illustrated in the following diagram.

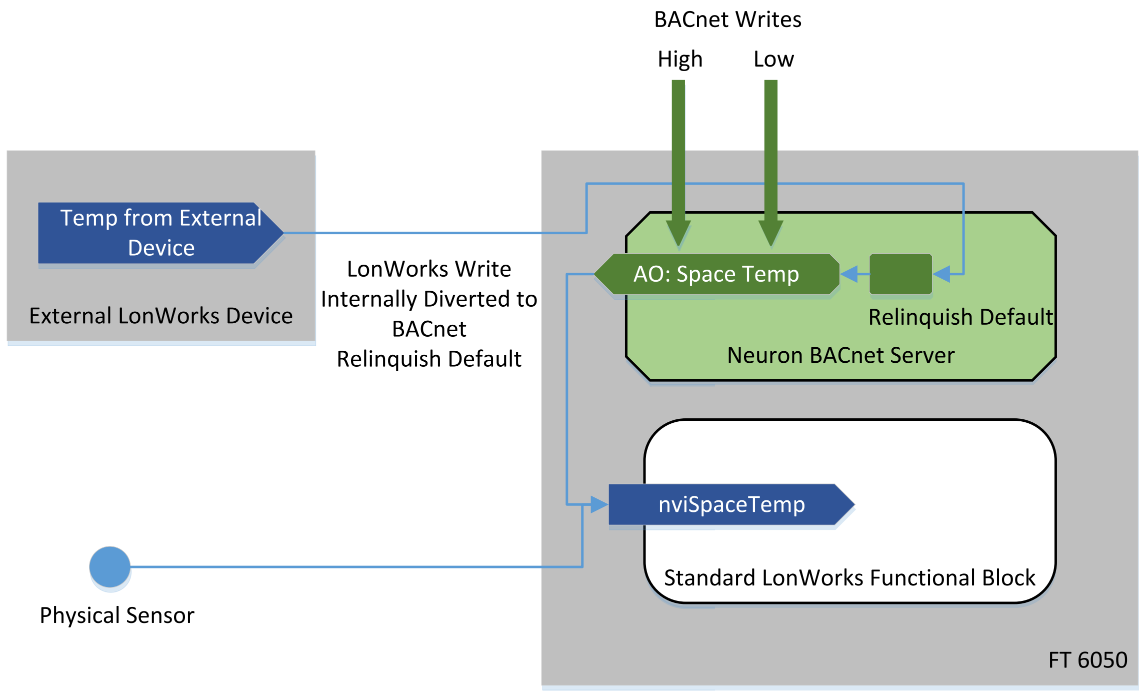


Figure 2. Write Operation Resolution

For example, in the LONMARK VAV functional profile, the **nviSpaceTemp** input can receive data either from a physical sensor, or from another LON device. The rules for choosing which value to take are laid out by the profile and they say that if there is a valid value from another LON device, this will override the physical input. If the validity of the value expires for any reason, usually due to the failure to refresh the value in a timely basis, the functional block reverts to using the physical input.

The BACnet interface extends this model. If there is valid BACnet data available, then this overrides both the physical input as well as the input from the other LON device.

BACnet has a mechanism for *commandable objects*, such as Analog Outputs, that makes this process seamless. The mechanism is based on a *priority array* property and a *relinquish default* property.

The Neuron BACnet Server intercepts all LonTalk/IP writes and diverts these to the relinquish default property. If no other value in the BACnet priority array has been written, this value flows through to the NV in the functional block.

If a BACnet write occurs at a given priority, then this value is forwarded to the NV. However a write to a lower priority is blocked. A write of a BACnet *null* value clears the position in the priority array allowing lower priorities to flow through.

If all the BACnet writes are relinquished (A null value is written at all the appropriate BACnet priority array value), then the relinquish default value, and hence the network variable update from the external LON device, is used again. This means, after a predetermined time with no further writes by another BACnet device, the priority array will clear and the system will revert to using the previous source.

BACnet statuses such as **Out Of Service**, **Override**, and **Fault** are treated as follows:

Out of Service	This is a BACnet-only concept. It allows a BACnet point to be disconnected from the live data, in this case the LON network variable, for testing and diagnostic purposes. It is fully functional within the Neuron BACnet Server but does not impact the functioning of the LON interface.
Override	Override is a flag that indicates that the BACnet value being reported is no longer a true reflection of the physical value. It is an optional BACnet property and is not supported by the Neuron BACnet Server.
Fault	A fault indicates some sort of problem with that value or measurement. This condition can be detected by the Neuron BACnet Server and is passed on to the BACnet client to indicate the condition in a logical manner.

BACnet allows the present value property to be read back from a BACnet output. A matching BACnet input is not required for reading back the true value of any BACnet output. BACnet outputs in the Neuron BACnet Server allow the BACnet client to observe the network variable values set by the physical sensors or other LON devices at all times, without having to write anything, and without having to create a shadow BACnet input specifically for this purpose.

BACnet Interface Overview

The following figure depicts the BACnet interface.

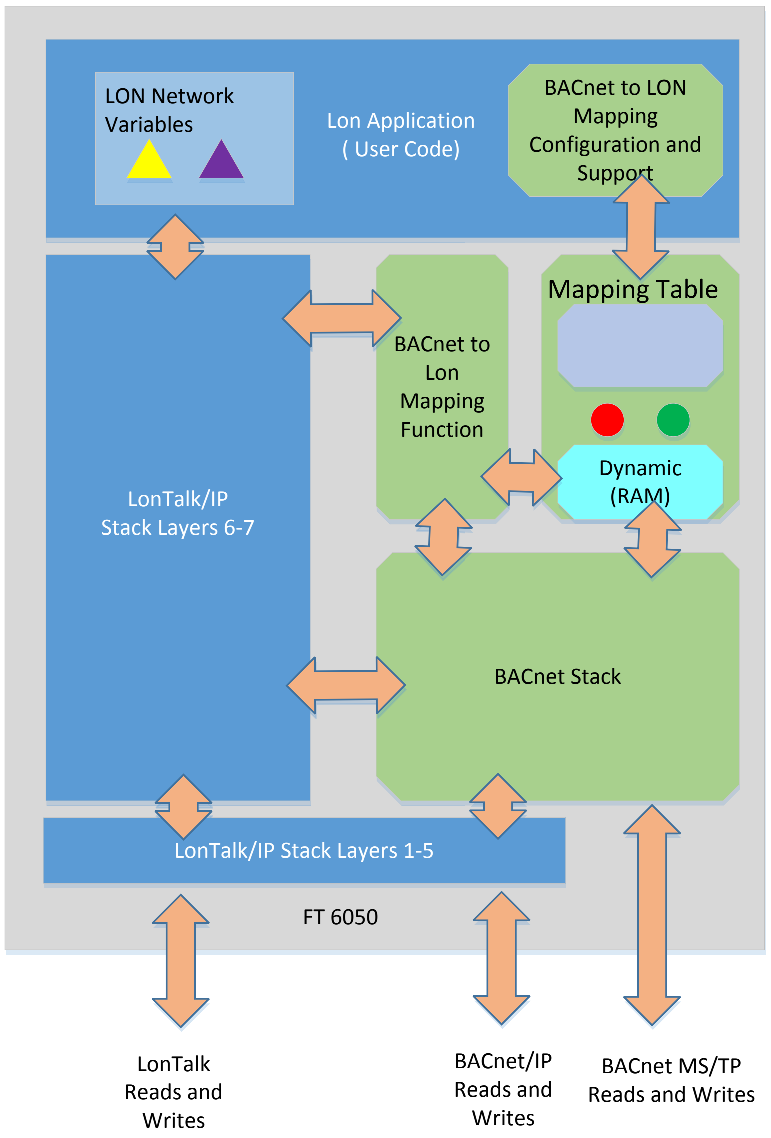

LON Network Variables
Lon Application (User Code)
BACnet to LON Mapping Configuration and Support
Mapping Table
BACnet to Lon Mapping Function
Dynamic (RAM)
LonTalk/IP Stack Layers 6-7
BACnet Stack
LonTalk/IP Stack Layers 1-5
FT 6050
LonTalk Reads and Writes
BACnet/IP Reads and Writes
BACnet MS/TP Reads and Writes

Figure 3. Functional Overview

In this figure, the items shown in blue are the Neuron LonTalk/IP firmware functions and data structures, and the ones in green are the BACnet related functions and data structures.

At the bottom of the diagram, arrows represent both LonTalk/IP and BACnet/IP read and write messages. These can be transferred over FT and are not exclusive, so normal LON operation can occur at the same time as BACnet activity.

BACnet messages are identified by LonTalk/IP message codes, and are routed through the Neuron BACnet Stack where they are interpreted, and BACnet-only operations may access the mapping table and respond to the BACnet client without any further impact to the LonTalk/IP side of the system.

BACnet messages that do affect the network variables are routed via the mapping functions, and then back to the LonTalk/IP Stack, where they are presented to the Application layer completely transparently to the application code.

There is a configuration requirement to map the desired BACnet objects to the LON functional blocks. Initially this is coded by the application developer, or via IzoT Net plug-ins.

Data Flow during a LON Write to a Network Variable

The following figure shows data flow when a LON device writes to a network variable.

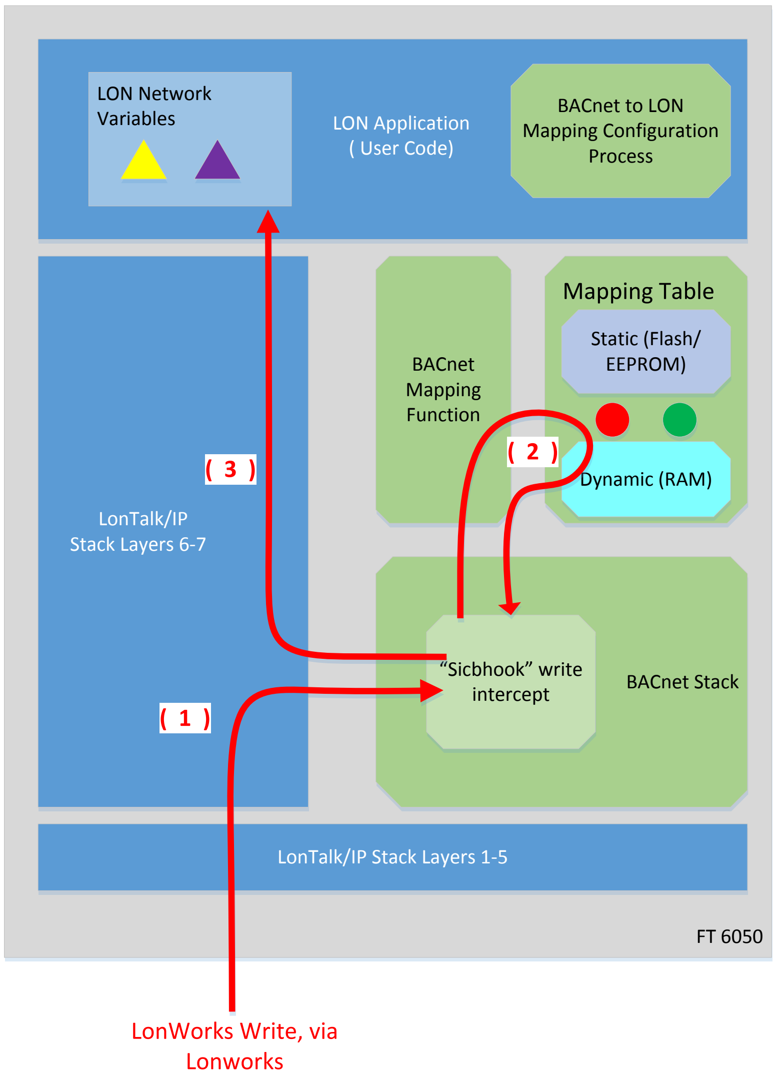

LON Network Variables
LON Application (User Code)
BACnet to LON Mapping Configuration Process
BACnet Mapping Function
Mapping Table
Static (Flash/ EEPROM)
Dynamic (RAM)
(3)
(2)
(1)
LonTalk/IP Stack Layers 6-7
"Sicbhook" write intercept
BACnet Stack
LonTalk/IP Stack Layers 1-5
FT 6050
LonWorks Write, via Lonworks

Figure 4. Data Flow during LON Write

A LON network variable write arrives at the LonTalk/IP Stack Layer 6 in the figure; this is noted as (1). During the processing of the write, the Neuron BACnet Stack is given an opportunity to examine the message, and if necessary, (2) modify the data content with higher priority data values (see *Write Operation Resolution* above).

The modified or unmodified message is passed back to the LonTalk/IP stack for further processing as normal at this point (3).

Data Flow During a LON Read of a Network Variable

A LON network variable read is handled by the LonTalk/IP Stack. The Neuron BACnet Stack does not participate in this task.

Data Flow During an Outgoing Network Variable Update

When the application program updates a network variable, the LonTalk/IP Stack generates an outgoing LonTalk/IP NV update message and also forwards the message to the Neuron BACnet Stack for processing. If a BACnet COV (change of value) subscription has been configured, a BACnet COV message is generated and sent, in addition to the LonTalk/IP NV update. If no COV subscription has been configured, then no further BACnet action occurs.

Data Flow During a BACnet Write to a BACnet Object

A BACnet Write operation is received as a LonTalk/IP foreign frame message with a message code specifying the payload is a BACnet/IP message. As shown in Figure 4, when the message is received by the LonTalk/IP Stack (1) it is routed through the Neuron BACnet Stack which extracts the BACnet properties and values, packages them as a LonTalk/IP network variable update, and passes the network variable update back to the LonTalk/IP Stack for further processing.

Data Types and BACnet to LON Mapping

The following BACnet objects are implemented in the Neuron BACnet Stack.

AI	**Analog Input**
AO	**Analog Output**
BI	**Binary (Boolean) Input**
BO	**Binary (Boolean) Output**

BACnet Instance Numbering

BACnet objects are identified by an *object identifier* with an *object type* and an *object instance* value. The object instance value can be any number between 0 and 2^22-2 (4194302) inclusive. Object identifiers have to be unique per BACnet device, but the object instance value only has to be unique per Object_Type value. The Neuron BACnet Stack automatically assigns BACnet instance numbers starting from 0 as the BACnet objects are created.

Device Object Identifier and Name

The Device Object's Identifier (Object_Identifier) and Name (Object_Name) properties for a BACnet device must be unique across a BACnet internetwork.

The Neuron BACnet Stack automatically generates unique Device Object Identifier and Object Name property values based on the IP address of the Neuron BACnet device. These property values can be adjusted as required by using the IzoT BACnet Browser or any other standard BACnet workstation that supports this functionality. If modified, these values are stored in persistent memory.

Mapping BACnet Objects to LON NVs

The Neuron BACnet Stack maps LON network variables (NVs) to BACnet objects. Many NVs are structures containing multiple fields, whereas BACnet objects are effectively single point values with properties. As a result, there is often a one-to-many mapping required between BACnet and LonTalk/IP.

For example, in the LONMARK VAV profile, the **nvoUnitStatus** output is member number 4 with a **SNVT_hvac_status** type which has seven fields. To map this NV to BACnet, the seven fields are mapped to seven BACnet Analog Input objects as shown in the following table.

SNVT	Mapping (for example)
SNVT_hvac_status.mode	maps to Analog Input
SNVT_hvac_status.heat_output_primary	maps to Analog Input
SNVT_hvac_status.heat_output_secondary	maps to Analog Input
SNVT_hvac_status.cool_output	maps to Analog Input
SNVT_hvac_status.econ_output	maps to Analog Input
SNVT_hvac_status.fan_output	maps to Analog Input
SNVT_hvac_status.in_alarm	maps to Analog Input

The Neuron BACnet Stack automatically assigns BACnet object instance numbers to the configured BACnet Objects. The mapping can be identified by the object name you define in your application.

Examples of mapping methods are found in the **BAClon\mapping.nc** file in both the BACevb and BACsimple sample projects. See *Mapping BACnet Objects to Network Variables* below for more information.

Queue Management

If there are more incoming BACnet/IP messages than the Neuron BACnet Stack can process, these messages will back up in the LonTalk/IP Stack incoming queue, and eventually messages will be discarded just like any other LonTalk/IP message overrun. You can monitor incoming buffer usage using either the Manage Test command in the IzoT Commissioning Tool or the Report Device Status and Statistics command in NodeUtil.

BACnet MS/TP messages are queued internally before processing, and you can set the size of this queue during development.

Adding the Neuron BACnet Stack to an FT 6050

To add the Neuron BACnet Stack to an application for the FT 6050, create a new Neuron C project as described in the *IzoT NodeBuider User's Guide*. After you have set up the project, copy the **BAClon** subdirectory from one of the sample projects to the new project directory.

Modify the project main source file, similar to what is shown in the source code of the sample projects. Add the following **#include** statements and source code modifications to the project.

1. **#include** statements

   ```
   #include "baclon\blonsys.nc"              // Required for BACnet
   #include "BAClon\mapping.nc"              // Must be included
   // after all_network
   // variables have been declared
   ```

2. Source file changes

   ```
   if (handle_BACnet())                      // Must be included in the
                                             // when (msg_arrives) task
   ```

3. `BAClon\mapping.nc`

   This file is used to create the BACnet objects and their mappings to network variables.

4. `BAClon\mapping.h`

   This file contains the function prototypes used by mapping.nc.

Additional Requirements for BACnet MS/TP

There are a number of parameters that need to be configurable by the network integrator in order to support BACnet MS/TP. There is programmatic support for your application to provide this capability. For example, your device can implement DIP switches or other user input methods. Your application can read these inputs and use them to configure the BACnet MS/TP interface.

The Neuron BACnet Stack uses the Device object's description to provide a default method allowing a user to dynamically configure parameter settings. The methodology requires the BACnet client to write a special string to the Description property. If this string is formatted appropriately, the corresponding parameter gets updated, and this change gets stored in persistent memory and persists across power cycles.

Setting the Baud Rate

BACnet requires that a device can support baud rates of 9600 bps and 38400 bps at a minimum. The Neuron BACnet MS/TP stack provides support for automatically determining the network baud rate using auto-baud rate detection, and also supports setting the baud rate programmatically.

For auto-baud rate detection, the Neuron BACnet MS/TP stack monitors the channel for valid MS/TP token frames at the configured baud rate. If, after a short interval, no valid MS/TP token frames have been received, but traffic is detected on the MS/TP network, the Neuron's MS/TP channel baud rate is automatically changed to another supported value. This cycle is repeated until a lock on the correct baud rate is established.

If there is silence on the network, the Neuron BACnet MS/TP stack goes into a Token Recovery mode. In this mode, a BACnet device waits for a multiple of its MAC address before attempting to recover a token by 'creating' and sending a new one. This means that if all devices are powered up at exactly the same time, it is likely that the device with the lowest MAC address will establish a new token, and the baud rate of that device will be adopted for the network.

If there is not silence on the network, then a device with the correct baud rate will eventually receive the token, and be allowed to transmit. If the baud rate is not correct, there will never be silence on the network, and the device will never receive a valid token, and this device will never (it is not allowed to) transmit.

To ensure that the network baud rate is set to the best value, follow these steps:

1. If there are other stations on the network that do not support auto-baud rate, set the MAC address for at least one of them to be lower than any auto-baud devices on the channel, including any Neuron BACnet devices.

2. Set the BACnet Router (a BACnet/IP to BACnet MS/TP or BACnet FT to BACnet MS/TP micro-router) to the lowest MAC address on the channel after any non auto-baud devices to establish the network's baud rate.

If a Neuron BACnet device has the lowest MAC address on the channel, the initial baud rate set in software will become the network baud rate.

Setting the MAC address

The example source code defaults the BACnet MS/TP MAC address to 47. To programmatically modify this, declare the following:

```
extern unsigned int This_Station;
This_Station = 23;      // New address, 0 to 127 are valid.
```

To alter the MAC address dynamically, write "**M:***hh*" to the Device Object Description Property, where *hh* is the new hexadecimal MAC address 00 to 7F are valid values

Mapping BACnet Objects to LON NVs

You can map BACnet objects to LON network variables (NVs) and application internal variables using function calls, as demonstrated in the **Initialize_BACnet_Objects()** function, found in the **mappings.nc** file. Analog Objects and Binary Objects use different mapping functions, as follows:

```
Create_Analog_Input(
               // Create an Analog Input
               // object and map
         "Space Temp",
               // The object name of the BACnet Object created
         &localTemperature,
               // The address of the LonWorks network variable
               // or application internal variable
         Get_Analog_Lon_Datatype_Handle(LDT_SNVT_temp_p));
               // The mapping type. There are a number of predefined
               // mapping types, such as this one, but users
               // can create their own as shown later. The predefined
               // types are listed in the mapping.h file.


Create_Analog_Output( .. ) ;
               // Exactly the same as above, but for a
               // BACnet Analog Output object.


Create_Binary_Input(
         "nvoSwitch1Out",
               // The object name of the BACnet object created
         &nvoSwitch1.state,
               // The address of the LonWorks variable of interest.
               // A field of a structure is used here.
         Get_Binary_Lon_Datatype_Handle(LDT_SNVT_switch__state));
               // The mapping type for a binary value.
               // This standard mapping type has an extension
               // that describes the field.

  Create_Binary_Output ( .. ) ;
               // Exactly the same as a Binary Input.
```

Complex Mapping

If a network variable has a structure or union with multiple fields, to make the full NV available through the BACnet interface, you can map each field to a separate BACnet object. For example, the **LDT_SNVT_hvac_overid_percent** mapping type provides a mapping of the **SNVT_hvac_overid.percent** field to a BACnet Analog Output or Input.

It is possible for a BACnet client to both read from and write to an output (e.g. Analog Output), subject to the rules of BACnet objects, such as out-of-service and priority arrays.

Mapping UNVTs

NVs based on UNVTs are mapped in a similar fashion to NVs based on SNVTs, but you must define the mapping macro for your user types. Refer to the **BAClon\mapping.h** and **mapping.nc** files, and use one of the pre-defined mappings as an example to generate an appropriate mapping for the your UNVT, and then you can apply this new mapping to your NVs based on UNVTS in the **BAClon\mapping.nc** file. See below for instructions how to create user defined mappings.

User Definable Scaling

You can add a user scaling factor to the internal list of scaling factors using either integer scaling or floating point scaling. Integer scaling requires less memory, and floating point scaling provides more precision.

For integer scaling, define a scale factor as shown in the following example:

```
UserScaleFactor userScaleFactor = DefineScaleFactor(
   1,    // The multiplier term (signed integer)
   -3,   // The power term (signed integer)
   0,    // The offset term (signed integer)
   );
```

For higher precision floating-point scaling, define a scale factor as shown in the following example:

```
UserScaleFactor userScaleFactor = DefineHiResScaleFactor(
   &fl_OnePointEight,      // The multiplier term (float)
   0,                      // The power term (note: signed integer)
   &fl_ThirtyTwo,          // The offset term (float)
   );
```

See **BAClon\mapping.nc** for examples. Once you have created a new scaling factor, create a new mapping type for it as defined in the next section.

User Definable Mapping Types

You can define new mapping types to support NVs based on SNVTs that are not already mapped, or that are based on UNVTs that you have defined. To create a new mapping function, follow these steps:

1. Add a new mapping tag to the end of the mapping tag **#define** list:

```
#define LDT_SNVT_elec_kwh_l       105
#define LDT_SNVT_new_user_tag     106
```

2. Define the new mapping type:

```
DefineAnalogSystemMappingType(      // Binary also available
      LDT_SNVT_new_user_tag,        // new tag as defined
      DF_ULong,                     // the LonWorks data format
      userScaleFactor,              // as defined
      UNITS_DEGREES_CELSIUS         // one of the units available
      );
```

Once you have defined the new mapping type has been defined, you can use it just like the built-in mapping types, for example:

```
Create_Analog_Input(
      "Space Temp",           // Object name
      &localTemperature,      // LonWorks datapoint
      Get_Analog_Lon_Datatype_Handle(LDT_SNVT_new_user_tag)
                              // new mapping type, as defined
      );
```

Reading and Updating the BACnet Priority Array

You can read and update entries in the BACnet priority array. As described in *Using the Neuron BACnet Server*, the Neuron BACnet Stack uses the BACnet priority array to select the highest priority value for the present value property. This is handled by the Neuron BACnet Stack, and you typically do not have to read or write the priority array from your application. You can update entries in the priority array to interact between modes of network operation. An example use case is as follows:

> A functional block receives an input from an input network variable and uses this to update the value of an output network variable. If the input network variable is set to the invalid value, the application controls the output network variable value. However, in the presence of a BACnet client, the BACnet client may also update the output network variable value.

The example use case can be implemented by mapping a BACnet output variable onto the LON output network variable as described in the sections above. The application typically writes its desired output to the relinquish default value of the BACnet output variable. In the absence of any BACnet writes, the relinquish default value flows through to the present value, and thereafter to the mapped output network variable.

Instead of writing the relinquish default value, you can update any of the priority array values by writing to the present value with a specified priority. The highest priority value (lowest numeric value) in the priority array will flow through to the present value, and thereafter to the mapped output network variable. You can override the output by updating the present value at a higher priority than any other active values.

You can cancel a value at a specified priority by writing a BACnet NULL to the present value with the specified priority.

You can process network variables with multiple fields in the same way, with a BACnet object mapped onto each field. The relinquish default and priority array processing will handle the aggregate operation correctly.

You can read and write values using floats and booleans similar to BACnet, or using LON NV types.

Neuron BACnet API Reference

You can read and update entries in the BACnet priority array using the Neuron BACnet API. This section provides a reference for the Neuron BACnet API functions. To use these functions, include the **AppAPI.h** header file in your Neuron C application. For an example, see the **APIdemo.c** application.

```
BAPI_ERR BACapiInitOrdinaryVariable(
     const void *ordinaryVariable,
     const uint8_t sizeofOrdinaryVariable);
```

This function initializes a local structured variable. A local structured variable is a structured variable that is not a network variable. A true network variable does not need initialization. There is no side effect if a network variable or a local variable that is not a structure is initialized. However, if a local structured variable is not initialized, the Neuron BACnet Server will not aggregate processing of the relinquish default and priority array across all fields of the variable. The server will instead only process the relinquish default and priority array for the one field that the BACnet object is mapped to.

```
void BACapiRegisterCallback(
     void (*callbackFunc) (const int8_t objectHandle));
```

This function registers a callback function that the Neuron BACnet Server calls every time an external BACnet client writes to the present value of a mapped variable.

```
BAPI_ERR BACapiWritePVsnvt(
     const int8_t handle,
     const uint8_t bac_priority,
     const void *value);

BAPI_ERR BACapiWritePVfloat(
     const int8_t handle,
     const uint8_t bac_priority,
     float_type *value);

BAPI_ERR BACapiWritePVbool(
     const int8_t handle,
     const uint8_t bac_priority,
     const Boolean value);
```

These functions write to a mapped variable. The Neuron BACnet Server returns the **handle** parameter when the original BACnet object to LON variable mapping is done in **mappings.nc**.

The **bac_priority** value sets the priority of the write operation, resulting in the value being stored in the priority array at that priority (1 to 16). One cannot write to priority 6, per the BACnet specification. Priority 1 is the highest priority.

```
BAPI_ERR BACapiWriteRDsnvt(const int8_t handle, const void *snvtPtr);
BAPI_ERR BACapiWriteRDfloat(const int8_t handle, float_type *tfloat);
BAPI_ERR BACapiWriteRDbool(const int8_t handle, const boolean val);
```

These functions update the relinquish default value.

```
BAPI_ERR BACapiReadPVfloat(const int8_t handle, float_type *tfloat);
BAPI_ERR BACapiReadPVbool(const int8_t handle, boolean *tbool);
BAPI_ERR BACapiReadPVsnvt(const int8_t handle, void *tsnvt);

BAPI_ERR BACapiReadPAfloat(
     const int8_t handle,
     const int bac_priority,
     float_type *tfloat);
BAPI_ERR BACapiReadPAbool(
     const int8_t handle,
     const int bac_priority,
     boolean *tbool);
BAPI_ERR BACapiReadPAsnvt(
     const int8_t handle,
     const int bac_priority,
     void *tsnvt);

BAPI_ERR BACapiReadRDfloat(const int8_t handle, float_type *tfloat);
BAPI_ERR BACapiReadRDbool(const int8_t handle, boolean *tbool);
```

These functions return the present value, the value in the priority array at a given priority, and the relinquish default values of a commandable BACnet object. The **ReadPV** functions return the same value as the local or network variable value that the BACnet object is mapped onto, and is included mainly for presenting the data in BACnet format.

Change of Value (COV) Support

The Neuron BACnet Stack supports Change of Value (COV) services for Binary Input and Analog Input object types. COV support for Analog Output and Binary Output object types is optional for BACnet and is not supported for the Neuron BACnet Stack.

The Subscribe COV service is fully supported. The BACnet optional Subscribe COV Property service is optional for BACnet and is not supported for the Neuron BACnet Stack.

Both Confirmed and Unconfirmed COV support is provided.

The optional lifetime parameter of COV subscriptions is supported.

COV subscription does not require any configuration, and standard BACnet clients will do the subscription automatically according to the BACnet specification. See the BACnet specification for further information.

Read Property Multiple Support

The Neuron BACnet Stack supports BACnet Read Property Multiple (RPM). This allows block transfers of multiple BACnet properties, which can speed up communications significantly.

The ultimate transfer size APDU is limited to somewhat less than a single packet on the network, currently set to about 206 bytes. Standard BACnet clients will attempt to do a RPM, and if the Neuron BACnet Stack responds with a reject packet due to size constraints, BACnet clients will then break their request down to smaller and smaller lists until a response can be built and sent successfully by the Neuron BACnet Stack.

Read Property Multiple does not require any configuration, and standard BACnet Clients will do the RPM automatically according to the BACnet specification. See the BACnet specification for further information.

Application Image Download

You can download a new application image to a Neuron BACnet device over the MS/TP interface using the standard BACnet file transfer service (*Atomic Write File* or *AWF* service). The AWF service only allows atomic writes of individual blocks of any given file, so additional checks are made to ensure the integrity of the file transfer. The ability to read a file (*ARF*) is not supported.

To transfer a new application image file to a Neuron BACnet device over MS/TP, follow these steps:

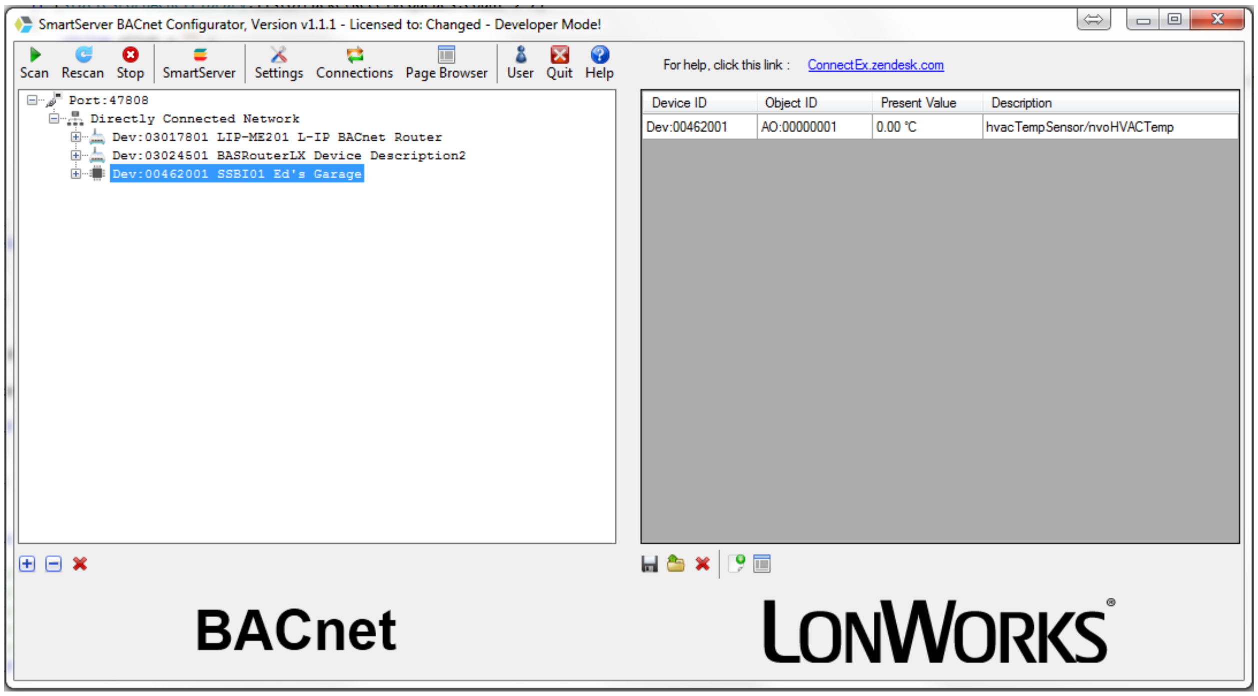
1. Verify the Neuron BACnet MS/TP stack is running on the device to be loaded. The application cannot be offline and the device cannot be unconfigured or applicationless for this to work. This is in contrast to an application load over LonTalk/IP which can work regardless of the device state.
2. Use the standard BACnet AWF service to transfer the <*filename*>**.ndb**, file to your Neuron BACnet device. This file is automatically created by the NodeBuilder software as part of the build process and is located in the NodeBuilder project's Release or Development folder. The Neuron BACnet Stack tracks the start-of-file block, the sequence of block transfers, including any retries, the CRC of the total transfer, and detects when the end of file (according to the prepended length) has been reached.
3. At the end of the file transfer, if the CRC is valid the new image is automatically written to flash memory and the Neuron core is restarted.
4. You can validate a successful download and switchover by reading the version number of the new application if the application implements an application defined configuration property with the version number.

BACnet Test Tools

You can use any of the following tools to test your Neuron BACnet devices:

- **IzoT BACnet Browser** – This is a standard BACnet Client distributed by ConnectEx. It can automatically discover BACnet devices, and displays the discovered devices and BACnet objects in the left BACnet Object List pane. To view the present value and other properties for a BACnet object, right-click the object in the BACnet Object List pane. The present value and a few other properties are displayed and monitored in the right Object Details pane. You can download the IzoT BACnet Browser for free from the following location:

http://www.connect-ex.com/demos_and_downloads/bacnet-browser-for-echelons-izot


SmartServer BACnet Configurator, Version v1.1.1 - Licensed to: Changed - Developer Mode!
Scan
Rescan
Stop
SmartServer
Settings
Connections
Page Browser
User
Quit
Help
For help, click this link : ConnectEx.zendesk.com
BACnet
LONWORKS

```
Port:47808
  Directly Connected Network
    Dev:03017801 LIP-ME201 L-IP BACnet Router
    Dev:03024501 BASRouterLX Device Description2
    Dev:00462001 SSBI01 Ed's Garage
```

Device ID	Object ID	Present Value	Description
Dev:00462001	AO:00000001	0.00 °C	hvacTempSensor/nvoHVACTemp

- Third-party BACnet Clients – you can use any BACnet client to read all BACnet points served by the Neuron BACnet Stack. Contact ConnectEx, Inc. if you encounter any compatibility issues. You can report problems on the following page:

  https://connectex.zendesk.com

- For much more rigorous testing of your application before submitting to BTL, use the **BACnet Test Client** that is available on the following page:

  http://www.bac-test.com/downloads/

- The **Wireshark** protocol analyzer is invaluable for analyzing BACnet traffic. A dissector for BACnet packets is built in. For more information on using Wireshark to analyze BACnet traffic, see the following white paper:

  http://www.bacnet.org/Bibliography/BACnet-Today-08/Karg_2008.pdf

  You can download Wireshark for free from the following location:

  http://www.wireshark.org/download.html

  To use Wireshark, follow these steps:

  1. Download Wireshark from *www.wireshark.org/download.html.*
  2. Install the Wireshark software.
  3. Start Wireshark.
  4. Set up a capture filter for UDP Port 47808 to reduce capture traffic.
  5. Start packet capture with Wireshark. BACnet messages will be dissected and displayed.
  6. If you did not set up a capture filter, you can set up a display filter so only BACnet messages are shown. For more information on how to use Wireshark, see *www.wireshark.org/docs*.

Other Resources

The Official BACnet Website

www.bacnet.org

The BACnet Standard

Standard 135-2012-- BACnet--A Data Communication Protocol for Building Automation and Control Networks (ANSI Approved). 2012

The BACnet Standard is available in paper and PDF form for approximately $170 from ASHRAE.

Browse to the address below and enter for "135" in the search box www.techstreet.com/.

Wiki

You can find additional BACnet information at www.bacnetwiki.com.

Vendor IDs

You can obtain a BACnet Vendor ID from:

www.bacnet.org/DL-Docs/Procedures-Vendor-ID-rev3-15-2012.pdf

PIC (Protocol Implementation Conformance Statement) Statement Information

You can find information on BACnet PIC statements at:

www.bacnet.org/DL-Docs/

Device Profiles

You can find information on BACnet device profiles at:

www.bacnetwiki.com/wiki/index.php?title=Device_Profiles

Appendix A

Glossary

This glossary provides definitions for terms discussed in this manual

Application Device

An IzoT or LON device that runs an ISO/IEC 14908-1 application (OSI Layer 7). The application may run on a Neuron Chip or Smart Transceiver, in which case the device is called a *Neuron hosted device*. If the application runs on another processor besides a Neuron Chip or Smart Transceiver, it is called a *host-based device*.

Application Image

For a Neuron hosted device, the application image is the device firmware that consists of the object code generated by the Neuron C compiler from the user's application program and other application-specific parameters, including the following:

- Network variable fixed and self-identification data
- Network variable device interface data
- Program ID string
- Optional self-identification and self-documentation data
- Number of address table entries
- Number of domain table entries
- Number and size of network buffers
- Number and size of application buffers
- Number of receive transaction records
- Input clock speed of target Neuron Chip or Smart Transceiver
- Transceiver type and bit rate

Application Program

The software code in an IzoT or LON device that defines how it functions. The application program, also referred to as the *application*, may be in the device when you purchase it, or you may load it into the device from application image files (**.APB**, **.NDL**, and **.NXE** extensions for Neuron hosted devices) using the IzoT Commissioning Tool or other network management tool. The application program interfaces with the LonTalk/IP Stack to communicate over the network. It may reside completely in the Neuron Chip or Smart Transceiver, or it may reside on an attached host processor (in a host-based device).

Binding

Process of connecting network variables. Binding creates logical connections (virtual wires) between IzoT and LON devices. Connections define the data that devices share with one another. Tables containing connection information are stored in the device's non-volatile memory, and may be updated by a network management tool or the ISI protocol.

Changeable-Type Network Variable

A network variable that has a type and length that can be changed to that of another network variable type. You can use changeable-type network variables to implement generic functional blocks that work with different types of inputs and outputs.

Channel

The physical media between devices upon which the devices communicate. The LonTalk/IP protocol is media independent; therefore, numerous types of media can be used for channels: Level 4 twisted pair, Cat 5 twisted pair, power line, fiber optics, RF, and other types.

Commissioning

The process in which the network management tool downloads network and application configuration data into a physical device. For devices with application programs not contained in ROM, the network management tool can also download the application program into non-volatile RAM in the device. Devices are usually either commissioned and tested one at a time, or commissioned and then brought online and tested incrementally.

Configuration Properties (CPs)

Configuration properties are data values that define the behavior of an application device by determining the manner in which device application data is manipulated and when device application data is transmitted. A configuration property can be applied to the device, a functional block on the device, or a network variable on the device. Configuration properties can determine the functions to be performed on the values stored in network variables. For example, a configuration property may specify a minimum change that must occur on a physical input to a device before the corresponding output network variable is updated.

Configured

A device state where the device has both an application image and a configured network image. This indicates that the device is ready for network operation.

Connection

The implicit addressing established during binding. A connection links one or more logical outputs (network variables or message tags) to one or more logical inputs.

Connect Button

A button on an ISI device that the user can press to create a connection. The Connect button on an FT 6000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* application is the **SW2** button on the right side of the board.

Connect Light

An LED on an ISI device that provides feedback related to the status of an ISI connection. The Connect light on an FT 6000 EVB running the *NcSimpleIsiExample* or *NcMultiSensorExample* application is **LED2**, which is located directly above the **SW2** button.

Connection Host

A device using ISI installation that initiates the enrollment process by sending a connection invitation specifying a connection assembly.

Connection Member

A device using ISI installation that has joined an ISI connection, but is not the connection host.

Device

A device that communicates on an IzoT or LONWORKS network using ISO/IEC 14908-1, LonTalk/IP, or BACnet/IP. A device may be an application device, network service device, or a router. Devices are sometimes referred to as *nodes* in LONWORKS documentation.

Device Interface (XIF)

The logical interface to a device, abbreviated as *XIF*. A device's interface specifies the number and types of functional blocks; number, types, directions, and connection attributes of network variables; and the number of message tags. The program ID for a device is used as the key to identify each device interface. Each program ID uniquely defines the static portion of the interface. However, two devices with identical static portions may differ if dynamic network variables are added or removed, or if the types of changeable network variables are changed. Thus it is possible to have devices with the same program ID but different device interfaces.

Device Interface (XIF) File

A file that documents a device's interface with a network. The file can be a text file (.**XIF** extension), or it can be a binary file (.**XFB** extension).

Device-Specific Configuration Property

A configuration property that has values that can be modified independent of the network database. Changes made to a device-specific configuration property are not updated in the network database.

Device Template

A device template contains all the attributes of a given device type, such as its functional blocks, network variables, and configuration properties. You can create a device template by importing a device interface (XIF) file supplied by the device manufacturer, or by uploading the device interface definition from the physical device. A device template is identified by its name and its program ID. Both must be unique within a network—you cannot have two device templates with the same name or the same program ID in a single network.

Download

An installation process in which data, such as the application program, network configuration, and/or application configuration, is transferred over the network into a device.

Free Topology

A connection scheme for the communication bus that eases traditional transmission line restrictions of trunks and drops of specified lengths and at specified distances, and terminations at both ends. Free topology allows wire to be strung from any point to any other, in bus, daisy chained, star, ring, or loop topologies, or combinations thereof. It only requires one termination anywhere in the network. This can reduce the cost of wiring significantly. Free topology can be implemented with either Level 4 or Cat 5 twisted pair cable.

FT 6000 EVB

An evaluation board that uses Echelon's IzoT FT 6050 Smart Transceiver. It features a compact design that includes the following I/O devices that you can use to develop prototype devices and run the FT 6000 EVB examples: 4 x 20 character LCD display, 4-way joystick with center push button, 2 push-button inputs, 2 LED outputs, light-level sensor, and temperature sensor.

Functional Block (FB)

A collection of network variables, configuration properties, and associated behavior that defines a specific system functionality. Functional blocks define standard formats and semantics for how information is exchanged between devices on a network. Each functional block implements a functional profile.

Functional Block Array

A set of identical functional blocks. A functional block array is useful if your device contains two or more similar I/O devices such as temperature or pressure sensors, switches, lights, or dials that will each have an identical external interface. In addition, a functional block array saves code space and reduces the number of when-tasks in your code.

Functional Profile

A template for a functional block that enables equipment specifiers to select the functionality they need for a system. Each functional profile defines mandatory and optional network variable and configuration property members along with their intended usage. A number of generic standard functional profiles are available for generic devices such as simple sensor and actuators. Many industry-specific standard functional profiles are available for industry-specific applications. Industry-specific standard profiles are developed through a review and approval process, including a cross-functional review to ensure the profile will interoperate within an individual subsystem and also provide interoperability with other subsystems in the network.

User-defined functional profiles can be created if no appropriate standard profiles are available.

I/O Object

An instantiation of an I/O model. An I/O objects consists of a specific I/O model, and its pin assignment, modifiers, and name.

Interoperable Self-Installation (ISI) Protocol

The standard protocol for performing self-installation in IzoT and LONWORKS networks. ISI is an application-layer protocol that lets you install and connect devices without using a separate network management tool. It is typically used in small networks, and may be used in any network with less than 200 devices with simple connection and configuration requirements.

ISI Mode

An installation scenario in which the ISI protocol is used (instead of a network management tool) to install devices and create network variables connections.

IzoT Net Server

A network operating system that provides services for interoperable IzoT and LONWORKS installation, maintenance, monitoring, and control tools such as the IzoT Commissioning Tool. Using the services provided by the IzoT Net Server, tools and plug-ins from multiple vendors can work together to install, maintain, monitor, and control IzoT and LONWORKS networks.

IzoT Network Database

A database managed and maintained by the IzoT Net Server that includes the network and device configuration data for an IzoT or LONWORKS network.

Izot Net Server Computer

A computer running the IzoT Net Server software. The IzoT Net Server manages and maintains a network database for each network managed by the server.

Out-Of-Service

A state of a BACnet object that allows writes to the Present_Value property to take place without propagating the new value to the hardware output itself. This is typically used for testing.

Local Client

An IzoT Net application running on the same computer as the IzoT Net Server.

NcMultiSensor Local Device

An FT 6000 EVB running the *NcMultiSensorExample* application that receives **SNVT_lux** and/or **SNVT_temp_p** output network variable updates from another device (an NcMultiSensor remote device). The local device displays the temperature and light level values received from the remote device in the Remote Info Mode panel on its LCD. A remote device may be another FT 6000 EVB board running the *NcMultiSensorExample* application.

IzoT Commissioning Tool (CT) Browser

An IzoT Net plug-in that provides a table view of the network variables and configuration properties of selected devices and functional blocks. You can use the IzoT CT Brower to monitor and control the network variables and configuration properties in a network.

IzoT CT Drawing

An IzoT CT drawing contains the graphical representation of an IzoT or LONWORKS network.

IzoT Commissioning Tool (CT)

A network management tool that uses Visio as its graphical user interface. You can use IzoT CT to design, commission, maintain, and document distributed control networks.

IzoT Network

A network of intelligent devices (such as sensors, actuators, and controllers) that communicate with each other using ISO/IEC 14908-1 control services with IP transport services over one or more communications channels.

LONMARK Logo

A distinctive logo applied to IzoT and LONWORKS devices that have been certified to the interoperability standards of LONMARK International.

LonTalk/IP Protocol

The LonTalk/IP Protocol is a communications protocol for control applications that provides comprehensive control services over standard IP. The upper layers of the LonTalk/IP Protocol are called the *LonTalk/IP control services*, and the lower layers are called the *LonTalk/IP transport services*. The LonTalk/IP control services are defined by Layers 4 through 7 of the ISO/IEC 14908-1 Control Network Protocol (CNP). The transport services are channel-type dependent. For native IP channels such as Ethernet or Wi-Fi, the transport services are defined by the RFC 768 User Datagram Protocol (UDP) transport layer and the RFC 791 Internet Protocol (IP) internet layer. For ISO/IEC 14908-1 native LON channels such as FT, the Layer 2 through 3 transport services are defined by the ISO/IEC 14908-1 protocol standard. For FT, the Layer 1 Physical layer communications media is defined by the ISO/IEC 14908-2 protocol standard.

LONWORKS Network

A network of intelligent devices (such as sensors, actuators, and controllers) that communicate with each other using the ISO/IEC 14908-1 Control Network Protocol or LonTalk/IP protocol over one or more communications channels.

LONWORKS Technology

The technology that allows for the creation of open, interoperable control networks that communicate with the ISO/IEC 14908-1 Control Network Protocol or the LonTalk/IP protocol. LONWORKS technology consists of the tools and components required to build intelligent devices and to install them in control networks.

Managed Network

A network where a shared network management server, such as the Izot Net Server, is used to perform network installation.

Mandatory Network Variable/Configuration Property

A network variable/configuration property that must be implemented by the functional block, as specified by the functional profile that the functional block is instantiating.

Monitored Connection

A network variable connection in which the current values are being monitored, typically by a network management tool or HMI application. .

Network Interface

An IzoT or LONWORKS device that provides a Layer 2 or Layer 5 LonTalk/IP interface to an external host computer such as a computer or a handheld maintenance tool.

Network Variable (NV)

A data value or structured set of values on a device that can be shared with other devices. Network variables are data items (such as temperature, the state of a switch, or actuator position setting) that a particular device application program expects to receive from other devices on the network (an *input network variable*) or expects to make available to other devices on the network (an *output network variable*).

Network Variable/Configuration Property Types

A network variable or configuration property type defines the structure and contents of the data object. A network variable type can be either a standard network variable type (SNVT) or a user-defined network variable type (UNVT). A configuration property type can be a standard configuration property type (SCPT) or a user-defined configuration property type (UCPT)

Neuron C

A programming language based on ANSI C that you can use to develop applications for Neuron Chips and Smart Transceivers. It includes network communication, I/O, interrupt-handling, and event-handling extensions to ANSI C, which make it a powerful tool for the development of IzoT and LONWORKS device applications.

Neuron Chip

A semiconductor component specifically designed for providing intelligence and networking capabilities to low-cost control devices. The Neuron Chip includes a communication port for connections to various network types.

Neuron Core

A processor core that includes up to four processors that provide both communication and application processing capabilities. Two processors execute the Layer 2 through 6 implementation of the ISO/IEC 14908-1 Control Network Protocol and the third executes Layer 7 and the application code. The Series 5000 and Series 6000 Neuron cores include a fourth processor for interrupt service routine (ISR) processing. The Neuron core is implemented in Neuron Chips and Smart Transceivers.

Neuron Firmware

A complete operating system including an implementation of the ISO/IEC 14908-1 protocol used by a Neuron Chip or Smart Transceiver. The Neuron firmware for the Series 6000 core includes implementations of the LonTalk/IP and BACnet/IP protocol stacks.

Neuron ID

A 48-bit number assigned to each Neuron core at manufacture time. The Series 6000 Neuron core uses an IP-standard MAC ID for the Neuron ID. Each Neuron Chip and Smart Transceiver has a unique Neuron ID, making it like a serial number.

Node Object

A functional block that monitors the status of all functional blocks in a device and makes the status information available for monitoring by a network management tool.

NodeBuilder Tool

A hardware and software platform that is used to develop applications for Neuron Chips and Smart Transceivers. The NodeBuilder tool provides complete support for creating, debugging, testing, and maintaining IzoT and LONWORKS devices. You can use the NodeBuilder tool all to create many types of devices, including VAV controllers, thermostats, washing machines, card-access readers, refrigerators, lighting ballasts, blinds, and pumps. You can use these devices in a variety of systems including building controls, lighting controls, factory automation, and transportation.

Non-const Device-specific Configuration Property

A configuration property that can be changed by the device application or a network management tool. An example of a non-const device-specific configuration property is the **SCPTnwrkCnfg** configuration property in the **Node Object** functional block of the *NcMultiSensorExample* and *NcSimpleIsiExample* applications. This configuration property stores the current network configuration mode (ISI or managed) of the example application.

OffNet

A management mode of a network management tool in which network configuration changes are stored in the network database, but not propagated to the devices on the network. To send the changes to the devices, you place the tool OnNet. If the tool is OffNet and attached to the network, you can still perform read operations on the network.

OnNet

A management mode of a network management tool in which network configuration changes are propagated immediately by the tool to the devices on the network.

Optional Network Variable/Configuration Property

A network variable or configuration property listed as an optional component in a functional profile. Functional blocks can elect not to implement optional network variables or configuration properties specified by the functional profile that the functional block is instantiating.

Out-Of-Service

A state of a BACnet object that allows writes to the present value to take place without updating local hardware. This is typically used for testing.

Peer-To-Peer

A control strategy in which independent intelligent devices share information directly with each other and make their own control decisions without the need or delay of using an intermediate, central, or master controller. Peer-to-peer control enhances system reliability by eliminating the master (a single point of failure) and reduces installation and configuration cost inherent in peer-to-peer designs.

Priority Array

An array of 16 output values for a BACnet output. Each value may be a valid or invalid value. The highest priority valid value is copied to the present value of the BACnet output, with the first entry having a priority of 1, the last entry having a priority of 16, and 1 being the highest priority.

Program ID

A unique, 64-bit identifier that identifies the device interface (XIF) for a device. Typically represented as a 16-hex digit ID.

Relinquish Default

A request that cancels' a write operation at a given priority. The request is made by writing a NUL value to the present value at the appropriate priority to be relinquished, making the value at the specified priority invalid. The highest valid value remaining in the priority array becomes the new present value. If no suitable values remain in the priority array, then the relinquish default value is transferred. In the Neuron BACnet Stack implementation, the relinquish default value is updated by any external LON input to the network variable.

Remote Client

An IzoT Net application that communicates with the IzoT Net Server running on a separate computer over a LonTalk/IP channel

NcMultiSensor Remote Device

An FT 6000 EVB running the *NcMultiSensorExample* application that sends **SNVT_lux** and/or **SNVT_temp_p** output network variables updates to an FT 6000 EVB running the *NcMultiSensorExample* application (the *NcMultiSensor local device*). The temperature and light level values are displayed in the Remote Info Mode panel on the LCD of the local device.

Remote Network Interface (RNI)

A network interface that enables you to connect an IzoT Net Server, LNS Server, or OpenLDV-based application to an IzoT or LONWORKS network via a TCP/IP connection. RNIs are available on the IzoT Router, SmartServer, and i.LON 600 Router.

Resource File

A file included with an IzoT or LONWORKS device that defines profiles and data types implemented by the device. Resource files hold definitions of standard and user-defined resources, including network variable and configuration property types, functional profiles, enumerations, and formatting rules to display network variable and configuration properties in a readable form. Resource files are used during device development, installation, maintenance, and management. Standard resource files are distributed by LONMARK International. User-defined resource files are created and managed by the device developer during device development.

Self-Installed Network

A network that has network addresses and connections created without the use of a network management tool. In a self-installed network, each device contains code which implements the ISI protocol and that replaces parts of the network management server's functionality, resulting in a network that no longer requires a special tool or server to establish network communication or to change the configuration of the network.

Service Button

A push button or other actuator on an IzoT or LONWORKS device that is used during installation to acquire the device's Neuron ID. For a Neuron hosted device, the button is connected to the service pin of the Neuron Chip or Smart Transceiver. When this pin is activated, the Neuron core sends a broadcast message containing its Neuron ID and program ID. The method used to implement the Service button varies from device to device. Examples of mechanical methods include grounding via a push button or using a magnetic reed switch. By attaching one of the device's I/O pins to the service pin, the service pin can also be put under software control as long as the application code is being executed. For example, the device can ground the pin when the device is moved or when a predefined series of I/O occurs. The service pin can also be used to drive an LED that indicates the device's state. The Service LED is solid on when the device is applicationless, blinks slowly when the device has an application and is unconfigured, is off when the device has an application and is configured. Some applications also implement additional service pin blink patterns.

Standard Configuration Property Type (SCPT)

A standardized definition of the structure, encoding, scaling, units, and usage for a configuration property type that is used to define the structure and semantics for a datapoint that is used to configure the operation of a device. SCPTs are defined and published by LONMARK International to simplify and speed application development and to facilitate interoperability. SCPTs are defined for a wide range of configuration properties used in many kinds of functional profiles, such as hysteresis bands, default values, minimum and maximum limits, gain settings, and delay times.

In addition to standard or user-defined network variable types, which define the data type, formatting rules, limits and units, a SCPT also define semantics. For example, the **SNVT_time_sec** standard network variable type defines a data type for exchanging durations of time, in seconds. The **SCPTmaxSentTime** standard configuration property type references **SNVT_time_sec**, but adds semantics by clarifying that this configuration property defines the maximum period of time between consecutive transmissions of the current value. See *types.lonmark.org* for a current list and documentation.

Standard Functional Profile

A standardized definition of a network visible component of the device interface. A profile encapsulates and defines a set of network variables and configuration properties for the network visible component, and specifies how they are used. Standard functional profiles are defined and published by LONMARK International. See *types.lonmark.org* for a current list and documentation. See *Functional Profile* for more information about functional profiles.

Standard Network Variable Type (SNVT)

A standardized definition of the structure, encoding, scaling, and units for a network variable type. SNVTs are defined and published by LONMARK International to simplify and speed application development and to facilitate interoperability by providing a well-defined interface for communication between devices. See *types.lonmark.org* for a current list and documentation.

Stencil

A collection of master shapes that can be reused in the IzoT Commissioning Tool.

TP/FT-10

The standard free topology twisted pair LONWORKS channel type. The channel bit rate is 78 kbps bit rate. Defined by the ISO/IEC 14908-2 standard. Also referred to as *FT*.

User-defined Configuration Property Type (UCPT)

A manufacturer-specific of the structure, encoding, scaling, units, and usage for a configuration property type that is used to define the structure and semantics for a datapoint that is used to configure the operation of a device. LONMARK-certified devices must have UCPTs documented in resource files according to a standard format, in order to allow the devices to be configured without the need for proprietary configuration tools. See *Standard Configuration Property Type (SCPT)* for more information on configuration property types.

User-defined Functional Profile

A manufacturer-specific definition of a network visible component of the device interface. A profile encapsulates and defines a set of network variables and configuration properties for the network visible component, and specifies how they are used. See *Functional Profile* for more information about functional profile templates.

User-defined Network Variable Type (UNVT)

A manufacturer-specific definition of the structure, encoding, scaling, and units for a network variable type. LONMARK-certified devices must have UNVTs documented in resource files according to a standard format, in order to allow the devices to be interoperable.

Virtual Functional Block

A functional block created by a network management tool that that contains the network inputs and outputs for a device that are not part of other functional blocks on the device.