User Guide

Babel Buster 2

Model BB2-7040

BACnet ZigBee Gateway

Rev. 1.2 – February 2011

User Guide

Babel Buster 2

Model BB2-7040
BACnet ZigBee Gateway

Rev. 1.2 – February 2011

IMPORTANT SAFETY CONSIDERATIONS:

Proper system design is required for reliable and safe operation of distributed control systems incorporating Babel Buster series gateways and other such devices. It is extremely important for the user and system designer to consider the effects of loss of power, loss of communications, and failure of components in the design of any monitoring or control application. This is especially important where the potential for property damage, personal injury, or loss of life may exist. By using the Babel Buster series gateway or any other Control Solutions, Inc., product, the user has agreed to assume all risk and responsibility for proper system design as well as any consequence for improper system design.

Information in this document is subject to change without notice and does not represent a commitment on the part of Control Solutions, Inc. This document is provided "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Control Solutions may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time. This product could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes may be incorporated in new editions of the publication.

Contents

1 Overview 1

1.1 How to Use This Guide 1

1.2 The BB2-7040’s Role on the ZigBee Network 1

1.3 Object Server Model for a Gateway 2

1.4 Overview of the BB2-7040 3

2 Out of the Box – Connecting for the First Time 4

3 Initial Connection of ZigBee Device 8

4 Creating a Data Parse Mask 12

5 Setting Up the ZigBee Device List 14

5.1 Network Identifier 14

5.2 Fixed 64-bit Address 15

5.3 Search String 15

6 Setting Up ZigBee Client Read and Write Maps 16

6.1 ZigBee Client Read Maps 16

6.2 ZigBee Client Write Maps 19

7 Setting Up BACnet 22

7.1 BACnet Device List 22

7.2 BACnet Client Read Maps 23

7.3 BACnet Client Write Maps 25

8 System Setup 28

8.1 Configuration Files 28

8.2 BACnet Port Settings 29

8.3 Local Port Settings 30

8.4 User Accounts and Passwords 31

9 Data Objects 32

10 ZigBee Diagnostic Pages 34

11 Programming ZigBee I/O in Basic 39

11.1 Web Interface Pages for Programming 39

11.2 Script Basic Enhancements Specific to ZigBee 41

12 Object Properties 44

12.1 Data Object Properties (Analog, Binary, Multi-state) 44

12.2 Device Object Properties 45

13 Trouble Shooting 47

13.1 Error Codes, Reliability Codes 47

13.2 Auto-Reset Errors 47

Appendix A Connecting Control Solutions Wireless Devices 48

A.1 Configuring the Temperature Sensor 48

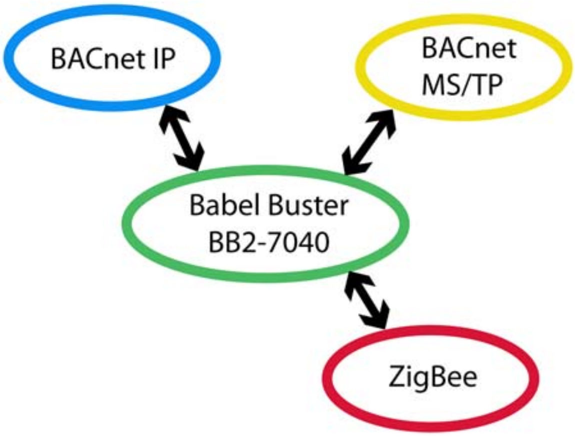
A.2 Configuring the Router 48

A.3 Generic Configuration and the X-CTU Profile 48
Appendix B Connecting Digi International Wireless Devices 50
B.1 Configuring the Digi XBee Sensor 50
B.2 Configuring the Digi XBee SmartPlug 50
B.3 Configuring the Digi Wall Router 50
Appendix C Connecting Point Six Wireless Devices 51
C.1 Changes Required in the BB2-7040 51
C.2 Compatibility Between Point Six and Digi Sensors 51
C.3 Configuring BB2-7040 to Recognize Point Six Sensors 51
Appendix D Gateway Setup Quick Start 55
Revision History 56

1 Overview

1.1 How to Use This Guide

Section 1 gives an overview of the BB2-7040 gateway. Section 2 talks about connecting your PC to the BB2-7040 for the first time. Section 3 talks about connecting your first ZigBee device. The remaining sections provide additional detail about using the BB2-7040.



1.2 The BB2-7040's Role on the ZigBee Network

The Babel Buster BB2-7040 will function as the Coordinator on the ZigBee network. Other devices on the network should be Endpoints or Routers. If you will have 10 endpoints or less, you do not need routers except to solve range problems. If you will have more than 10 endpoints, you need routers to coordinate traffic back to the coordinator.

The most anticipated application scenario that the BB2-7040 is intended to support is a network of wireless sensors connected to a wired BACnet LAN or backbone. In this instance, it is important for the BB2-7040 to be the coordinator since it needs to be on the receiving end of all ZigBee traffic. The other possible application for the BB2-7040 is to make a wireless connection to a piece of BACnet equipment where the BB2-7040 and the BACnet equipment is intended to be a slave to a larger system. In this instance, the BB2-7040 should be reprogrammed with the ZigBee module functioning as a router. Contact the factory about obtaining a router version of the BB2-7040.

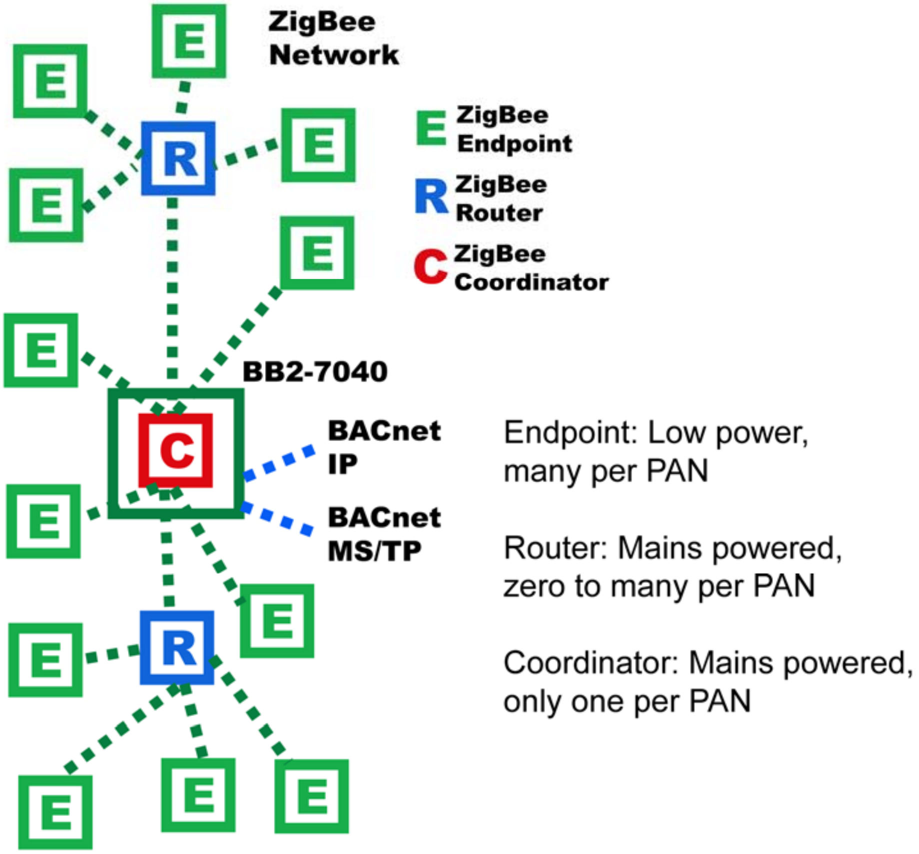
Using ZigBee as a wireless link between a BACnet device and the BACnet network, with two BB2-7040's – one on each end – is possible. However, only one of the BB2-7040's can be a coordinator, meaning one must be reprogrammed as a router. Exchange of a small set of simple data points can be supported in this manner. BACnet protocol specification does define BACnet tunneling over ZigBee. This tunneling is not yet supported by BB2-7040.

1.3 Object Server Model for a Gateway

Control Solutions gateways are not simple protocol translators. It is not possible to do an effective job of simply converting one protocol directly to another. Any attempt to do so would likely have negative effects on the networks on both sides of the gateway. An effective solution requires an intelligent device that can properly and efficiently act as a native device on each network. Control Solutions gateways function as two native devices, one on each network, with a shared data base in between them. They function as clients and/or servers on each network.

The central data element in every Control Solutions gateway is an "object". Each object has rules for accessing that object which are specific to the protocol of the network. Each object has at least two sets of rules, one set for each of the two (or more) networks that may access the object. The object model is often optimized to cater to a specific protocol, and will most often favor the more complex protocol. For example, all Control Solutions gateways that include BACnet connectivity will have objects that are defined primarily as BACnet objects, with auxiliary rules for accessing the objects via Modbus (or some other protocol).

Control Solutions gateways function as servers, providing a copy of the most recent data found in its data base when a client requests that data. In master/slave terms, the server is a slave while the client is a master. Some applications will treat the gateway as a server from both (all) networks connected. But most applications will want the gateway to be a server on one side, and a client on the other side.


ZigBee Network
E ZigBee Endpoint
R ZigBee Router
C ZigBee Coordinator
BB2-7040
BACnet IP
BACnet MS/TP
Endpoint: Low power, many per PAN
Router: Mains powered, zero to many per PAN
Coordinator: Mains powered, only one per PAN
E
R
C

1.4 Overview of the BB2-7040

Control Solutions' Babel Buster® BB2-7040 is a BACnet client/server device that functions as a ZigBee Coordinator and gateway. After setting up the appropriate mappings in the BB2-7040, data from ZigBee sensors will appear as BACnet objects on BACnet IP and/or BACnet MS/TP. Data written to BACnet objects can also be propagated to ZigBee actuators by the BB2-7040.

The BB2-7040 expects most ZigBee endpoints to be low power devices that spend most of their time sleeping. Therefore the traditional gateway behavior of polling slave devices will not work. The BB2-7040 operates primarily in Listen mode, simply waiting for ZigBee devices to report in. When the ZigBee devices do report in, the BB2-7040 searches its known device list and data maps to see if the reported data should be saved as a local BACnet object's Present Value.

The collection of BACnet objects found in the BB2-7040 will maintain the most recently received ZigBee data as their Present Value. Other BACnet clients may then query the BB2-7040 server to obtain that most recent data using a simple Read Property request.

BACnet clients may use a Write Property request to any of the BB2-7040's output objects to send data to a ZigBee device when mapped for writing. The BB2-7040 will transmit to the ZigBee device, and a properly configured ZigBee device will receive the transmission at the end of its sleep period.

The BB2-7040 may also function as a BACnet client, and will use Read and Write Property requests to share data with other BACnet servers (slaves). The BB2-7040 therefore has the ability to transfer data between ZigBee devices and other BACnet devices that operate as a slave only.

The BB2-7040 initial release provides some degree of higher level support for ZigBee products transmitting raw data packets. The ZigBee Alliance has approved two functional profiles as of this writing. The HVAC profile is still in development. As specifications become available, along with devices that implement the profiles, the BB2-7040 will be updated. Firmware in the BB2-7040 is field upgradeable.

The 8/18/2009 version of the XBee OEM manual includes some discussion of ZigBee Device Objects (ZDO) and XBee support of ZDO. The BB2-7040 will not prevent use of any of the ZDO support; it simply does not automatically do ZDO support for you. It will at some later time as implementation of ZDO becomes more wide spread.

The BB2-7040 is based on the XBee OEM module from Digi International. The API command reference may be found in section 10 in the document at this link (as of March 2010):
http://ftp1.digi.com/support/documentation/90000976_D.pdf

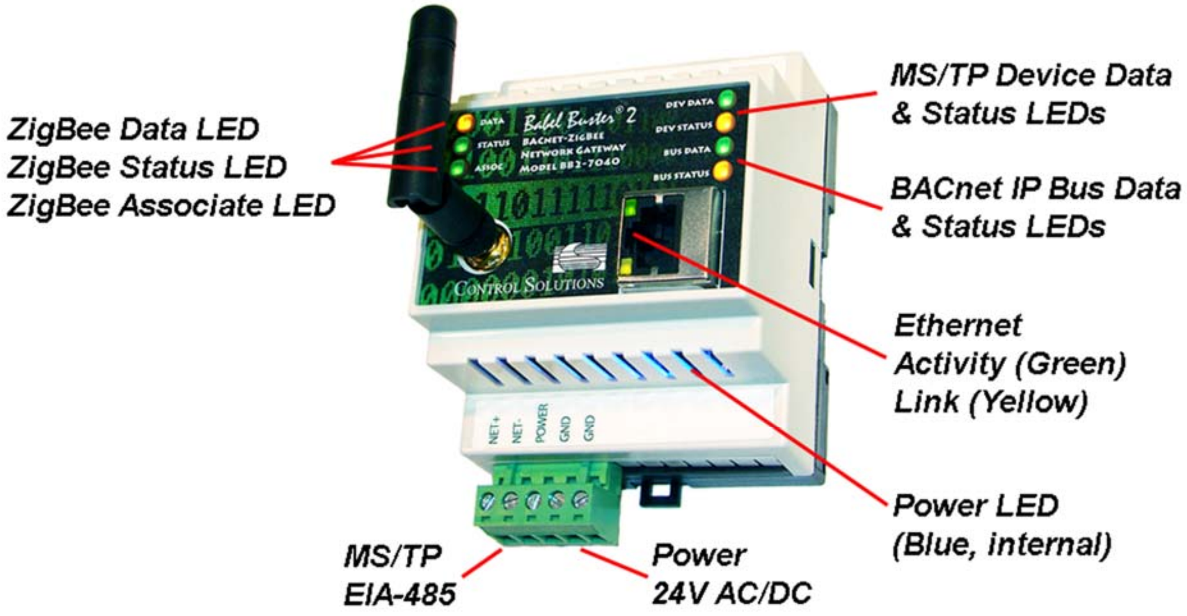
If this link is no longer valid, contact support@csimn.com for an updated link.

Available Models:

BB2-7040-US	Standard BB2-7040 for North America and Australia, ZigBee Coordinator
BB2-7040-EU	Standard BB2-7040 for Europe (lower powered RF), ZigBee Coordinator
BB2-7040-US-R	BB2-7040 North America/Australia, ZigBee Router (non-stock, build to order)
BB2-7040-EU-R	BB2-7040 Europe, ZigBee Router (non-stock, build to order)

2 Out of the Box – Connecting for the First Time

You only need to connect 24V power and Ethernet to begin working with the BB2-7040. Connect the Ethernet via a switch (or crossover cable) as you would any other Ethernet network device.

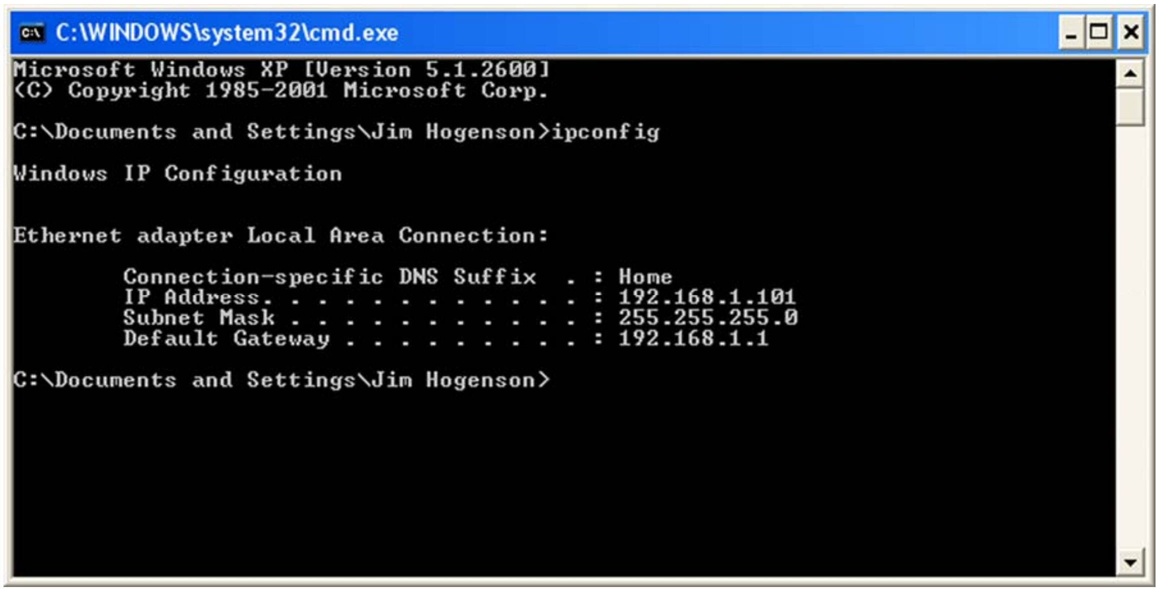


Upon power-up, the embedded server will take about a minute to boot up. A blue LED should be visible inside the device indicating power is present. Both the Ethernet Link and Activity LEDs should light up upon power-up assuming Ethernet is connected to a switch. After bootup, the Link LED indicates a network connection is present, and the Activity LED will flash any time network traffic is detected.

The ZigBee Status LED should come on solid green and remain on. The Associate LED should flash green at a rate of about once a second. When data is received by the ZigBee radio, the data light will flicker. If you do not have any ZigBee devices powered up, this LED will remain off.

You will need to start by establishing a connection between your PC and the BB2-7040. The BB2-7040 comes preconfigured to a static IP address of 10.0.0.101. This will be the local domain your PC is already on with certain types of network routers. If this is the case, you need to be sure nothing else is at 10.0.0.101 before putting it on the network, or simply do not connect it to your office network.

If your PC is on some other domain (192.168.0.0 is a common one), you will need to set up a temporary routing table entry on your PC. Start by finding out what IP address your PC is currently at. Do this by opening a command prompt, and typing the command ‘ipconfig’. Note the address.

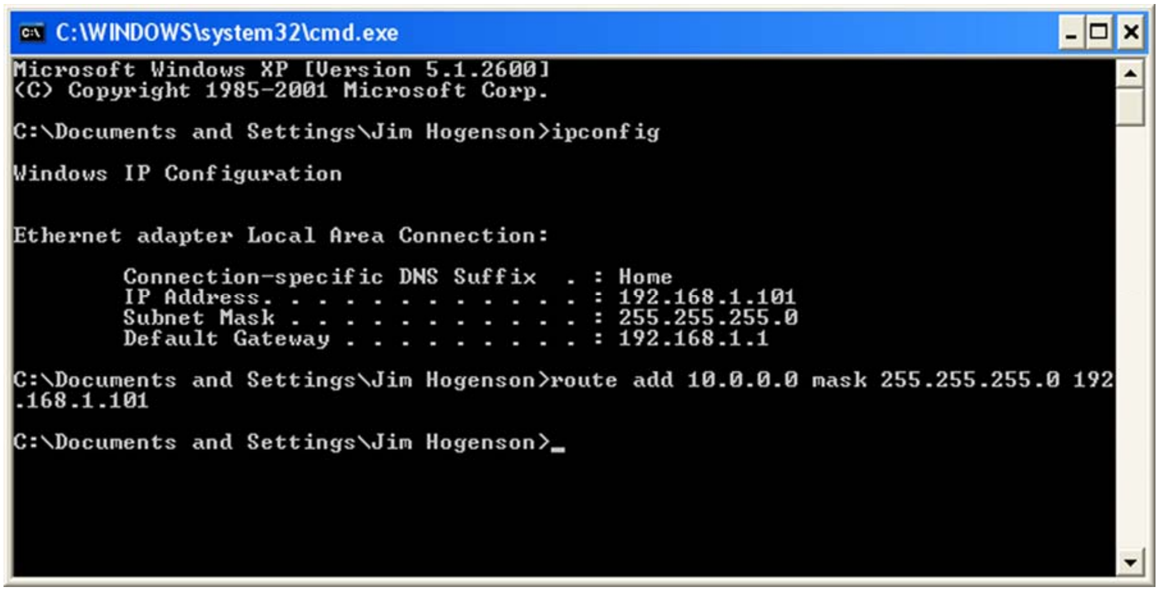


```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Jim Hogenson>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : Home
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\Documents and Settings\Jim Hogenson>
```

Once you have identified your PC’s IP address, use the route command to add a routing table entry. The syntax, using the above IP address as the example, would be:

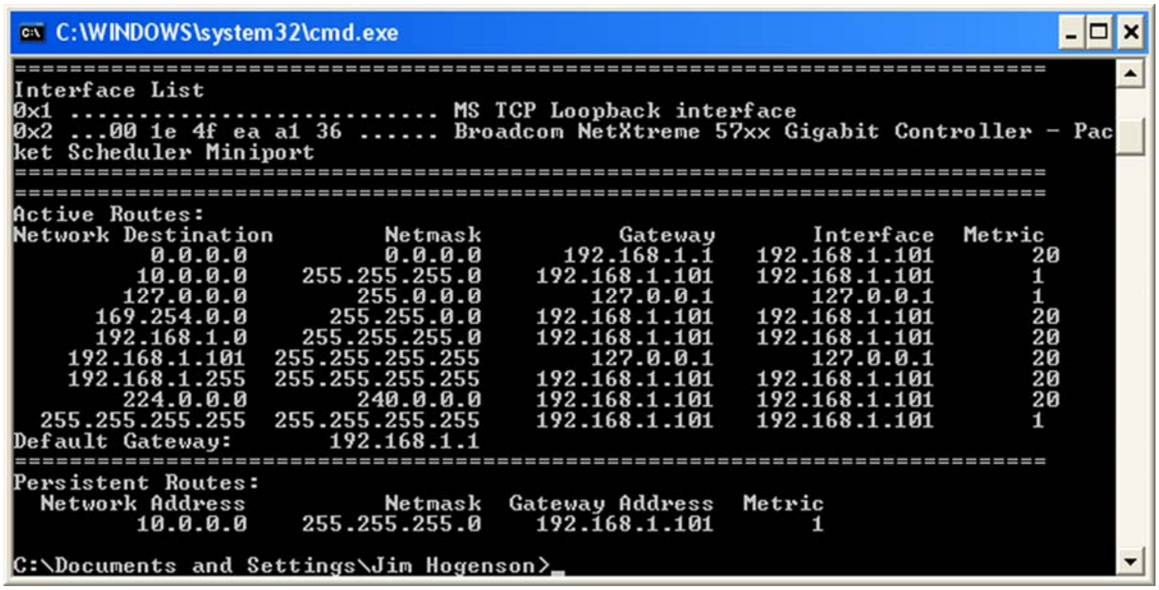
```
route add 10.0.0.0 mask 255.255.255.0 192.168.1.101
```

Note that the mask you use should also match whatever you found with ipconfig.



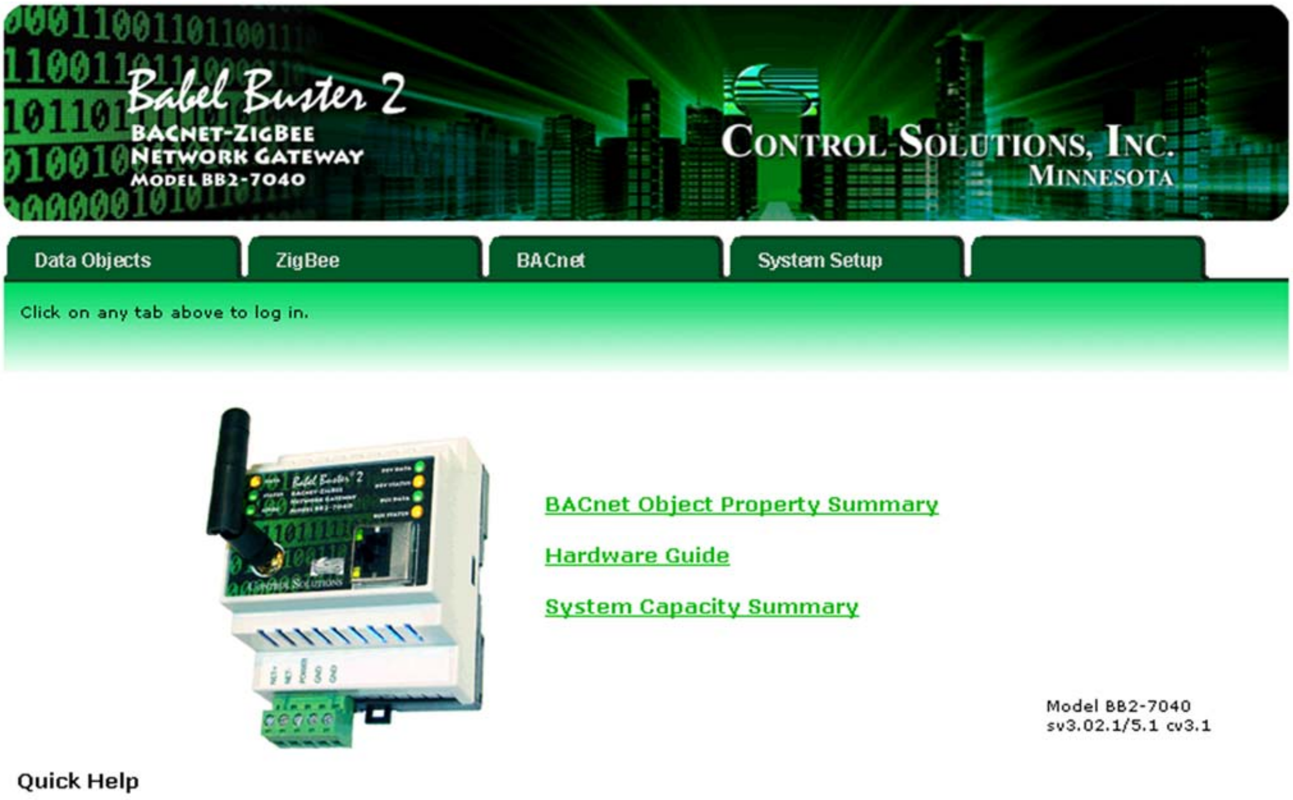
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Jim Hogenson>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : Home
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\Documents and Settings\Jim Hogenson>route add 10.0.0.0 mask 255.255.255.0 192
.168.1.101

C:\Documents and Settings\Jim Hogenson>_
```

Once you have done the route command, you can simply do ‘route print’ to verify your routing table. The result of ‘route print’ should look somewhat like the example that follows, although this will vary greatly from one network and PC to another.



```
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x2 ...00 1e 4f ea a1 36 ...... Broadcom NetXtreme 57xx Gigabit Controller - Pac
ket Scheduler Miniport
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1   192.168.1.101     20
         10.0.0.0    255.255.255.0    192.168.1.101   192.168.1.101      1
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1      1
      169.254.0.0      255.255.0.0    192.168.1.101   192.168.1.101     20
      192.168.1.0    255.255.255.0    192.168.1.101   192.168.1.101     20
    192.168.1.101  255.255.255.255        127.0.0.1       127.0.0.1     20
    192.168.1.255  255.255.255.255    192.168.1.101   192.168.1.101     20
        224.0.0.0        240.0.0.0    192.168.1.101   192.168.1.101     20
  255.255.255.255  255.255.255.255    192.168.1.101   192.168.1.101      1
Default Gateway:       192.168.1.1
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
         10.0.0.0    255.255.255.0    192.168.1.101       1

C:\Documents and Settings\Jim Hogenson>_
```

Once your route entry is established, open your browser, and browse to "http://10.0.0.101". You should see the "home" page of the BB2-7040 illustrated below.



Click on any tab above to log in.

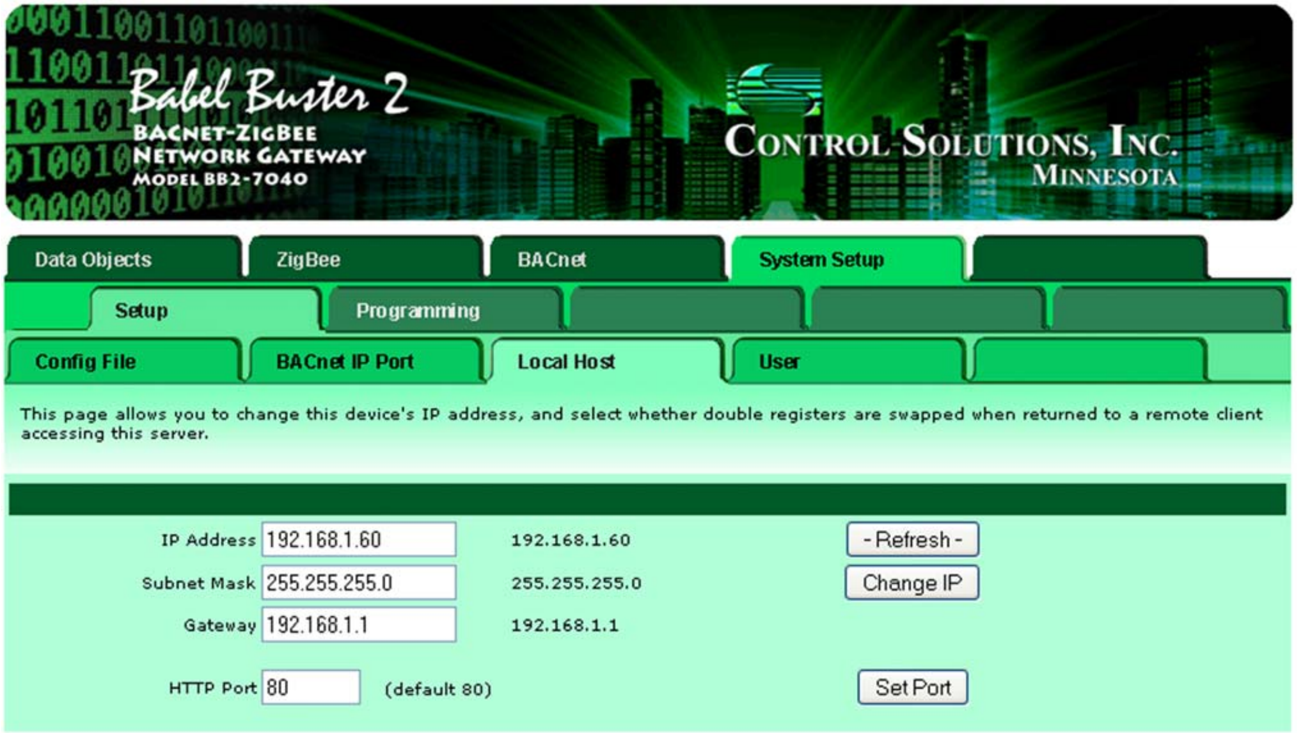
Model BB2-7040
sv3.02.1/5.1 cv3.1

You may now begin to browse your BB2-7040. You will need to log in with a password. There are two default user accounts pre-configured as shipped:

User: system	Password: admin	
User: root	Password: buster	(must log in as 'root' to change IP address)

Once you have established a connection with the gateway, you can proceed to change its IP address. Set the IP address and subnet mask to something more compatible with your network. If your BB2-7040 will be attempting to connect to other IP devices, you must also specify the gateway.

You must be logged in as "root" in order to change IP address. If not logged in as root, close your browser, and re-login. Then change the IP address and click the Change IP button. The new address will be saved to non-volatile memory which takes several seconds. Once the update is done, you must reboot (power cycle) to get the new IP address to take effect.

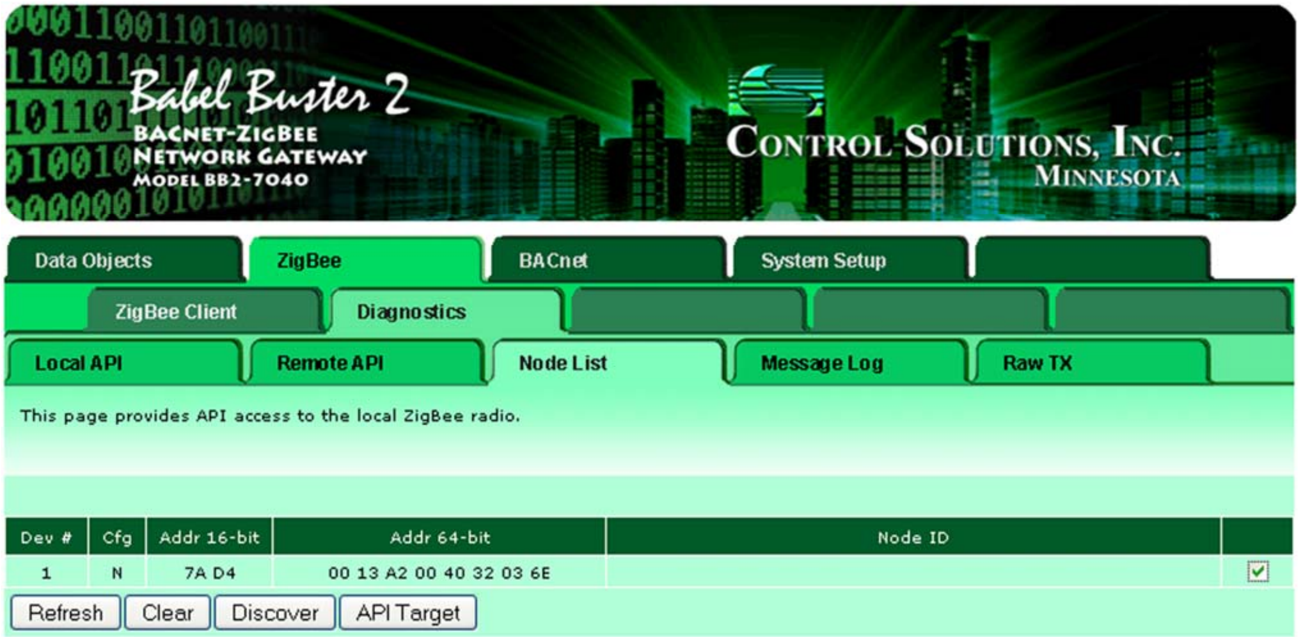


This page allows you to change this device's IP address, and select whether double registers are swapped when returned to a remote client accessing this server.

IP Address	192.168.1.60	192.168.1.60
Subnet Mask	255.255.255.0	255.255.255.0
Gateway	192.168.1.1	192.168.1.1
HTTP Port	80	(default 80)

3 Initial Connection of ZigBee Device

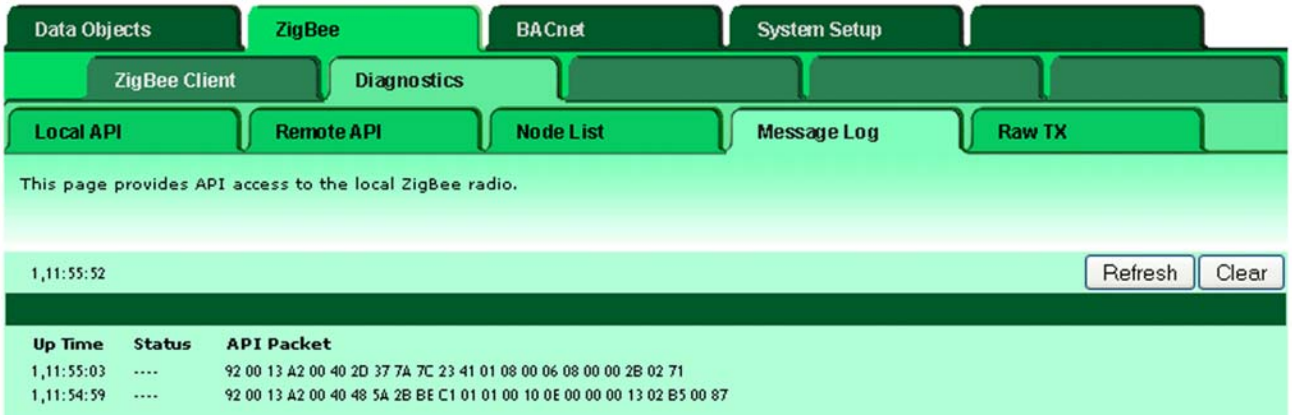
This section will show the general sequence of activity for connecting a ZigBee device for the first time. We will be using a generic XBee Series 2 ZigBee device with firmware programmed to be an endpoint using API mode. This will be typical of Control Solutions sensors, Digi International sensors, and various other ZigBee devices.



This page provides API access to the local ZigBee radio.

Dev #	Cfg	Addr 16-bit	Addr 64-bit	Node ID	
1	N	7A D4	00 13 A2 00 40 32 03 6E		☑

When a device is first connected, it will hopefully show up in the Node List. For this to happen, it must be configured with the same stack profile as the BB2-7040 (ZS parameter), and must be set for the same PAN ID, or set to "join any" PAN. Devices configured to use AT mode instead of API mode will also not show up here, but that does not mean you cannot communicate with them. Check the Message Log page to see if your device is sending anything the BB2-7040 can hear.

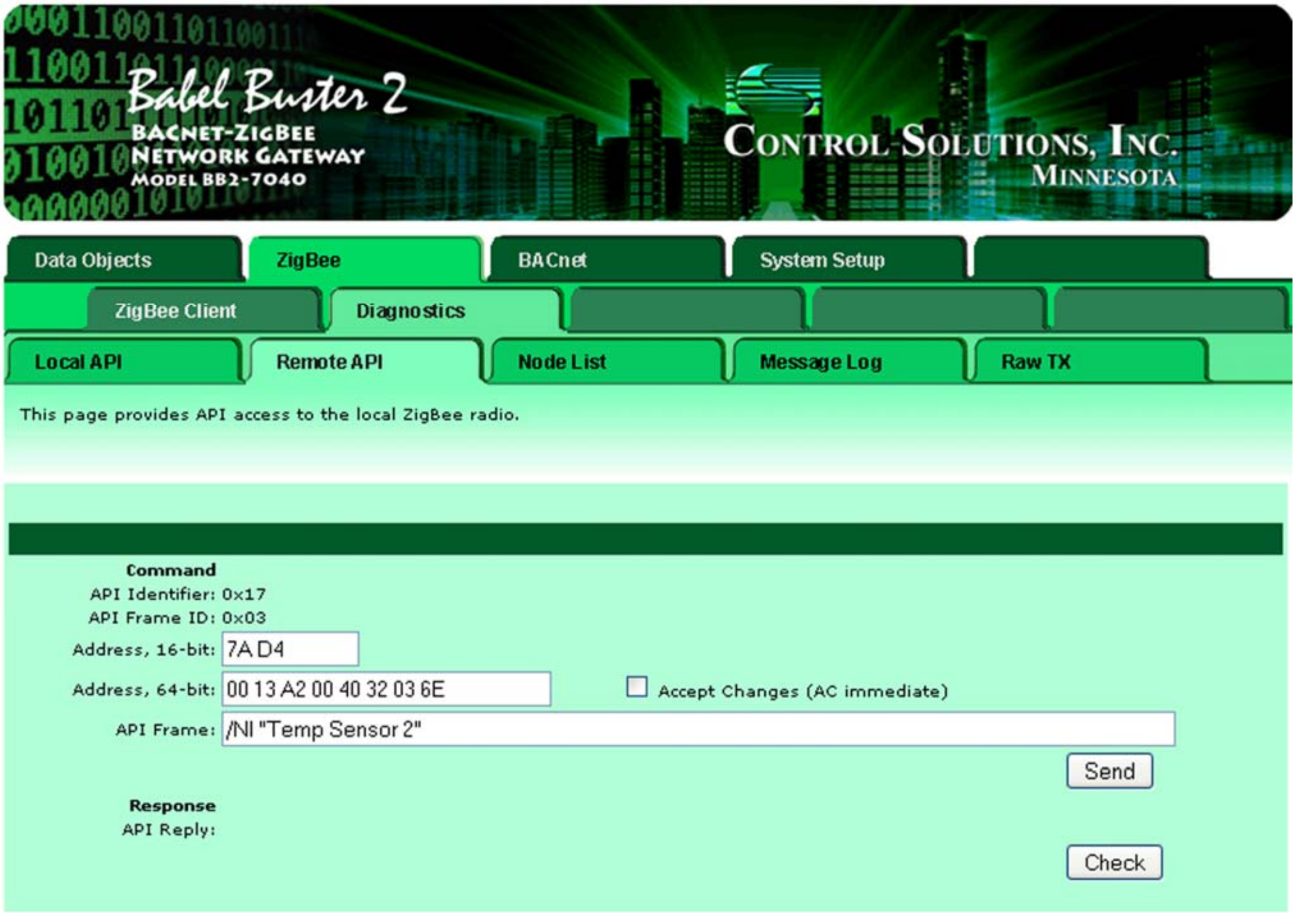


This page provides API access to the local ZigBee radio.

```
1,11:55:52

Up Time     Status    API Packet
1,11:55:03  ----      92 00 13 A2 00 40 2D 37 7A 7C 23 41 01 08 00 06 08 00 00 2B 02 71
1,11:54:59  ----      92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 13 02 B5 00 87
```

The first byte in the packet is always the packet type. The next 8 bytes are the device's 64-bit address. This is always unique per device. It will generally be found on a label or documentation accompanying the device. It is essentially the MAC address of this ZigBee device.

Assuming your device shows up on the Node List, you can check the box at the end of the row and click API Target. This will automatically copy this device's address to the address windows of the Remote API page. Next, go to the Remote API page.



This page provides API access to the local ZigBee radio.

Command

API Identifier: 0x17
API Frame ID: 0x03
Address, 16-bit: 7A D4
Address, 64-bit: 00 13 A2 00 40 32 03 6E
□ Accept Changes (AC immediate)
API Frame: /NI "Temp Sensor 2"

Response

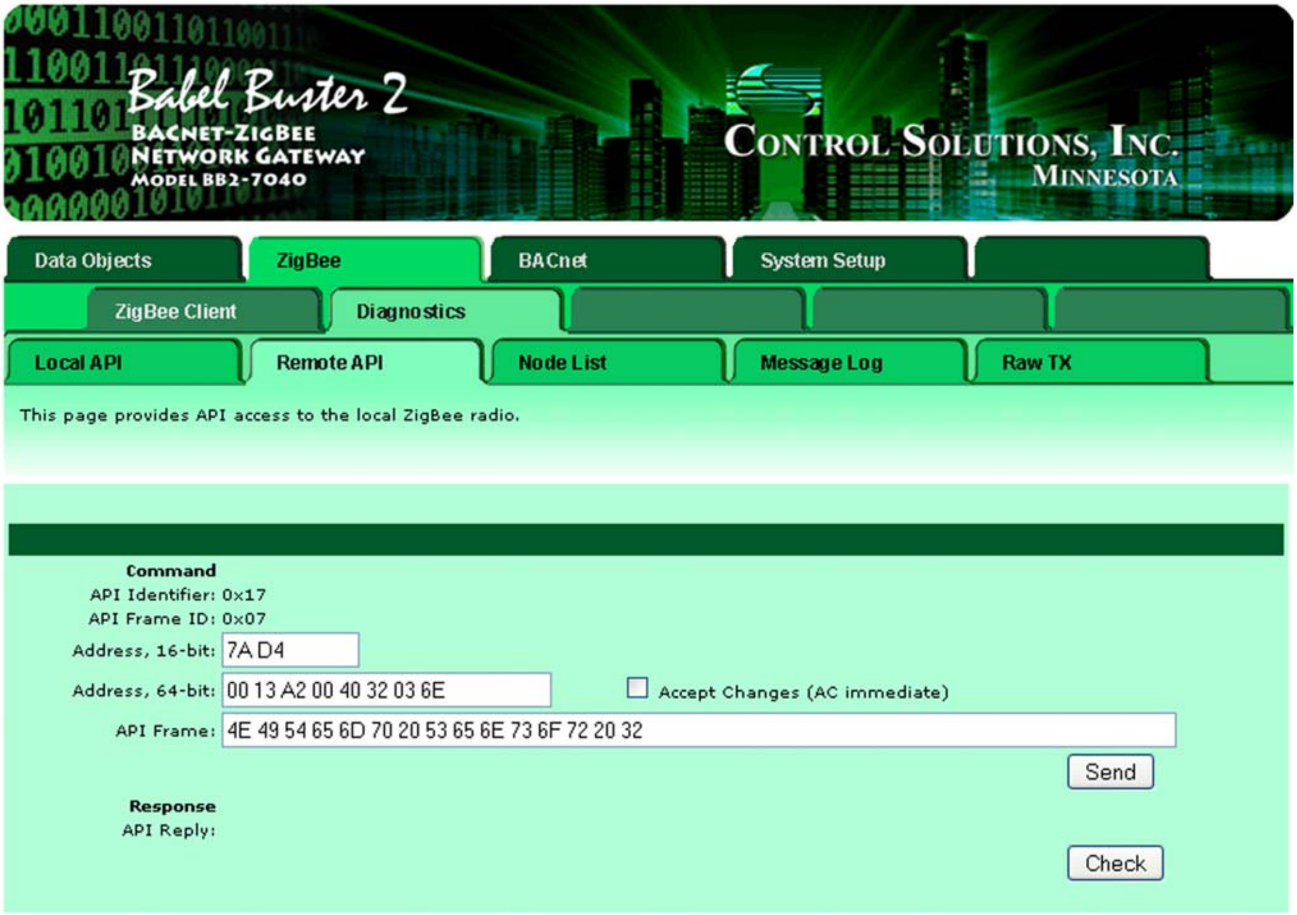
API Reply:

One of several commands you will send to a remote device, assuming it recognizes remote API commands, will be the NI or Network Identifier command. This assigns a name to this device, and after the WR command is sent, it becomes stored in the remote ZigBee device's non-volatile memory.

Commands sent to remote devices ultimately turn into a string of raw binary bytes most often represented as two hexadecimal digits. If you recognize 'FF' as meaning decimal 255, you've got this hex stuff figured out already. If not, go to Wikipedia.com and search for 'hexadecimal'.

The BB2-7040 provides some short cuts in its command syntax to make the commands more user friendly. Any AT command to be sent to the remote device using remote API commands can be typed as /AB where 'AB' is the 2-character AT command. Follow the command with hexadecimal bytes separated by spaces, character strings preceded by '/' (with no embedded spaces), or a string including spaces delimited by quotes.

The above example shows setting the Network Identifier to be "Temp Sensor 2". As soon as you click Send, the hexadecimal interpretation actually sent will be displayed instead, as illustrated below. Initially there will be no reply showing. It takes a little bit of time for a reply to come back most (but not all) of the time.



This page provides API access to the local ZigBee radio.

Command

API Identifier: 0x17

API Frame ID: 0x07

Address, 16-bit: 7A D4

Address, 64-bit: 00 13 A2 00 40 32 03 6E   □ Accept Changes (AC immediate)

API Frame: 4E 49 54 65 6D 70 20 53 65 6E 73 6F 72 20 32

Response

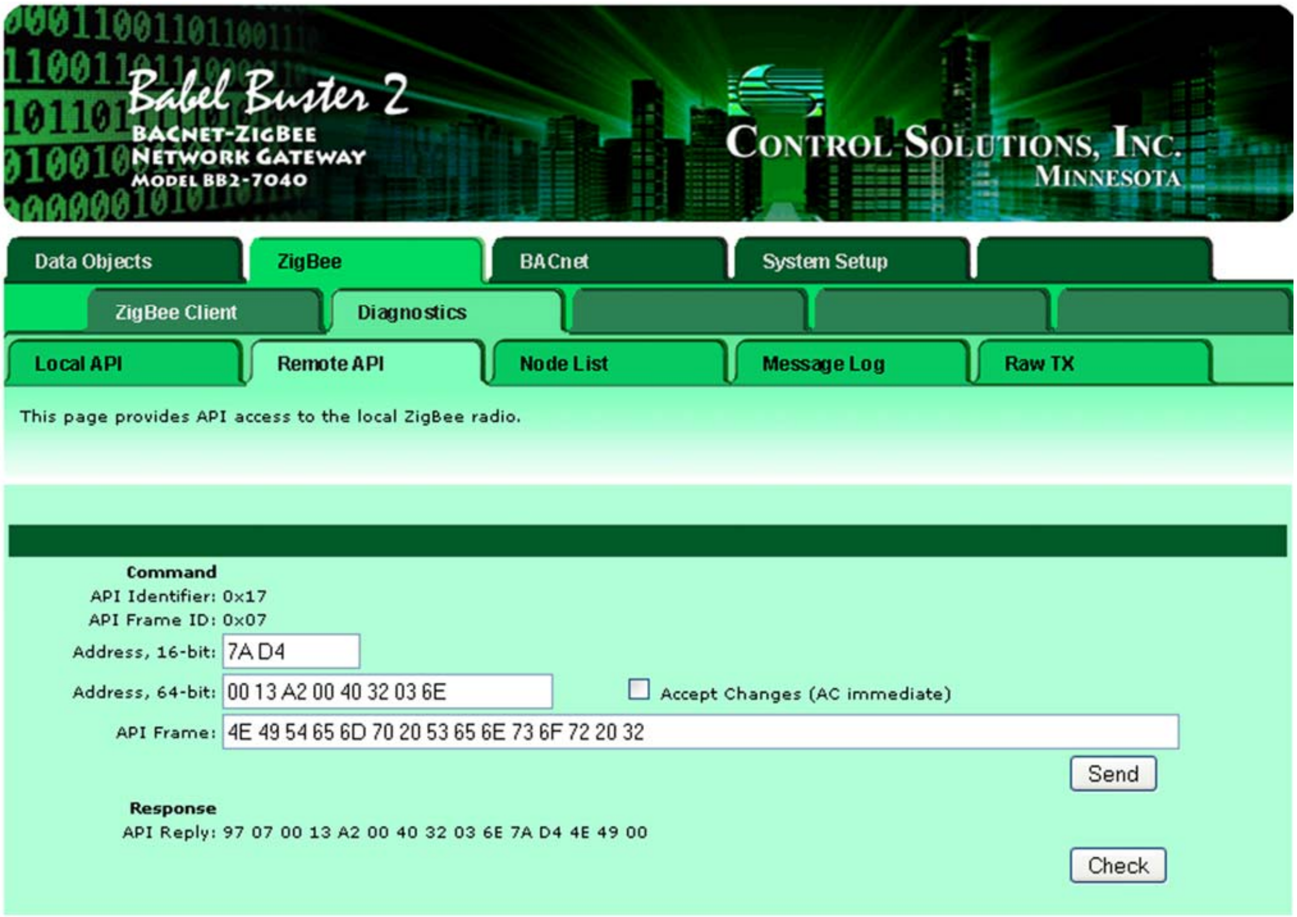
API Reply:

Once Send has been clicked and you see the confirmation of the packet actually sent in hexadecimal form, click Check to see if you have received a reply yet. The reply will be displayed as illustrated below when received.

When a device is in sleep mode, it is important to note that your Send will not be received until the end of the device's sleep period. As first shipped, devices are often not configured to sleep. But once configured, you will have to wait a while for your response. While it may seem inconvenient, the longer you have to wait for each response, the longer your battery life is going to be since you now know the sleep period is long.

The reply received in response to the above command is illustrated below. The first several bytes are confirmation of the address of the device. The last three bytes are of interest. They should be the 2-character AT command followed by a status byte. A status byte of '00' means the command was executed ok. A nonzero value means something was wrong, such as invalid parameter, invalid command, etc.

This short example is simply an overview of seeing a ZigBee device come online for the first time. The /NI command is an example of only one of several commands you will need to send to a typical sensor before it is fully configured. More discussion about configuration is yet to come.



This page provides API access to the local ZigBee radio.

Command

API Identifier: 0x17
API Frame ID: 0x07
Address, 16-bit: 7A D4
Address, 64-bit: 00 13 A2 00 40 32 03 6E
□ Accept Changes (AC immediate)
API Frame: 4E 49 54 65 6D 70 20 53 65 6E 73 6F 72 20 32

Response

API Reply: 97 07 00 13 A2 00 40 32 03 6E 7A D4 4E 49 00

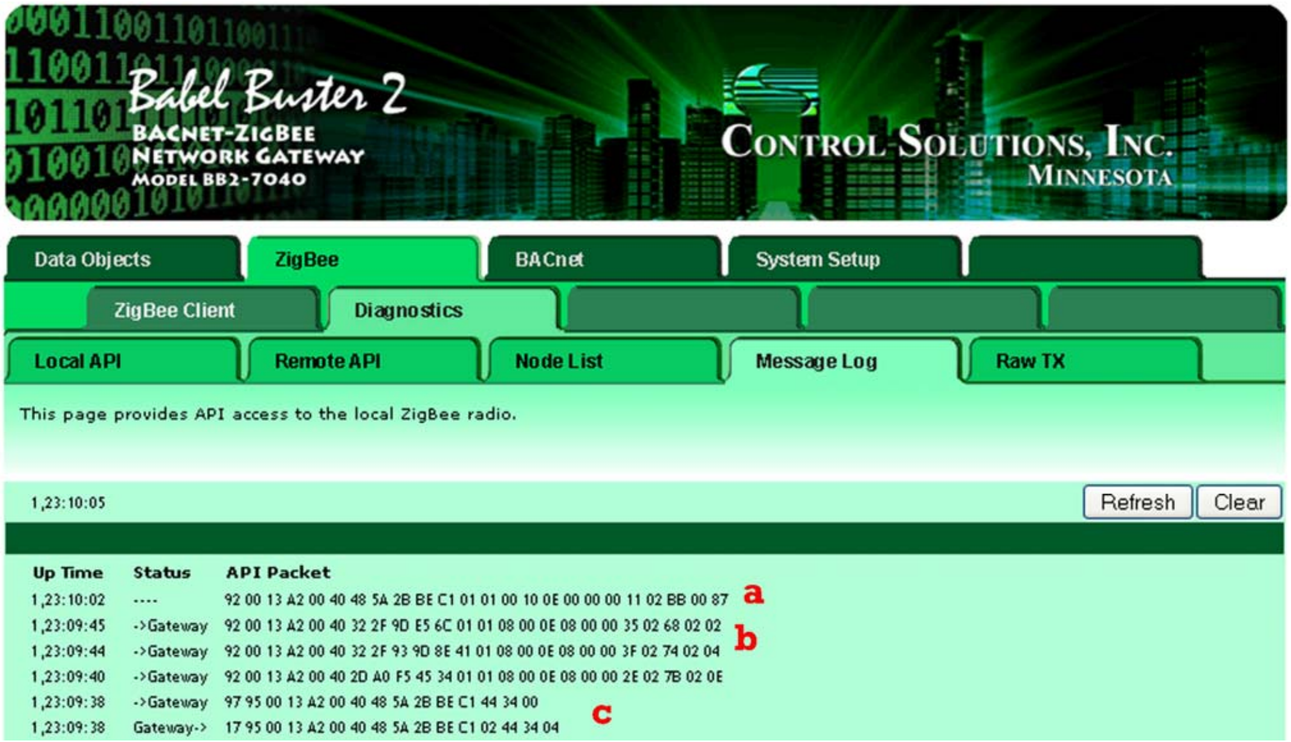
Status codes for XBee endpoint devices using API mode are as follows:

00	No error
01	Error (unspecified)
02	Invalid Command
03	Invalid Parameter
04	Remote command transmission failed.

4 Creating a Data Parse Mask

The data parse mask may have been provided for you in a related document pertaining to a specific endpoint, or in an appendix of this document. If you need to create a new data parse mask for a new device, start by getting the device online and sending its data packets to the BB2-7040. Correlate the packets observed with the manufacturers documentation to begin to recognize its data.

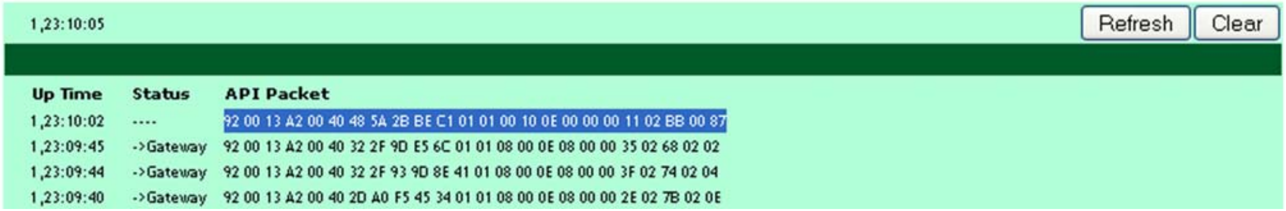
The screen shot below shows several packets observed in the Message Log. The line marked “a” was received with no rule recognizing that packet. The packets marked “b” were received and recognized by rules with data parse masks. The packets at “c” show the gateway sending a command to a device and receiving its reply.



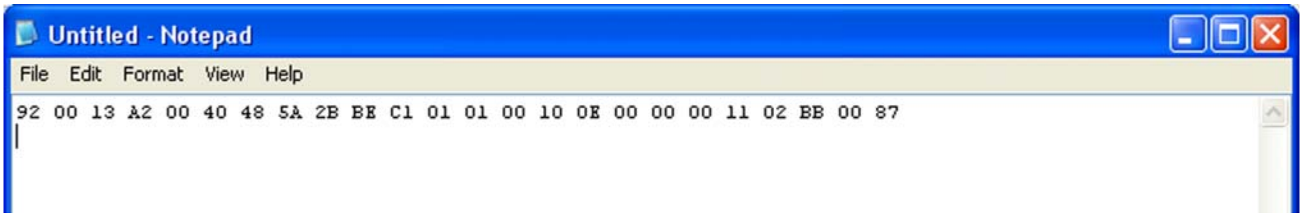
This page provides API access to the local ZigBee radio.

```
1,23:10:05

Up Time     Status      API Packet
1,23:10:02  ----        92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87
1,23:09:45  ->Gateway   92 00 13 A2 00 40 32 2F 9D E5 6C 01 01 08 00 0E 08 00 00 35 02 68 02 02
1,23:09:44  ->Gateway   92 00 13 A2 00 40 32 2F 93 9D 8E 41 01 08 00 0E 08 00 00 3F 02 74 02 04
1,23:09:40  ->Gateway   92 00 13 A2 00 40 2D A0 F5 45 34 01 01 08 00 0E 08 00 00 2E 02 7B 02 0E
1,23:09:38  ->Gateway   97 95 00 13 A2 00 40 48 5A 2B BE C1 44 34 00
1,23:09:38  Gateway->   17 95 00 13 A2 00 40 48 5A 2B BE C1 02 44 34 04
```

Let us copy the unrecognized packet and paste into a program like Notepad.

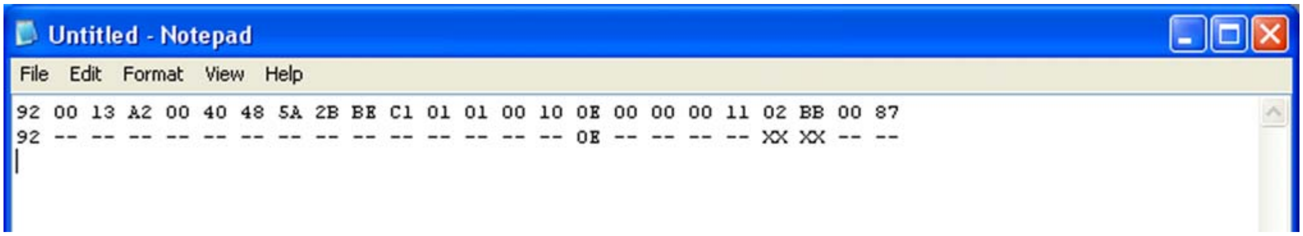


```
1,23:10:05

Up Time     Status      API Packet
1,23:10:02  ----        92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87
1,23:09:45  ->Gateway   92 00 13 A2 00 40 32 2F 9D E5 6C 01 01 08 00 0E 08 00 00 35 02 68 02 02
1,23:09:44  ->Gateway   92 00 13 A2 00 40 32 2F 93 9D 8E 41 01 08 00 0E 08 00 00 3F 02 74 02 04
1,23:09:40  ->Gateway   92 00 13 A2 00 40 2D A0 F5 45 34 01 01 08 00 0E 08 00 00 2E 02 7B 02 0E
```



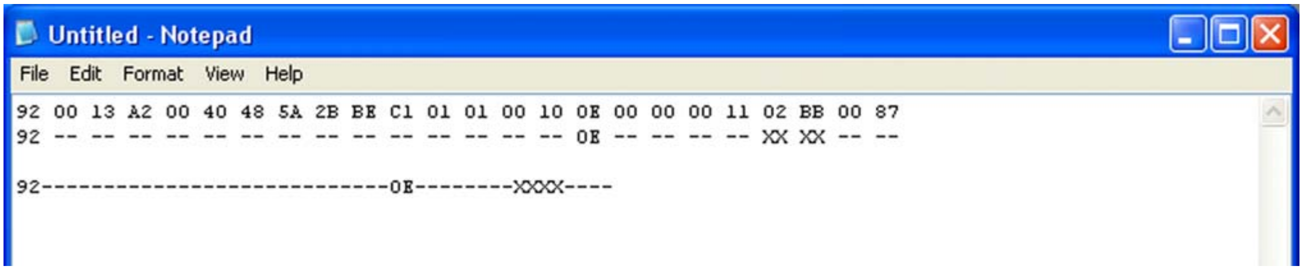
```
92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87
```

We now begin to construct the data parse mask based on documentation provided by the manufacturer. This example happens to be a Digi XBee sensor. We know that the first byte of the API packet should always be (hex) 92. We will not check the address here since the device handling should do that for us. We can therefore treat the address bytes as “don’t care” at this level, making device substitution easier at a later time. We know that the channel mask should always be hex) 0E. We know that the next to last pair of bytes is a data point of interest, in this case raw temperature reading.



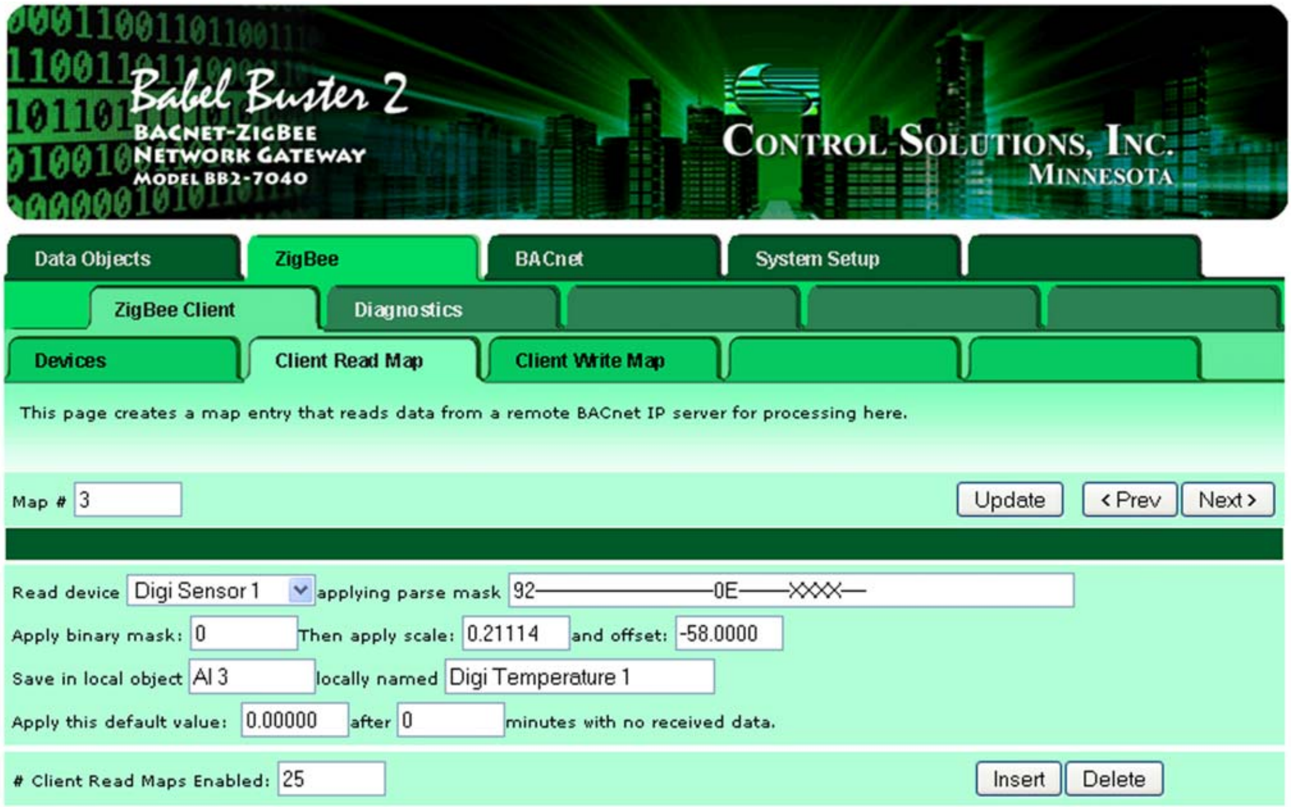
```
92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87
92 -- -- -- -- -- -- -- -- -- -- -- -- -- -- 0E -- -- -- -- XX XX -- --
```

After typing mask characters out in a one to one correlation with the packet, make another copy of that line and then eliminate all embedded blanks.



```
92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 11 02 BB 00 87
92 -- -- -- -- -- -- -- -- -- -- -- -- -- -- 0E -- -- -- -- XX XX -- --

92----------------------------0E--------XXXX----
```

Finally, copy the data parse mask into the Client Read Map, select the device from the device list, add scale and offset, select a local object as the destination of the data, and click Update. The BB2-7040 will now be listening for this data.



This page creates a map entry that reads data from a remote BACnet IP server for processing here.

Map # 3

Read device Digi Sensor 1 applying parse mask 92----------------------------0E--------XXXX----

Apply binary mask: 0 Then apply scale: 0.21114 and offset: -58.0000

Save in local object AI 3 locally named Digi Temperature 1

Apply this default value: 0.00000 after 0 minutes with no received data.

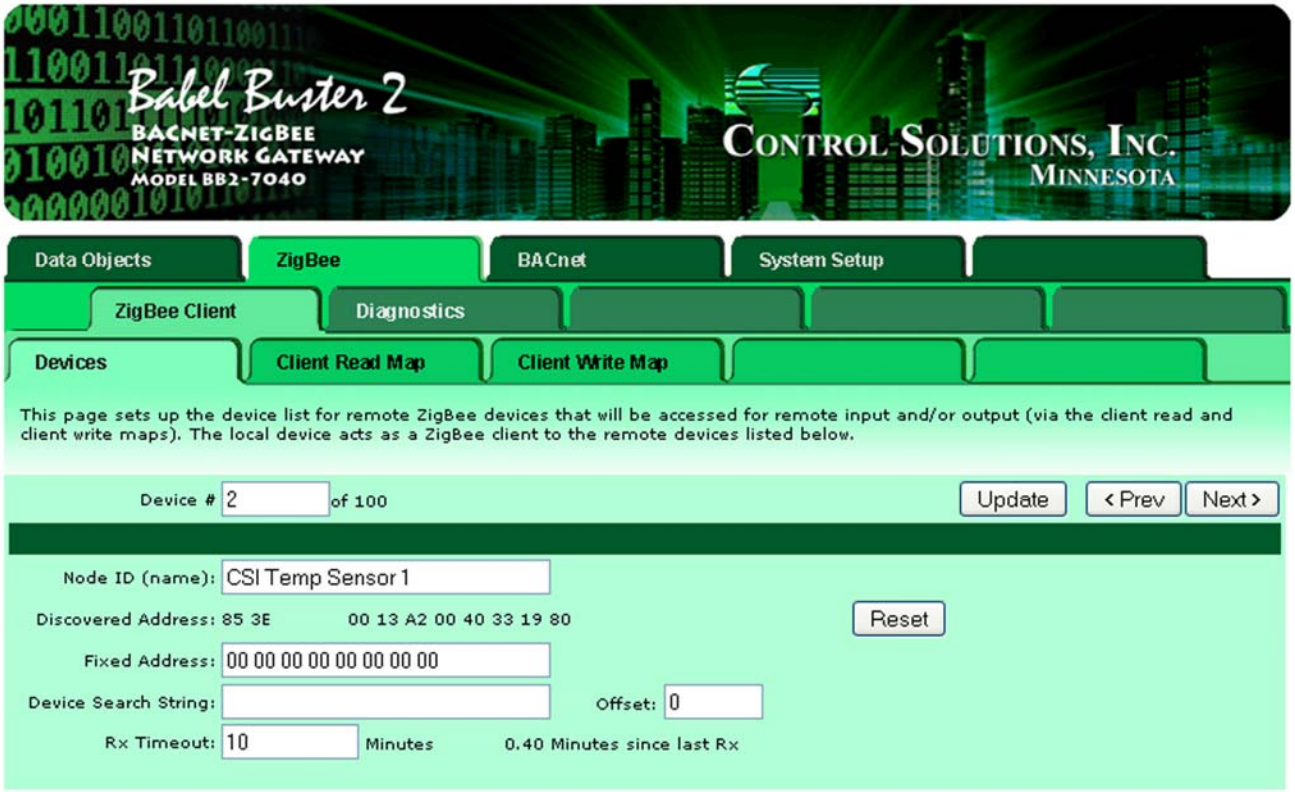
Client Read Maps Enabled: 25

See Section 6 for further detail on constructing a Data Parse Mask.

5 Setting Up the ZigBee Device List

There are three ways to get the BB2-7040 to recognize a device. Devices must be recognized one way or another so that we know the right data is ending up in the right BACnet objects.

5.1 Network Identifier



This page sets up the device list for remote ZigBee devices that will be accessed for remote input and/or output (via the client read and client write maps). The local device acts as a ZigBee client to the remote devices listed below.

Device # 2 of 100

Node ID (name): CSI Temp Sensor 1

Discovered Address: 85 3E 00 13 A2 00 40 33 19 80

Fixed Address: 00 00 00 00 00 00 00 00

Device Search String: Offset: 0

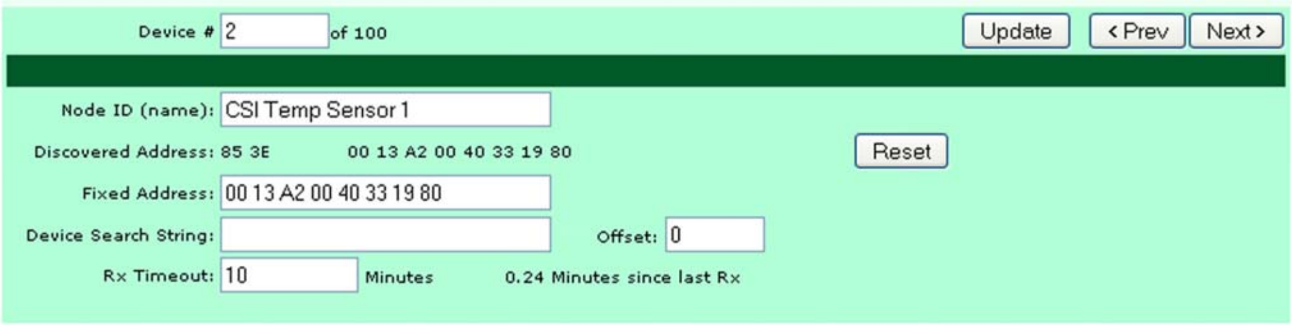
Rx Timeout: 10 Minutes 0.40 Minutes since last Rx

The Network Identifier is a 20-character ASCII string naming the device. The name needs to be unique on the PAN. The combination of a valid ASCII string and non-zero Rx Timeout value signal your desire to have the BB2-7040 try to find this device by Network Identifier. The name will be broadcast, asking any ZigBee device matching that name to reply with its 64-bit address. From that point forward, the BB2-7040 will use the 64-bit address that was reported.

IMPORTANT: Network Identifier might be considered the ideal approach since you can replace a sensor by naming that sensor rather than keep track of its 64-bit address. However, the BB2-7040 will keep looking for that name until it finds it. An undesired side effect is that every time the ZigBee coordinator transmits the command looking for the name, it wakes up all sleeping devices on its PAN. If the name is not found for any reason, the continued retries that continue to wake up all sensors will result in premature end of battery life in all of the sensors.

You can avoid this potential battery life problem by using the fixed 64-bit address as outlined in the following section. While it is less convenient in the sense that you need to know its 64-bit address, it is more convenient in the sense that you do not have to take any steps to assign it a name. You only need to assign a PAN ID and set the timing parameters. The ZigBee device will be immediately recognized the first time it transmits because the 64-bit address is always included in every transmission. More importantly, the ZigBee coordinator does not waste any device’s battery life by searching the network for names.

5.2 Fixed 64-bit Address



Device # 2 of 100

Node ID (name): CSI Temp Sensor 1

Discovered Address: 85 3E 00 13 A2 00 40 33 19 80

Fixed Address: 00 13 A2 00 40 33 19 80

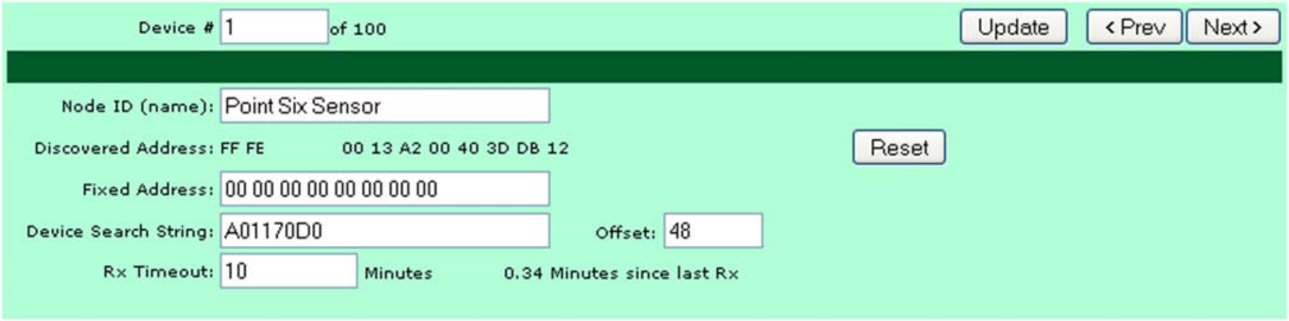
Device Search String:

Offset: 0

Rx Timeout: 10 Minutes 0.24 Minutes since last Rx

In the event setting the Network Identifier is not possible, but the 64-bit address is known, you can enter that fixed address along with non-zero Rx Timeout. The name becomes a local reference only. The 64-bit address will be automatically used by the BB2-7040 any time this device is referenced.

5.3 Search String



Device # 1 of 100

Node ID (name): Point Six Sensor

Discovered Address: FF FE 00 13 A2 00 40 3D DB 12

Fixed Address: 00 00 00 00 00 00 00 00

Device Search String: A01170D0

Offset: 48

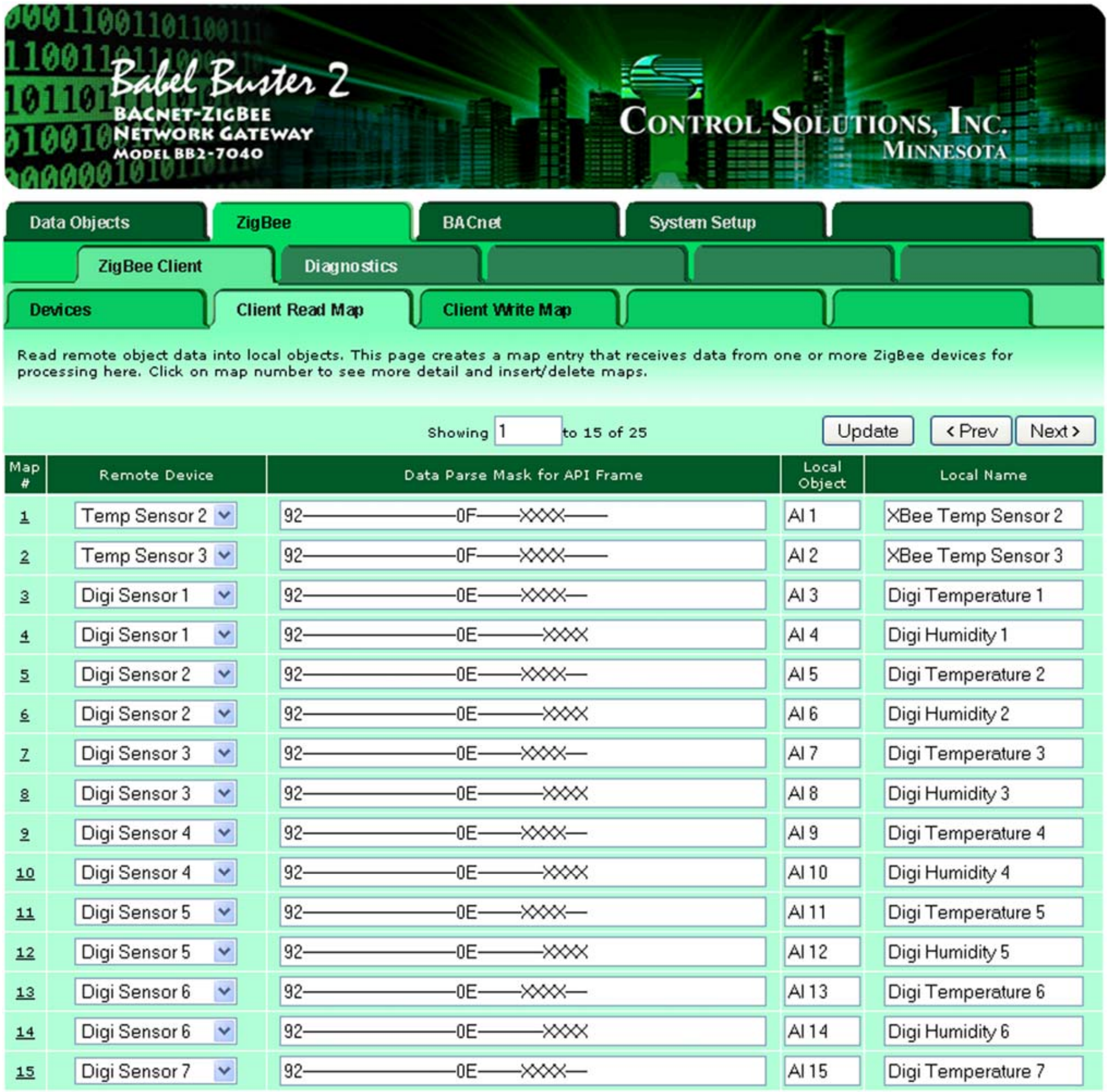
Rx Timeout: 10 Minutes 0.34 Minutes since last Rx

A method used by some ZigBee devices for identification is including an ASCII string, such as device serial number, in every data stream sent by that device. When this method is used, enter the string to search for and the byte offset indicating the location in the packet where the search should begin. It is expected that the search string will always begin in a fixed location within the packet. A non-zero Rx Timeout must also be entered to signal that this device entry is now valid.

The Rx Timeout is the amount of time that must elapse with no packet received from the device after which the mapped BACnet object will be flagged with a Fault status, and the object's reliability code set to indicate device timeout.

6 Setting Up ZigBee Client Read and Write Maps

6.1 ZigBee Client Read Maps



Read remote object data into local objects. This page creates a map entry that receives data from one or more ZigBee devices for processing here. Click on map number to see more detail and insert/delete maps.

Showing 1 to 15 of 25

Map #	Remote Device	Data Parse Mask for API Frame	Local Object	Local Name
1	Temp Sensor 2	92————————0F———XXXX———	AI 1	XBee Temp Sensor 2
2	Temp Sensor 3	92————————0F———XXXX———	AI 2	XBee Temp Sensor 3
3	Digi Sensor 1	92————————0E———XXXX——	AI 3	Digi Temperature 1
4	Digi Sensor 1	92————————0E————XXXX	AI 4	Digi Humidity 1
5	Digi Sensor 2	92————————0E———XXXX——	AI 5	Digi Temperature 2
6	Digi Sensor 2	92————————0E————XXXX	AI 6	Digi Humidity 2
7	Digi Sensor 3	92————————0E———XXXX——	AI 7	Digi Temperature 3
8	Digi Sensor 3	92————————0E————XXXX	AI 8	Digi Humidity 3
9	Digi Sensor 4	92————————0E———XXXX——	AI 9	Digi Temperature 4
10	Digi Sensor 4	92————————0E————XXXX	AI 10	Digi Humidity 4
11	Digi Sensor 5	92————————0E———XXXX——	AI 11	Digi Temperature 5
12	Digi Sensor 5	92————————0E————XXXX	AI 12	Digi Humidity 5
13	Digi Sensor 6	92————————0E———XXXX——	AI 13	Digi Temperature 6
14	Digi Sensor 6	92————————0E————XXXX	AI 14	Digi Humidity 6
15	Digi Sensor 7	92————————0E———XXXX——	AI 15	Digi Temperature 7

A note about "reading" ZigBee devices: ZigBee is optimized for low power battery operated sensors that sleep most of the time. This means the traditional "polling" of the remote device cannot be done with ZigBee as it can with Modbus or other remote hard wired devices. Most Babel Buster family members are set up to periodically poll remote devices. While the same terms such as "read map" are used in the BB2-7040, the definition of "read" really means "prepared to receive". The read maps found here never initiate any radio traffic, they only respond to receiving radio transmissions from remote ZigBee devices.

Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only read data from remote devices. Go to the Client Write Map to write data to those devices. The full parameter set is different for read versus write.

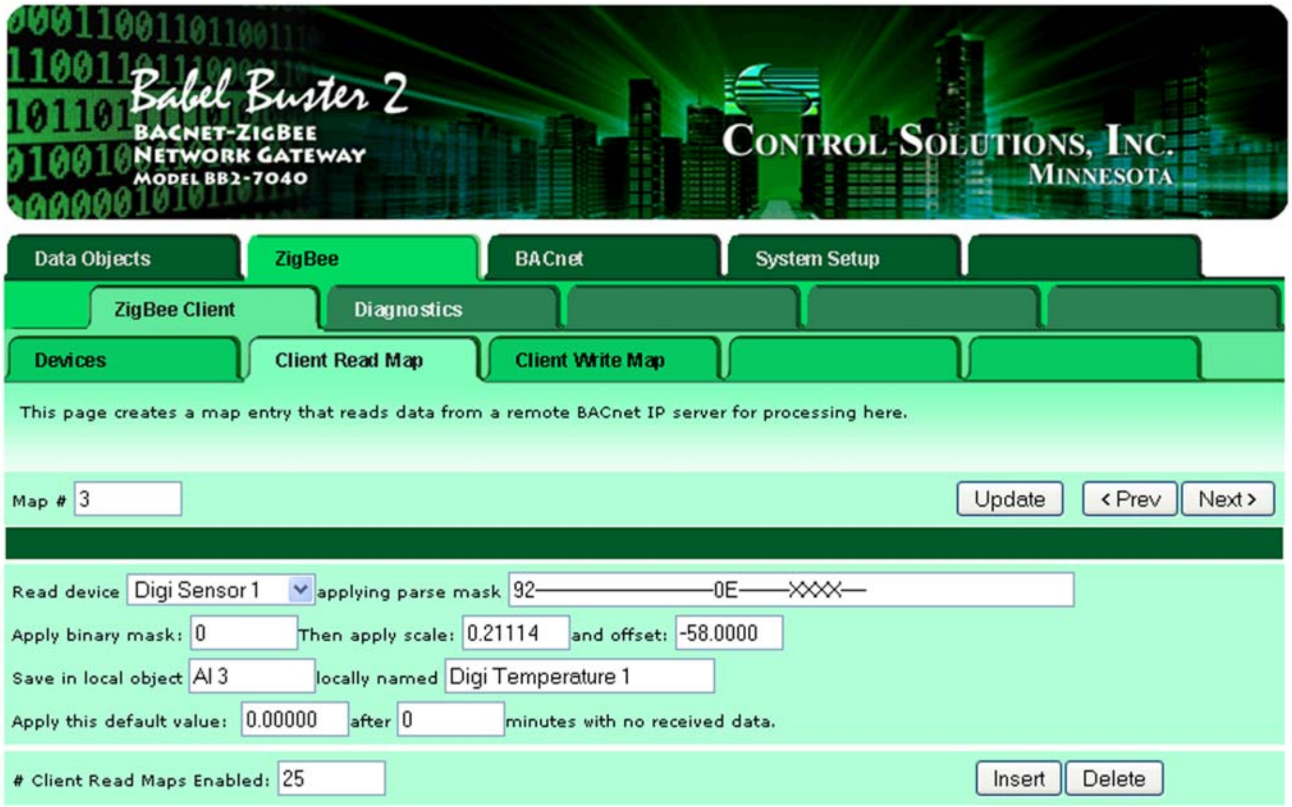
An abbreviated version of a list of maps is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

For each remote object to be read, select the device from the list, and provide a data parse map. Select a local object where the resulting data should be placed. Details about the data parse mask may be found on the extended map view page by clicking the map number in the left most column.

Local Object is internally a coded number consisting of BACnet object type multiplied by 1000, then added to the object number starting from #1. These are translated into abbreviations that are easy to interpret on the web page as follows:

AI n = Analog Input #n
AO n = Analog Output #n
AV n = Analog Value #n
BI n = Binary Input #n
BO n = Binary Output #n
BV n = Binary Value #n
MI n = Multi-state Input #n
MO n = Multi-state Output #n
MV n = Multi-state Value #n

Object numbers start at #1. The maximum available number varies by object type, and these limits may be found on the System Capacities link from the home/index page (click graphic at top).



This page creates a map entry that reads data from a remote BACnet IP server for processing here.

Map # 3

Read device Digi Sensor 1 applying parse mask 92————————0E——XXXX—

Apply binary mask: 0 Then apply scale: 0.21114 and offset: -58.0000

Save in local object AI 3 locally named Digi Temperature 1

Apply this default value: 0.00000 after 0 minutes with no received data.

Client Read Maps Enabled: 25

Clicking map number in the first column of the tabular map list gets you to this view. Rule number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

For each remote object to be received, select an object name from the list and provide a data parse mask (see below). The names in the device list are defined in the Devices page.

When the remote data is received, data may be manipulated before being written to the local object. First, binary data is parsed from the received packet based on the data parse mask. Second, the raw data is masked by ANDing with the binary mask if non-zero. Next, the value will be converted to floating point, and the value will be multiplied by the scale factor, then the offset is added. The final result is written to the local object number given (with conversion back to binary or integer as applicable). The name is optional and used only for display purposes.

The periodic poll time determines how often the remote object will be read. This number, if nonzero, will override the default poll time given in the Devices page for the remote device being read.

The default value will be stored into the local object after the timeout period elapses with no data received. Setting the count to zero will disable the default, and the object will retain the most recent value obtained.

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote device effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

DATA PARSE MASK

As of v3.01, the BB2-7040 is parepared to parse raw binary (integer) data from received ZigBee packets. Most ZigBee devices have very limited resources (keeping power consumption low), and therefore will not support floating point. The BB2-7040 is prepared to accept binary/integer data and scale and convert as necessary to produce analog floating point data. In the event devices are found which do use IEEE754 floating point, support for that format will be added in a later revision (contact tech support).

Getting acquainted with a sensor for the first time will require some experimenting. You must start by configuring and commissioning the ZigBee sensor, causing it to transmit periodic sensor data samples. When data packets start to show up, between inspection of the packets and review of sensor documentation, you can begin to construct a data parsing mask that will result in getting the data of interest into BACnet objects. When a ZigBee transmission is received, its data packet will be parsed into as many objects as match the parsing mask. There may be multiple parsing masks for any given remote device.

An example of a packet received from a remote XBee sensor appears on the first line below:

```
92 00 13 A2 00 40 48 5A 2A 39 A3 01 01 00 10 0E 00 00 00 0F 02 AB 00 89
92 -- -- -- -- -- -- -- -- -- -- -- 01 -- -- 0E -- -- -- -- -- -- XX XX
```

The easiest way to construct a mask for this type of packet is copy it to a text editor, then begin typing mask characters on the line below it as illustrated above. Once a complete mask is assembled, remove the embedded blanks, and copy the mask that appears below into the mask window on this page (above).

```
92----------------------01----0E------------XXXX
```

The data mask fields are always 2-character pairs (with the exception of “[nn]”), and may be any of the following:

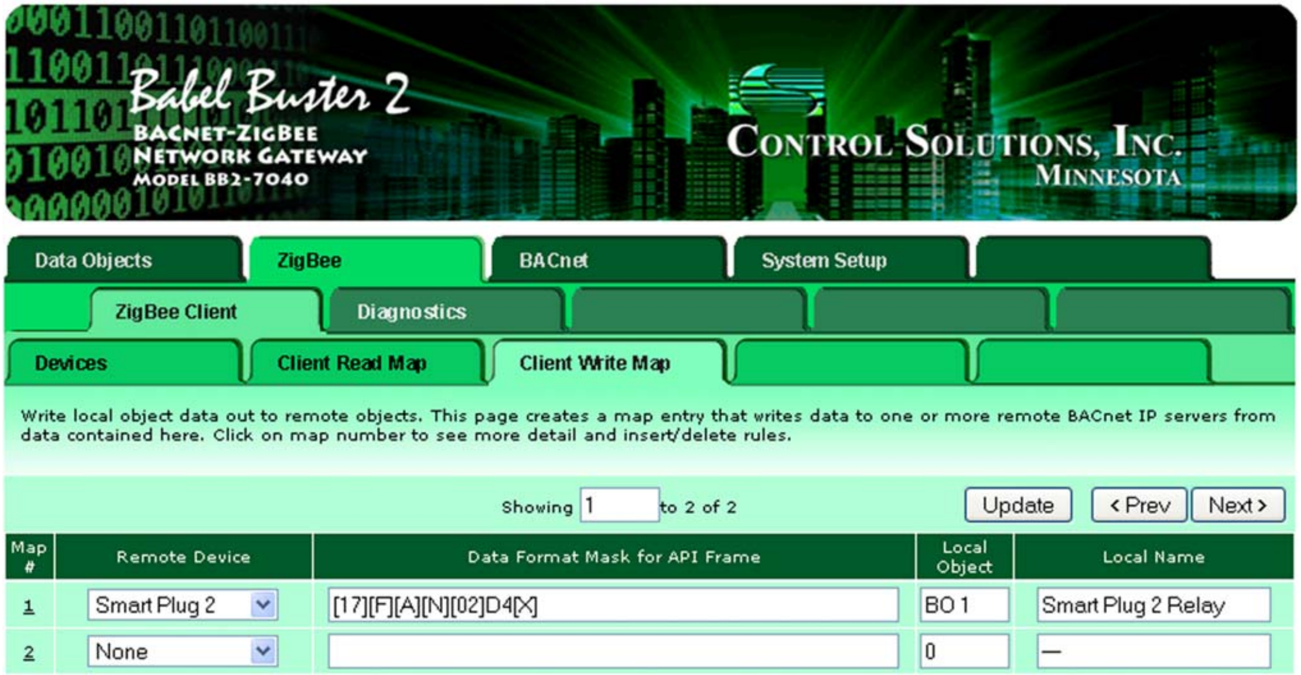
"--" is a don't care byte. It must be two hyphens for 2 hex digits, and must be in pairs as only even bytes can be skipped.
"NN" is an exact match value where N is a hex digit; for example if the byte must be 0x92, we put 92 in the mask.
"XX" is a hex value field to be parsed; field may be XX to XXXX to XXXXXXXX for 8, 16, 32-bit values respectively
"HH" is a hex digit that is ASCII encoded in the data stream. If ASCII is used, HHHH would be equivalent to XX.
"[nn]" means skip nn (decimal) bytes; [04] is the same as 8 hyphens.

“TT” needs to show up in pairs (“TTTT” in the mask) and is unique to Control Solutions ZB25-T or ZB25-TH sensors. This mask will parse the thermistor reading and linearize it to a normalized reading in degrees Celsius x 100. Refer to Appendix A for examples specific to the Control Solutions ZB25 series sensors.

The first byte in the above example is the API frame type code. The next 10 bytes are the 64-bit and 16-bit addresses of the remote device the packet was received from. Do not force matching on these bytes. These are automatically checked against the addresses found in the device list. You only need to select a device by name.

The example above says we must see 0x92 in the first byte of the API frame, and values 0x01 and 0x0E where shown. We are going to extract data from the positions indicted by XXXX. In this case, we are extracting the current (amps) level from a Digi International Smart Plug. Visit the BB2-7040 product page at www.csimn.comfor links to Digi documentation.

6.2 ZigBee Client Write Maps



Write local object data out to remote objects. This page creates a map entry that writes data to one or more remote BACnet IP servers from data contained here. Click on map number to see more detail and insert/delete rules.

Showing 1 to 2 of 2

Map #	Remote Device	Data Format Mask for API Frame	Local Object	Local Name
1	Smart Plug 2	[17][F][A][N][02]D4[X]	BO 1	Smart Plug 2 Relay
2	None		0	—

A note about "writing" ZigBee devices: ZigBee is optimized for low power battery operated sensors that sleep most of the time. This means that any attempt to "write" really means queuing up a message for some time later. Each write map includes a timeout. The meaning of timeout is simply that the local BACnet object whose data is supposed to be transmitted will have its reliability code set showing the timeout, and the BB2-7040 will retry. The timeout should always be set longer than the sleep period of the remote ZigBee device intended to receive this data.

Map number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only write data to remote devices. Go to the Client Read Map to read data from those devices. The full parameter set is different for read versus write.

An abbreviated version of a list of maps is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

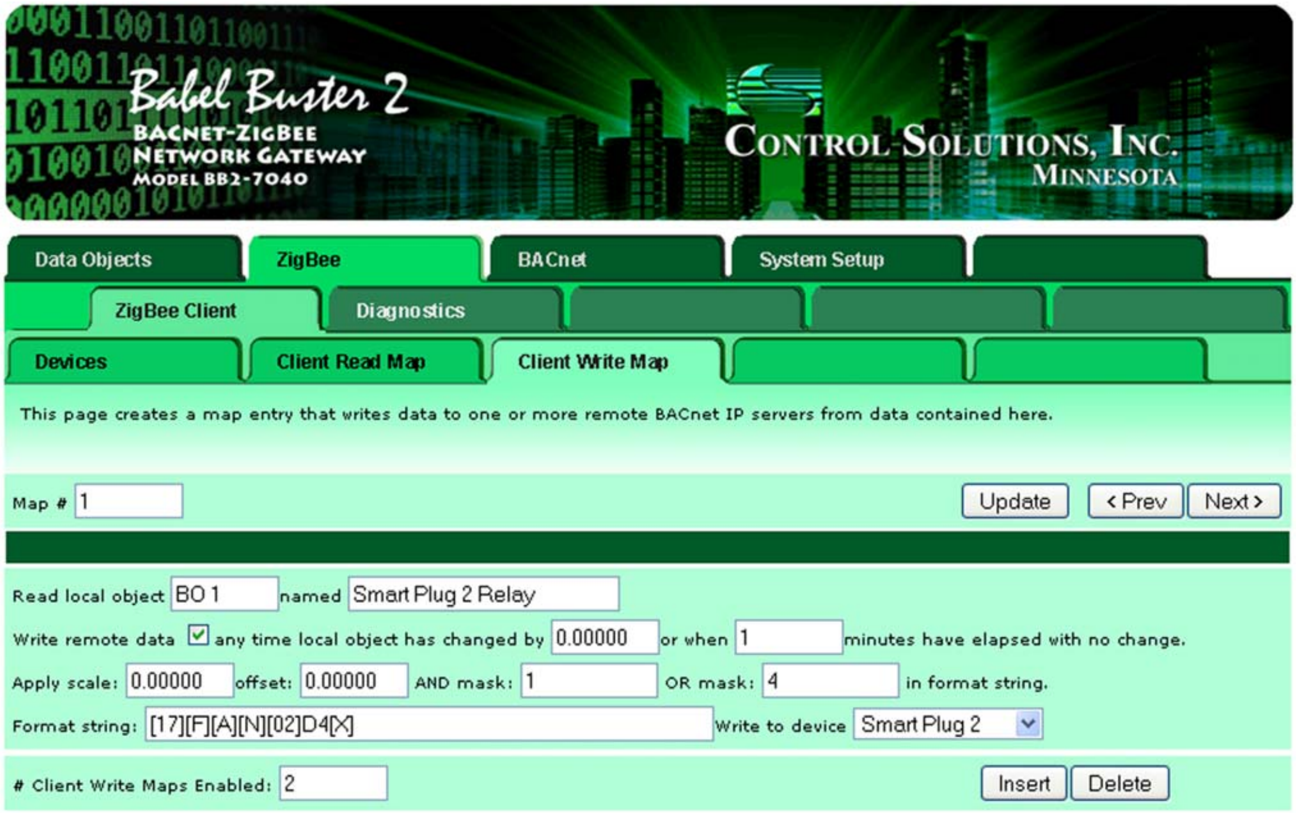
For each remote object to be written, select the device to write and provide a data formatting mask. Select the local object from which data will be written to the remote device. The names in the device list are defined in the Devices page. Details about the data format mask may be found on the extended map view page by clicking the map number in the left most column.

Selecting "none" for remote device effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

Local Object is internally a coded number consisting of BACnet object type multiplied by 1000, then added to the object number starting from #1. These are translated into abbreviations that are easy to interpret on the web page as follows:

AI n = Analog Input #n
AO n = Analog Output #n
AV n = Analog Value #n
BI n = Binary Input #n
BO n = Binary Output #n
BV n = Binary Value #n
MI n = Multi-state Input #n
MO n = Multi-state Output #n
MV n = Multi-state Value #n

Object numbers start at #1. The maximum available number varies by object type, and these limits may be found on the System Capacities link from the home/index page (click graphic at top).



This page creates a map entry that writes data to one or more remote BACnet IP servers from data contained here.

Map # 1

Read local object BO 1 named Smart Plug 2 Relay

Write remote data ☑ any time local object has changed by 0.00000 or when 1 minutes have elapsed with no change.

Apply scale: 0.00000 offset: 0.00000 AND mask: 1 OR mask: 4 in format string.

Format string: [17][F][A][N][02]D4[X] Write to device Smart Plug 2

Client Write Maps Enabled: 2

Clicking the map number in the first column of the tabular map list gets you to this view. Rule number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

The local object data may be written periodically, or when it changes, or both. To send upon change (send on delta), check the check box and enter the amount by which the local object must change before being written to the remote device. To guarantee that the remote object will be written at least occasionally even if the data does not change, enter a time value in minutes for periodic write.

Data from the local object may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it. The data is then converted to integer/binary in preparation for transmission to the ZigBee device. After conversion to integer, the data is ANDed with the AND mask, and then ORed with the OR mask. The resulting data is then inserted into the format string (see below).

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote device effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

DATA FORMAT MASK

As of v3.01, the BB2-7040 is parepared to format raw binary (integer) data transmitted in ZigBee packets. Most ZigBee devices have very limited resources (keeping power consumption low), and therefore will not support floating point. The BB2-7040 is prepared to send binary/integer data and scale and convert as necessary to convert from analog floating point data. In the event devices are found which do use IEEE754 floating point, support for that format will be added in a later revision (contact tech support).

An example of a format mask is as follows:

[17][F][A][N][02]D4[X]

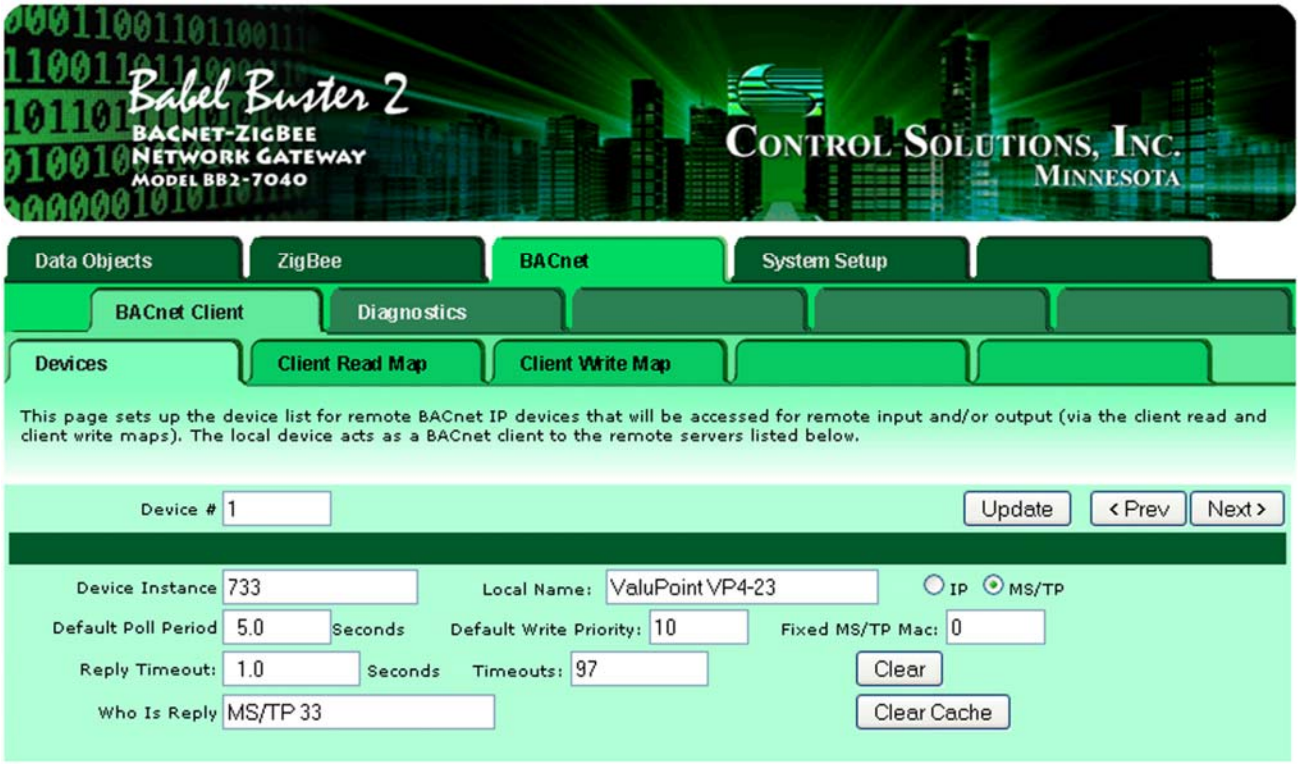
For this particular example, we select a device from the list which names a Digi International Smart Plug. Further, we use a value of 1 for the AND mask, and 4 for the OR mask. The result of this configuration is that we will send D4=4 or D4=5 to switch the Smart Plug's relay on and off. In our example, we are taking data from a local Binary Output object. The result is that we now have BACnet control of the Smart Plug's relay being operated via ZigBee.

The fields available for a format mask are as follows:

[nn] is interpreted as a hex number producing 1 byte in the transmit packet
[F] is replaced with the next auto-generated frame ID as 1 byte, hex 0x0F as a number must be [0F]
[A] is replaced with the 64-bit address (as 8 bytes) of the device named in the map
[N] is replaced with 16-bit address (as 2 bytes) of the device named in the map
[X] is replaced with object data producing one byte (8-bit) from the value
[XX] is replaced with object data producing 2 bytes (16-bit)
[XXXX] is replaced with object data producing 4 bytes (32-bit) - data is big endian as required by ZigBee
All other ASCII characters are copied verbatim with their ASCII code placed in the respective byte position in the packet.

7 Setting Up BACnet

7.1 BACnet Device List



This page sets up the device list for remote BACnet IP devices that will be accessed for remote input and/or output (via the client read and client write maps). The local device acts as a BACnet client to the remote servers listed below.

Device # 1

Device Instance 733 | Local Name: ValuPoint VP4-23 | ○ IP ● MS/TP

Default Poll Period 5.0 Seconds | Default Write Priority: 10 | Fixed MS/TP Mac: 0

Reply Timeout: 1.0 Seconds | Timeouts: 97

Who Is Reply MS/TP 33

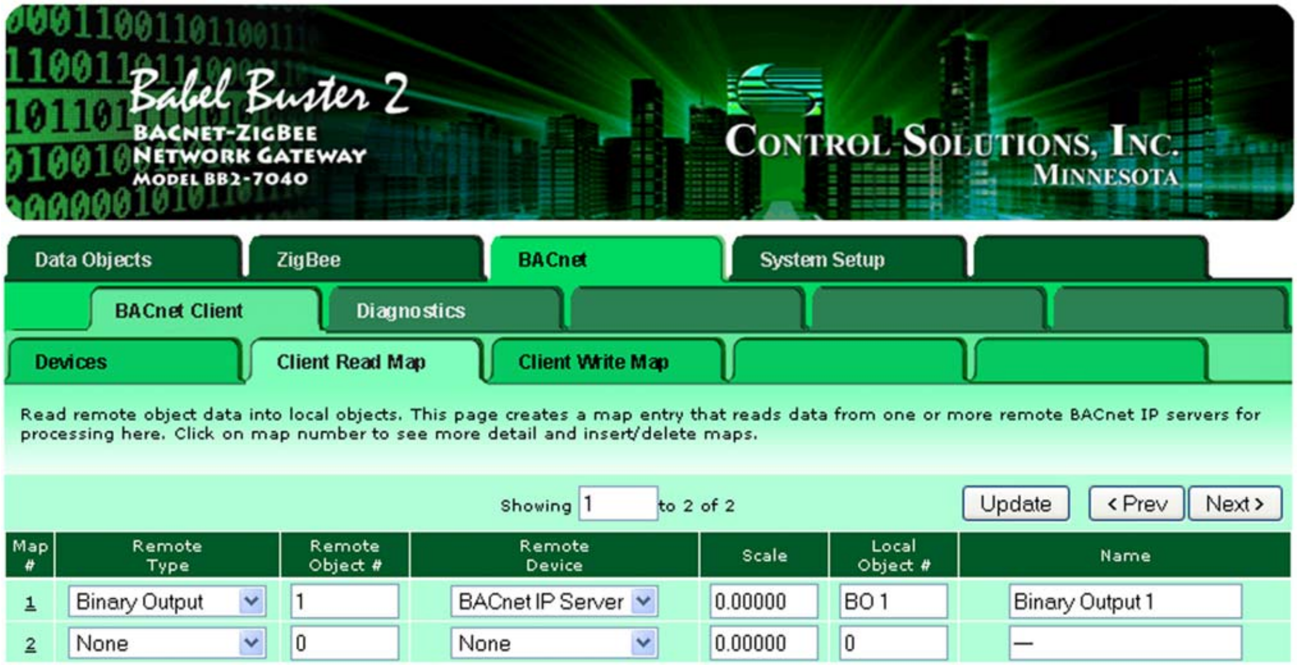
Device number simply shows you where you are on the device list. Click "next" and "prev" to scroll through the list.

Remote BACnet devices to be accessed by this device are specified here. Enter the Device Instance of the remote device, a name to reference in other pages, a poll rate, default reply timeout, and default write priority. Select IP or MS/TP (physical network on which to look for this device). Then click "update".

The gateway broadcasts a "who-is" looking for this device when a read or write map wants to use this device. When (if) it responds, its IP address or MS/TP mac address is listed here simply as a diagnostic. Timeouts resulting from inability to reach this device are tabulated on this page as well, and may be cleared by clicking the Clear button. To cause the who-is process to be repeated, click Clear Cache.

MS/TP slave devices that to not support Who-Is/I-Am can still be supported here. When this is the case, enter the slave device's Mac address in the Fixed MS/TP Mac window. This effectively replies to the implied Who-Is. A non-zero Mac means the Mac is fixed, therefore a fixed Mac of zero cannot be used.

7.2 BACnet Client Read Maps



Read remote object data into local objects. This page creates a map entry that reads data from one or more remote BACnet IP servers for processing here. Click on map number to see more detail and insert/delete maps.

Showing 1 to 2 of 2

Map #	Remote Type	Remote Object #	Remote Device	Scale	Local Object #	Name
1	Binary Output	1	BACnet IP Server	0.00000	BO 1	Binary Output 1
2	None	0	None	0.00000	0	—

Map number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only read data from remote devices. Go to the Client Write Map to write data to those devices. The full parameter set is different for read versus write.

An abbreviated version of a list of maps is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

For each remote object to be read, enter the object instance and type, and location (device). The names in the device list are defined in the Devices page.

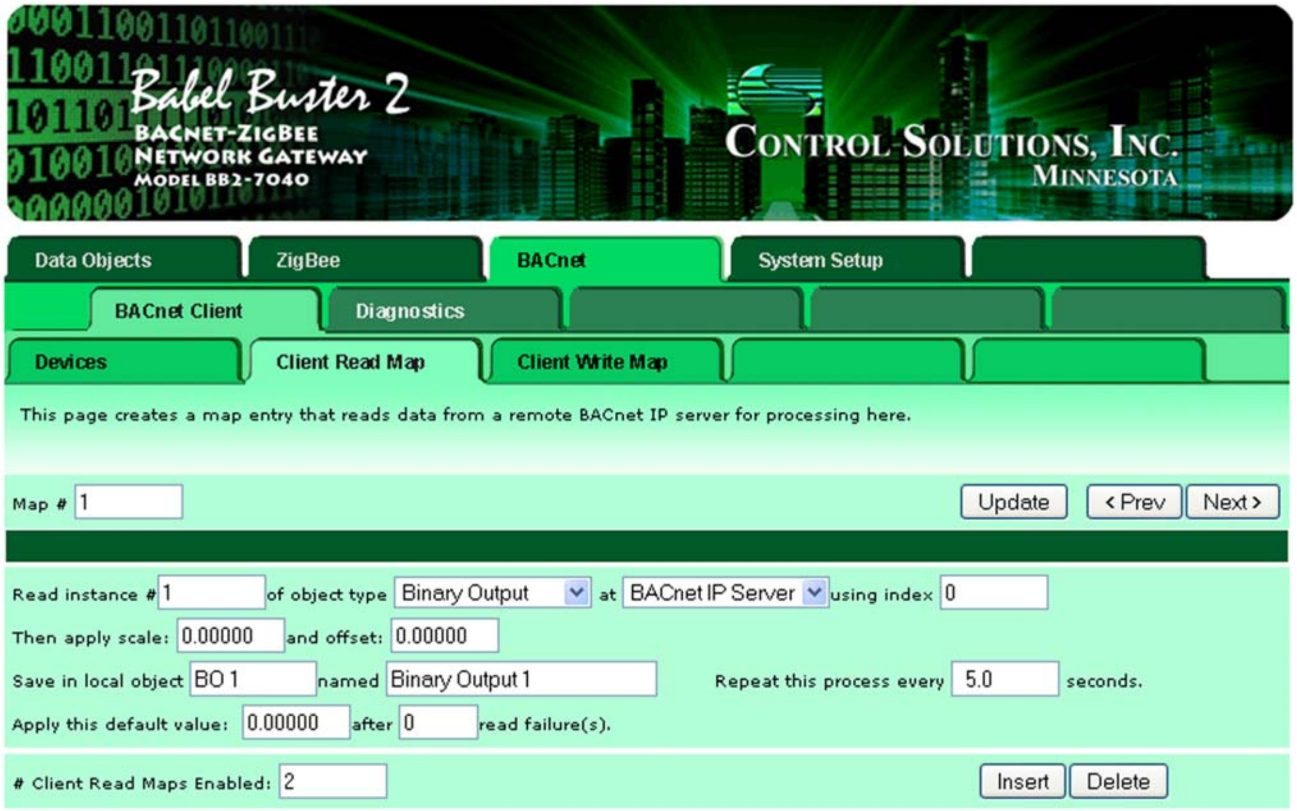
When the remote object is read, data may be manipulated before being written to the local object. The value will be multiplied by the scale factor. The final result is written to the local object number given. The name is optional and used only for display purposes.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

Local Object is internally a coded number consisting of BACnet object type multiplied by 1000, then added to the object number starting from #1. These are translated into abbreviations that are easy to interpret on the web page as follows:

AI n = Analog Input #n
AO n = Analog Output #n
AV n = Analog Value #n
BI n = Binary Input #n
BO n = Binary Output #n
BV n = Binary Value #n
MI n = Multi-state Input #n
MO n = Multi-state Output #n
MV n = Multi-state Value #n

Object numbers start at #1. The maximum available number varies by object type, and these limits may be found on the System Capacities link from the home/index page (click graphic at top).



This page creates a map entry that reads data from a remote BACnet IP server for processing here.

Map # 1

Read instance # 1 of object type Binary Output at BACnet IP Server using index 0

Then apply scale: 0.00000 and offset: 0.00000

Save in local object BO 1 named Binary Output 1 Repeat this process every 5.0 seconds.

Apply this default value: 0.00000 after 0 read failure(s).

Client Read Maps Enabled: 2

Rule number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

For each remote object to be read, enter the object instance and type, and location (device). The names in the device list are defined in the Devices page. Use index value of 0 if no index.

When the remote object is read, data may be manipulated before being written to the local object. The value will be multiplied by the scale factor, then the offset is added. The final result is written to the local object number given. The name is optional and used only for display purposes.

The periodic poll time determines how often the remote object will be read. This number, if nonzero, will override the default poll time given in the Devices page for the remote device being read.

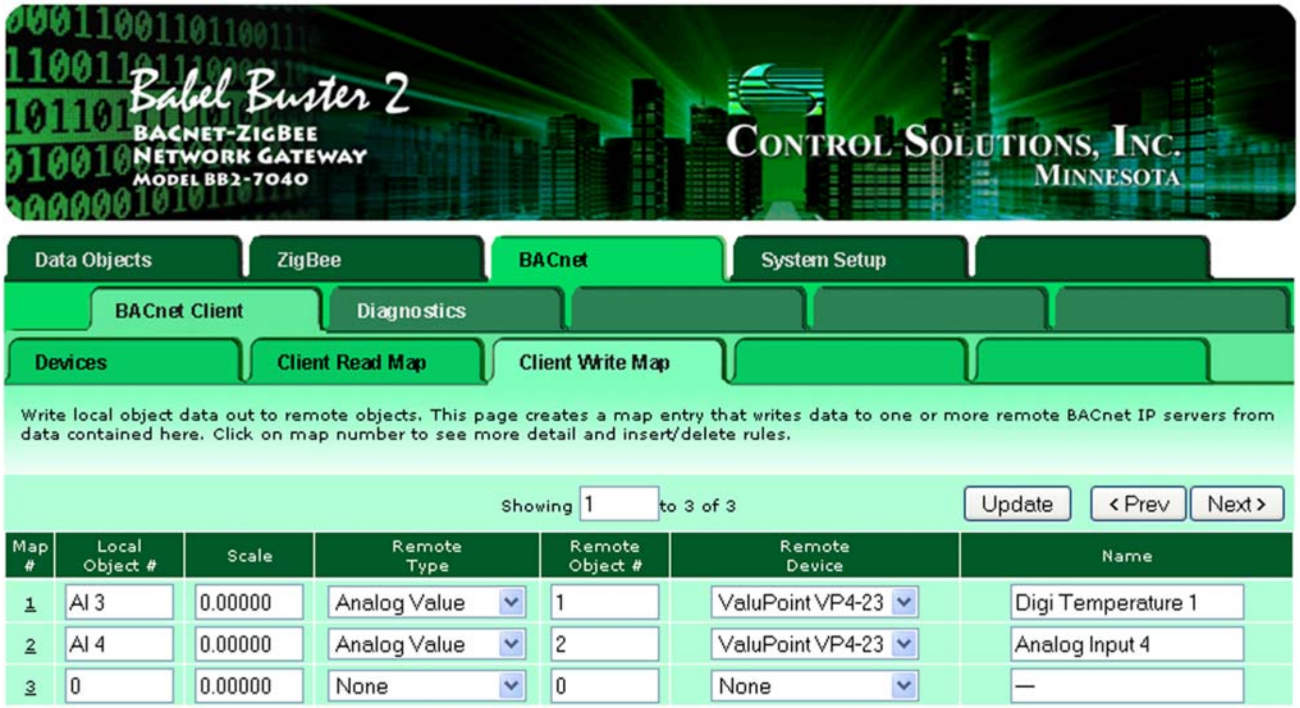
The default value will be stored into the local object after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained.

Delete will remove the rule number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

7.3 BACnet Client Write Maps



Write local object data out to remote objects. This page creates a map entry that writes data to one or more remote BACnet IP servers from data contained here. Click on map number to see more detail and insert/delete rules.

Showing 1 to 3 of 3

Map #	Local Object #	Scale	Remote Type	Remote Object #	Remote Device	Name
1	AI 3	0.00000	Analog Value	1	ValuPoint VP4-23	Digi Temperature 1
2	AI 4	0.00000	Analog Value	2	ValuPoint VP4-23	Analog Input 4
3	0	0.00000	None	0	None	—

Map number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only write data to remote devices. Go to the Client Read Map to read data from those devices. The full parameter set is different for read versus write.

An abbreviated version of a list of maps is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

Data from the local object given will be multiplied by the scale factor before being written. For each remote object to be written, enter the object instance and type, and location (device). The names in the device list are defined in the Devices page. The name is optional and used only for display purposes.

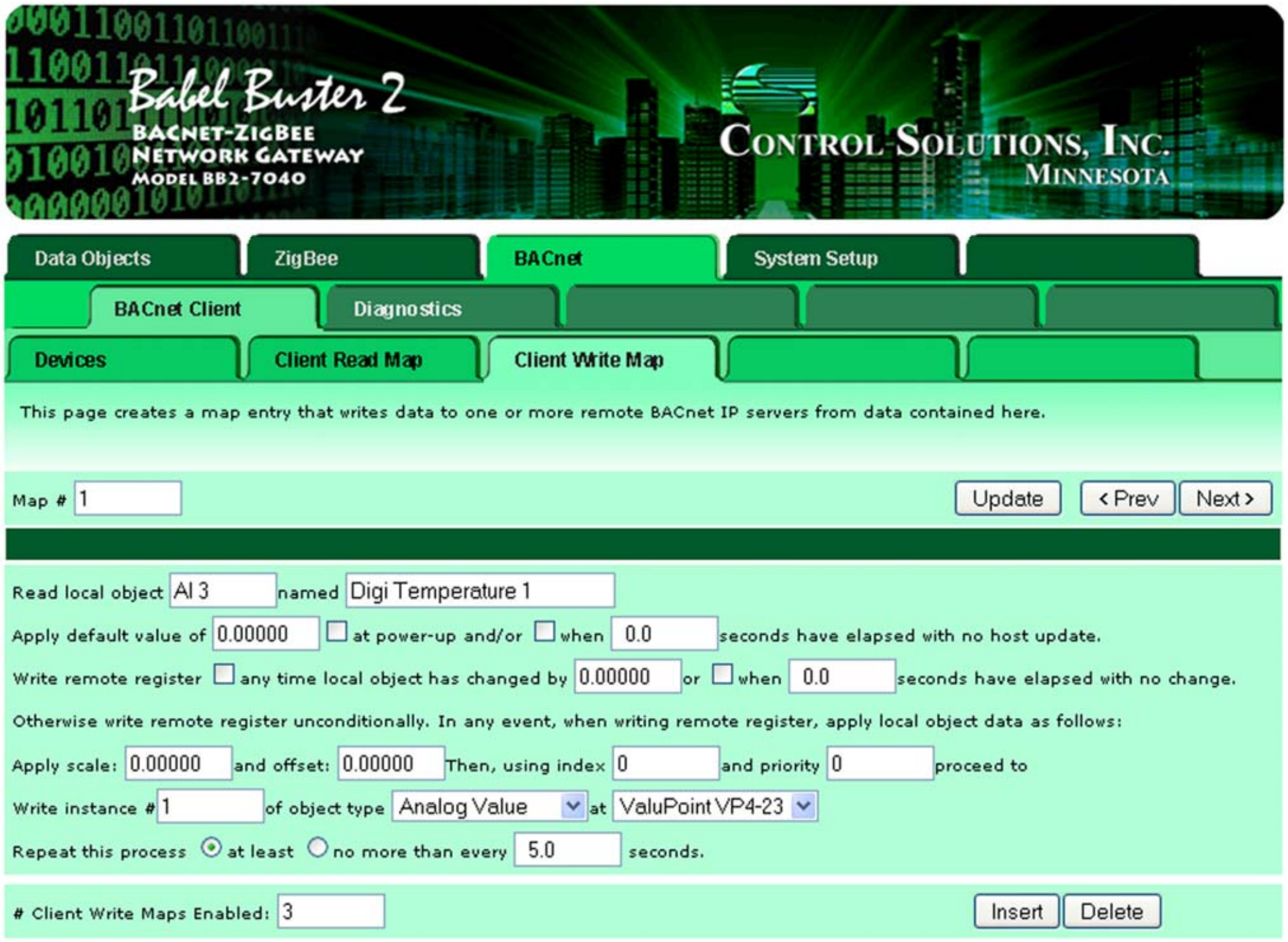
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

Local Object is internally a coded number consisting of BACnet object type multiplied by 1000, then added to the object number starting from #1. These are translated into abbreviations that are easy to interpret on the web page as follows:

AI n = Analog Input #n
AO n = Analog Output #n
AV n = Analog Value #n
BI n = Binary Input #n
BO n = Binary Output #n
BV n = Binary Value #n
MI n = Multi-state Input #n

MO n = Multi-state Output #n
MV n = Multi-state Value #n

Object numbers start at #1. The maximum available number varies by object type, and these limits may be found on the System Capacities link from the home/index page (click graphic at top).



This page creates a map entry that writes data to one or more remote BACnet IP servers from data contained here.

Map # 1

Read local object AI 3 named Digi Temperature 1

Apply default value of 0.00000 □ at power-up and/or □ when 0.0 seconds have elapsed with no host update.

Write remote register □ any time local object has changed by 0.00000 or □ when 0.0 seconds have elapsed with no change.

Otherwise write remote register unconditionally. In any event, when writing remote register, apply local object data as follows:

Apply scale: 0.00000 and offset: 0.00000 Then, using index 0 and priority 0 proceed to

Write instance # 1 of object type Analog Value at ValuPoint VP4-23

Repeat this process ● at least ○ no more than every 5.0 seconds.

Client Write Maps Enabled: 3

Rule number simply tells you where you're at on the list of object maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

The local object data may be written periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local object must change before being written to the remote device. To guarantee that the remote object will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local object may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it.

For the remote object to be written, enter the object instance and type, index if applicable (leave at 0 if not), and priority to use of the object being written is commandable. The names in the device list are defined in the Devices page.

The repeat time may determine how often the remote object will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic in the event changes are frequent, click the "no more than" button and enter the minumum time that should elapse before another write to the remote device.

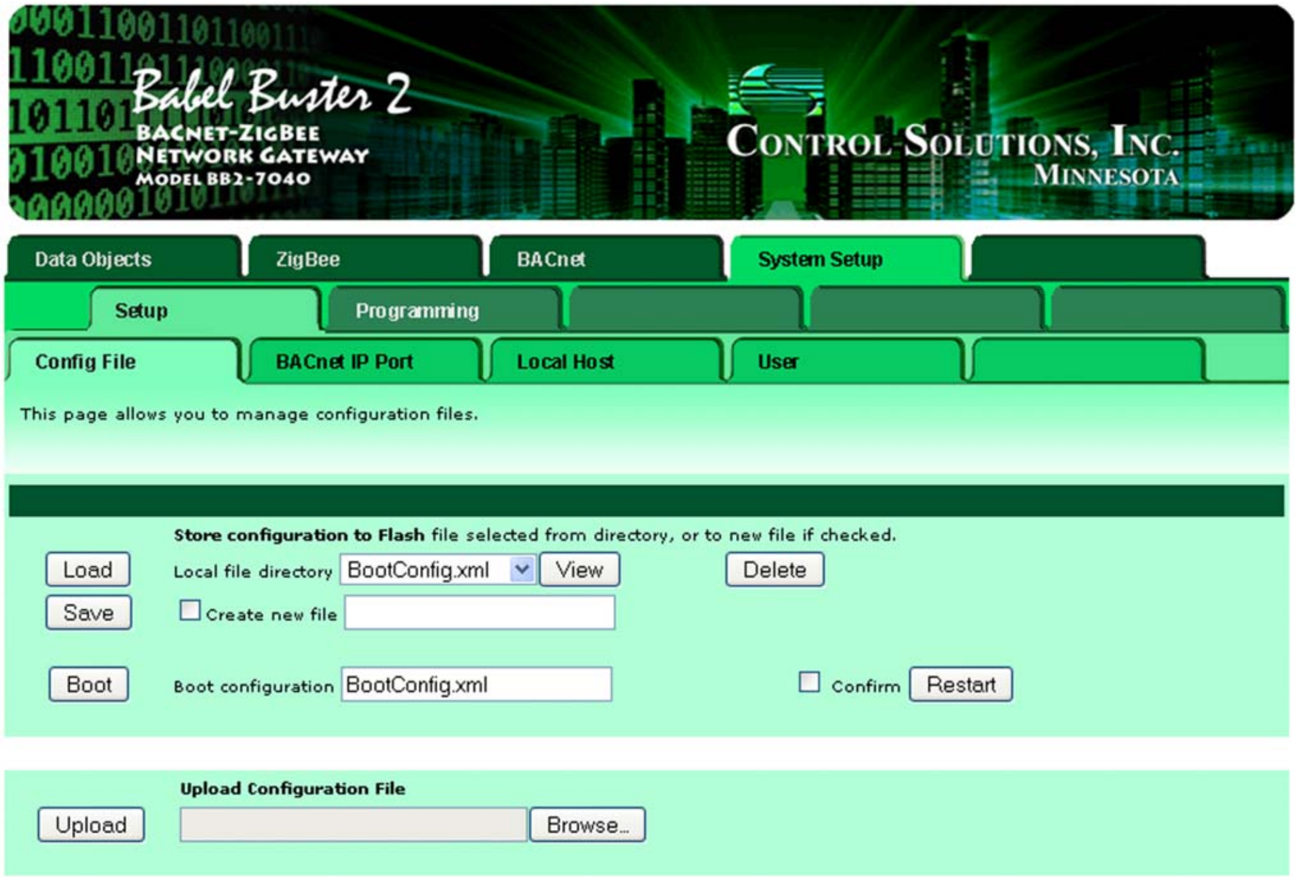
Delete will remove the rule number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

8 System Setup

8.1 Configuration Files



This page allows you to manage configuration files.

Store configuration to Flash file selected from directory, or to new file if checked.

Local file directory: BootConfig.xml

□ Create new file

Boot configuration: BootConfig.xml

□ Confirm

IMPORTANT: Configuration changes will be lost the next time you cycle power if you did not click the "save" button to place those changes in non-volatile memory (Flash file).

Click "Save" to store the present configuration to a Flash file. The configuration will overwrite the selected file in the local file directory unless you check "create new file" and enter a new file name.

Click "Load" to load the currently selected file in the local file directory. This means reconfigure the system from this file. Changes take effect immediately and could have consequences. Be certain to understand the changes you are invoking before clicking "load".

Click "Boot" to select a different boot configuration file. This is the Flash file that is automatically loaded upon power-up.

Click "View" to look at the selected file. It should be an XML file, and your browser will recognize it as such if properly formatted.

Click "Delete" to remove a file from the local file directory.

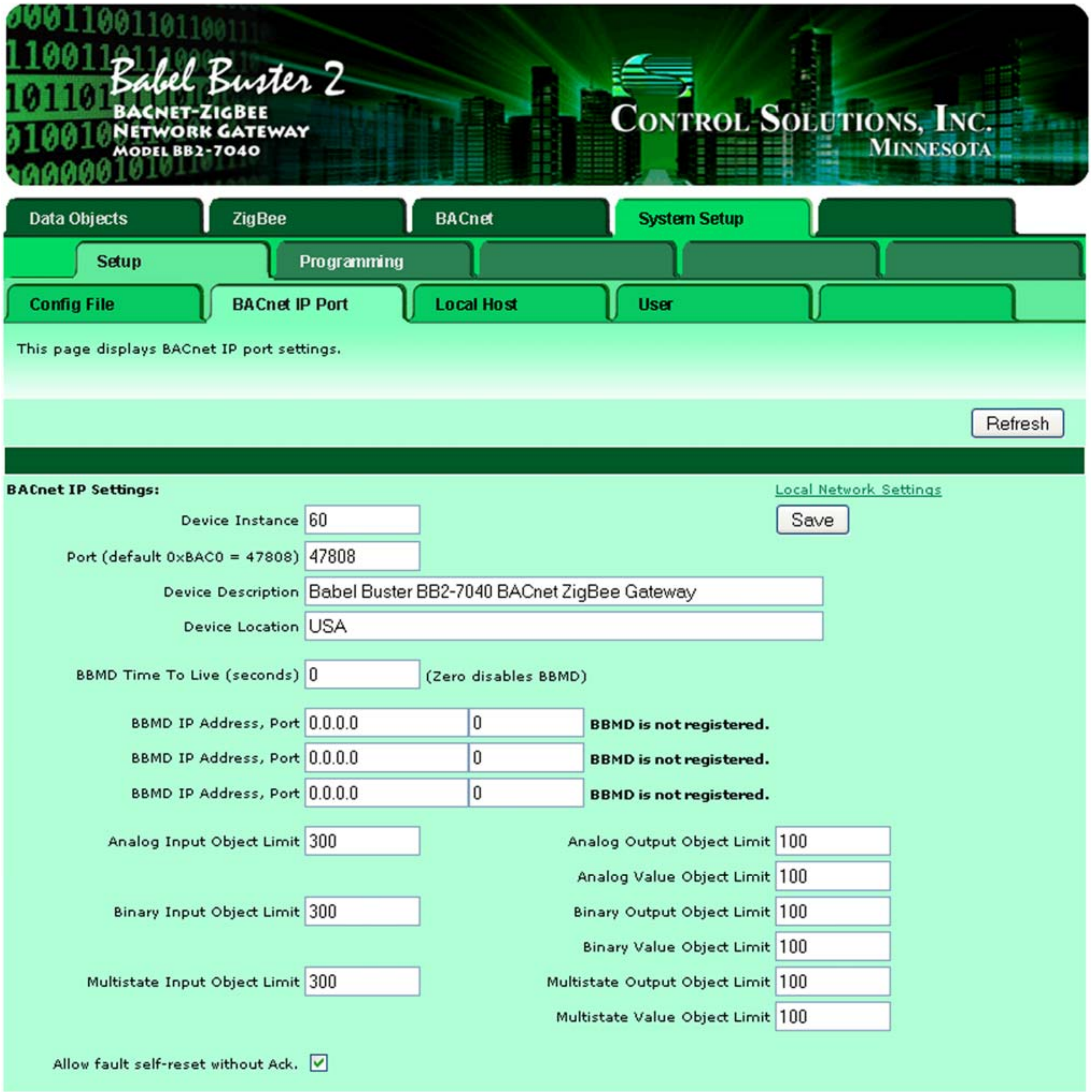
Click "Browse" to select a file for upload from your PC. Once selected, then click "Upload" to complete the process.

Check "Confirm" and click "Restart" to re-boot the system. This will not cycle power to the hardware, but will reboot the processor as if power had been cycled. Use this to force an IP address change remotely. Only the root user is allowed to do this. Upon restart, this page will not reload and the HTTP connection will be lost.

Note: There may be a delay of several seconds while Flash memory is updated by any of the above actions.

Note: Your browser may cache files. If you view a file, make configuration changes, save the file, then view the file again, you may see the old file cached by the browser. To see the updated file, go to "Internet Options" in your browser's "Tools" menu, and delete temporary Internet files (or delete cache files). Also, if you upload a file, make changes on your PC, and re-upload the same file, the browser may send the old file. Again, you will need to find the button inside your browser options that lets you delete the cached files from your PC.

8.2 BACnet Port Settings



This page displays BACnet IP port settings.

BACnet IP Settings:

Device Instance: 60

Port (default 0xBAC0 = 47808): 47808

Device Description: Babel Buster BB2-7040 BACnet ZigBee Gateway

Device Location: USA

BBMD Time To Live (seconds): 0 (Zero disables BBMD)

BBMD IP Address, Port: 0.0.0.0 | 0 **BBMD is not registered.**

BBMD IP Address, Port: 0.0.0.0 | 0 **BBMD is not registered.**

BBMD IP Address, Port: 0.0.0.0 | 0 **BBMD is not registered.**

Analog Input Object Limit: 300

Analog Output Object Limit: 100

Analog Value Object Limit: 100

Binary Input Object Limit: 300

Binary Output Object Limit: 100

Binary Value Object Limit: 100

Multistate Input Object Limit: 300

Multistate Output Object Limit: 100

Multistate Value Object Limit: 100

Allow fault self-reset without Ack. ☑

Enter a device instance from 1 to 4,194,303. Enter a port number (note that 47808 is the standard port expected by most BIP devices).

The description is the device property returned when Description, property ID 28, is read from the gateway. The location is the device property returned when Location, property ID 58, is read from the gateway.

If you will be using BBMD, the port, time-to-live, and IP address of the remote BBMD must be given (for up to three BBMD's). The local BBMD will attempt to register with the remote BBMD's whose addresses are given. Disable BBMD by setting time to live to zero (do this if not familiar with BBMD - this feature is only used in very large networks, and there should be only one BBMD enabled on any given subnet).

If there are no other BBMD's we want to register with, but we want to allow other 'foreign' devices to register with us, enable BBMD by setting the time to live, but leave all BBMD IP addresses set to 0.0.0.0. Up to 128 foreign devices may register with us, while we may register with up to 3 other BBMD's. Broadcast messages received from any of the other BBMD's will be resent to all 128 foreign devices registered with our BBMD.

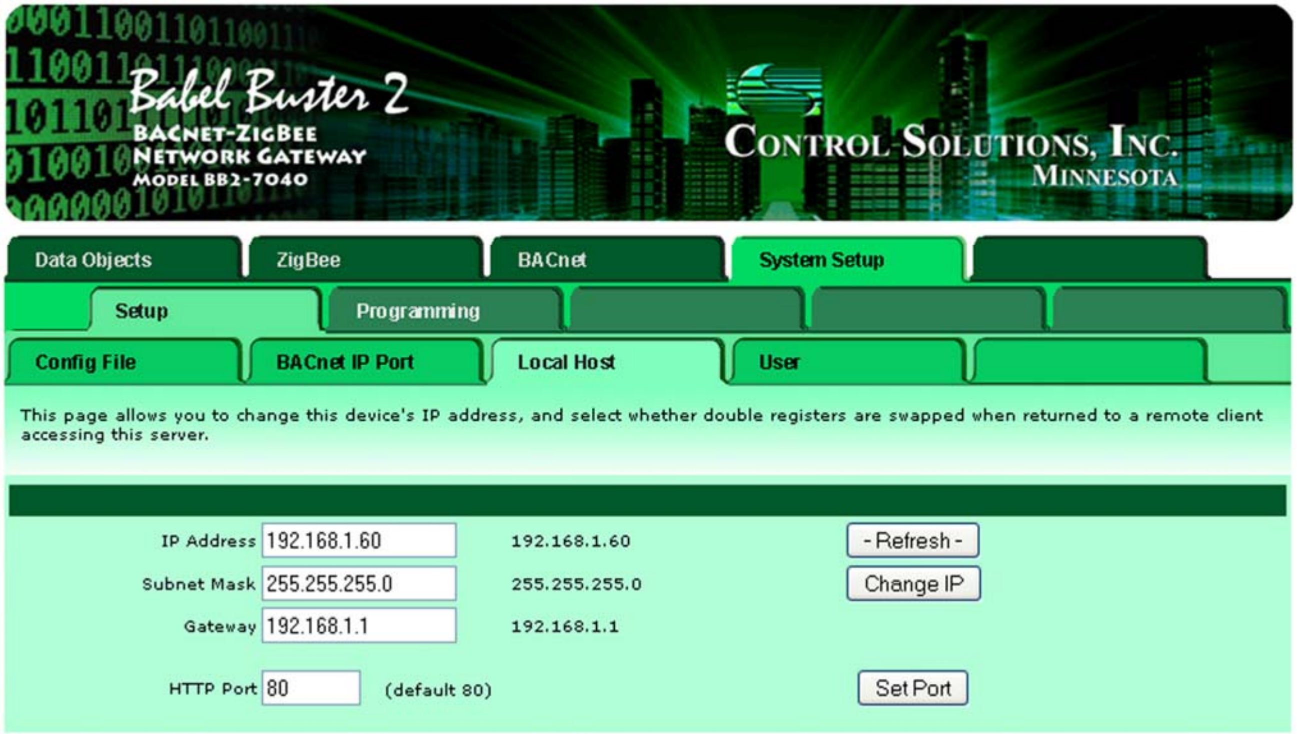
The object limits default to what the device supports, but may be set lower so that auto-discover routines in other BACnet clients only find the set of objects you are actually using. A minimum of 1 object of each type is required for proper device functioning.

Check the "allow" check box to allow faults to self-reset. These faults include things like Modbus RTU exception response, timeouts, etc. Normally the client needs to read the realiability code to acknowledge the fault before it will automatically reset. By checking the "allow" check box, faults will automatically self-reset without acknowledgement. This is required any time the client does not periodically read reliability codes but does check fault status.

Click Save to store. This store process will take a little while as these parameters are being saved to non-volatile memory. A change in port number will not take effect until the next system restart.

MSTP Port parameters: Enter parameters as applicable. The Max Masters count must be equal to or higher than the highest MAC address on the MSTP link. Click Update, then save the configuration file on the Config File page. The IP port parameters are saved in configuration memory, the MSTP parameters are saved in the XML configuration file.

8.3 Local Port Settings



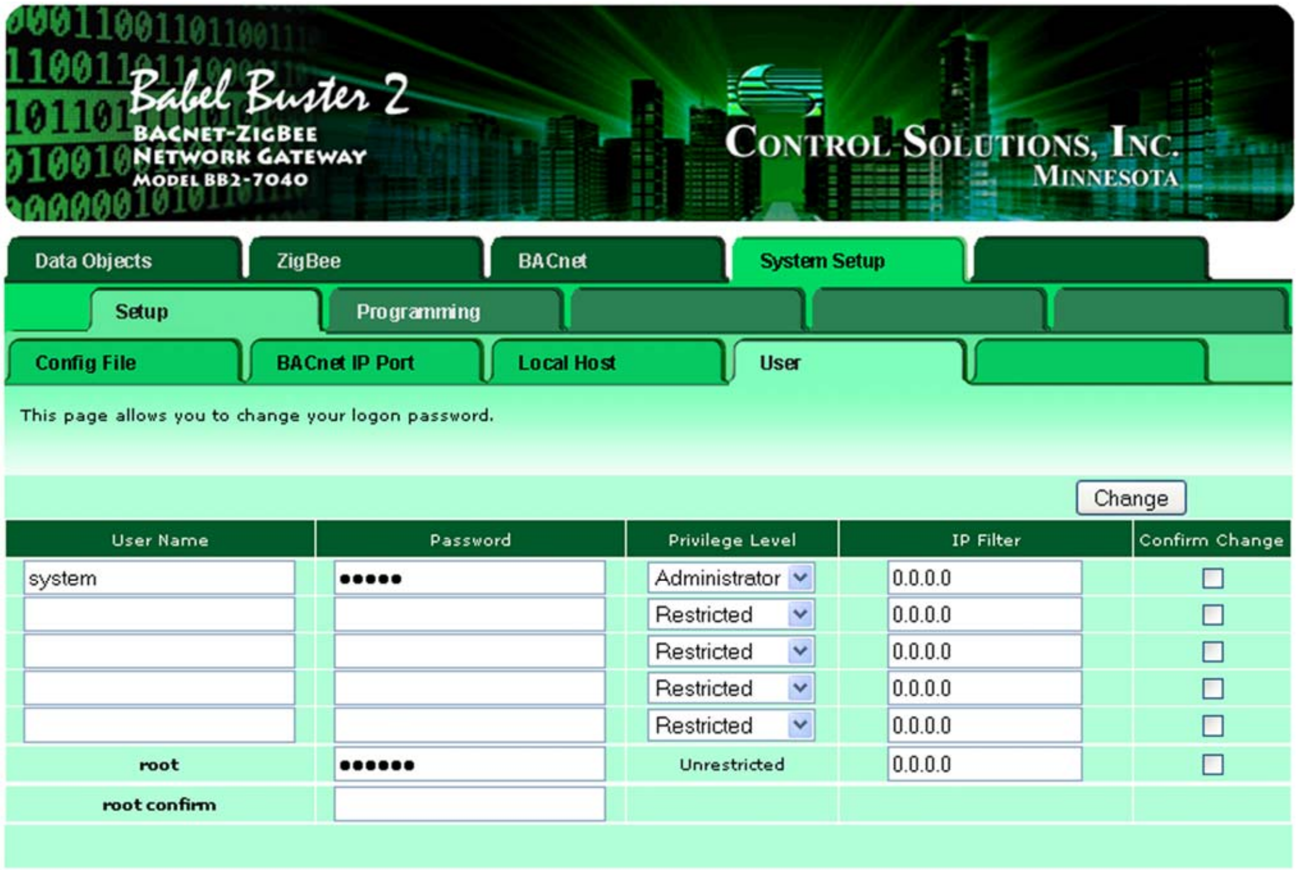
This page allows you to change this device's IP address, and select whether double registers are swapped when returned to a remote client accessing this server.

IP Address	192.168.1.60	192.168.1.60
Subnet Mask	255.255.255.0	255.255.255.0
Gateway	192.168.1.1	192.168.1.1
HTTP Port	80	(default 80)

To change the IP address of this device, enter the address, subnet mask, and gateway, then click "change IP". Set the IP address to 255.255.255.255 to specify that DHCP should be used to obtain an IP address upon power-up. IP address change will take effect upon next power cycle.

The default port for web page serving is 80. If you wish to change it, enter the port number and click Set Port. This change will take effect upon the next power-up. If the port is anything other than 80, you must include the port number in the URL. For example, if you would normally use http://10.0.0.101/ to get here and you change the port to 8215, you would now use http://10.0.0.101:8215/. (Note: The port change is only accepted if you are logged in as the root user.)

8.4 User Accounts and Passwords



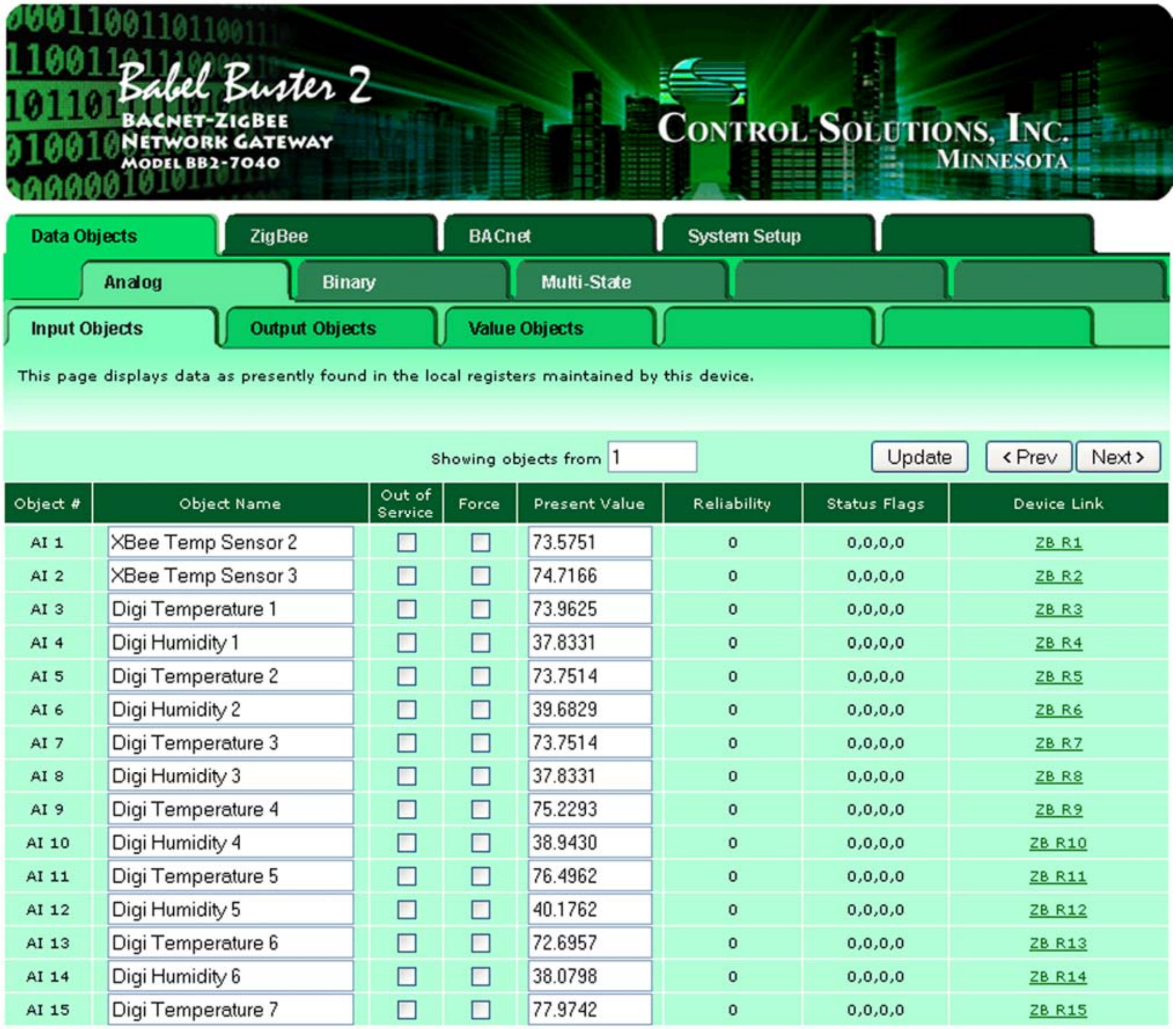
This page allows you to change your logon password.

User Name	Password	Privilege Level	IP Filter	Confirm Change
system	●●●●●	Administrator	0.0.0.0	□
		Restricted	0.0.0.0	□
		Restricted	0.0.0.0	□
		Restricted	0.0.0.0	□
		Restricted	0.0.0.0	□
root	●●●●●●	Unrestricted	0.0.0.0	□
root confirm				

You are logged in as the root user if you see this page. You may enter user names, their passwords, set their privilege levels, and enter an optional IP filter. If the IP filter is a non-zero valid IP address, the user may only log in from that IP address.

Changes will take at least a full minute before this page will refresh.

9 Data Objects



This page displays data as presently found in the local registers maintained by this device.

Showing objects from 1

Object #	Object Name	Out of Service	Force	Present Value	Reliability	Status Flags	Device Link
AI 1	XBee Temp Sensor 2	□	□	73.5751	0	0,0,0,0	ZB R1
AI 2	XBee Temp Sensor 3	□	□	74.7166	0	0,0,0,0	ZB R2
AI 3	Digi Temperature 1	□	□	73.9625	0	0,0,0,0	ZB R3
AI 4	Digi Humidity 1	□	□	37.8331	0	0,0,0,0	ZB R4
AI 5	Digi Temperature 2	□	□	73.7514	0	0,0,0,0	ZB R5
AI 6	Digi Humidity 2	□	□	39.6829	0	0,0,0,0	ZB R6
AI 7	Digi Temperature 3	□	□	73.7514	0	0,0,0,0	ZB R7
AI 8	Digi Humidity 3	□	□	37.8331	0	0,0,0,0	ZB R8
AI 9	Digi Temperature 4	□	□	75.2293	0	0,0,0,0	ZB R9
AI 10	Digi Humidity 4	□	□	38.9430	0	0,0,0,0	ZB R10
AI 11	Digi Temperature 5	□	□	76.4962	0	0,0,0,0	ZB R11
AI 12	Digi Humidity 5	□	□	40.1762	0	0,0,0,0	ZB R12
AI 13	Digi Temperature 6	□	□	72.6957	0	0,0,0,0	ZB R13
AI 14	Digi Humidity 6	□	□	38.0798	0	0,0,0,0	ZB R14
AI 15	Digi Temperature 7	□	□	77.9742	0	0,0,0,0	ZB R15

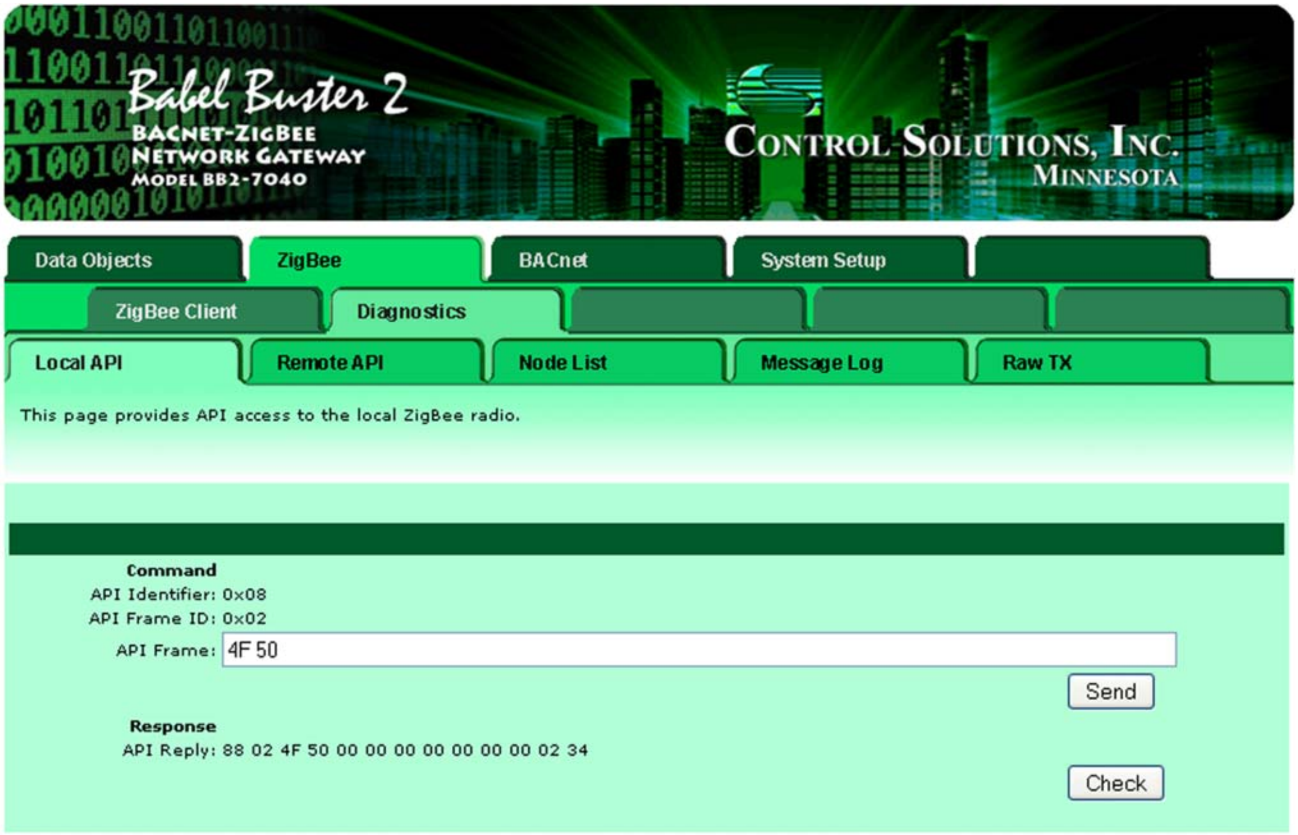
The source of data for an Analog Input object will be reading an object in another BACnet device, or receiving of data from a ZigBee device. The device link will point to a BACnet client read map or a ZigBee client read map.

Out of Service means any polling of the slave device will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next poll unless the object is out of service.

Reliability codes may be any of the following:
BACnet device timeout (82)
BACnet error returned by server/slave BACnet device (83)
ZigBee device timeout (84)
ZigBee device error (85)
ZigBee reported invalid command (86)
ZigBee reported invalid parameter (87)
Other (99)

Device link will indicate BAC or ZB, followed by R for read or W for write, and a number which is the rule number in the table of read or write rules for mapping external devices or objects to this BACnet object. The designation R means read from external device, and W means write to external device.

10 ZigBee Diagnostic Pages



This page provides API access to the local ZigBee radio.

Command

API Identifier: 0x08

API Frame ID: 0x02

API Frame: 4F 50

Response

API Reply: 88 02 4F 50 00 00 00 00 00 00 00 02 34

The BB2-7040 contains a Digi International/Maxstream XBee PRO (Ember 250 chip) operating as a coordinator. You can do a variety of things with the BB2-7040 using API commands for the XBee PRO as found in XBee PRO documentation at www.digi.com. You can also find links to applicable documentation on the BB2-7040 product page at www.csimn.com.

The XBee PRO ZigBee coordinator is programmed to operate in API mode (rather than AT mode) as defined in XBee PRO documentation. The API mode allows access to a much greater range of capability and is required for BB2-7040 use.

Local API commands are constructed and sent on this page. API commands to remote devices are constructed and sent from the Remote API page. Local API commands affect the local radio.

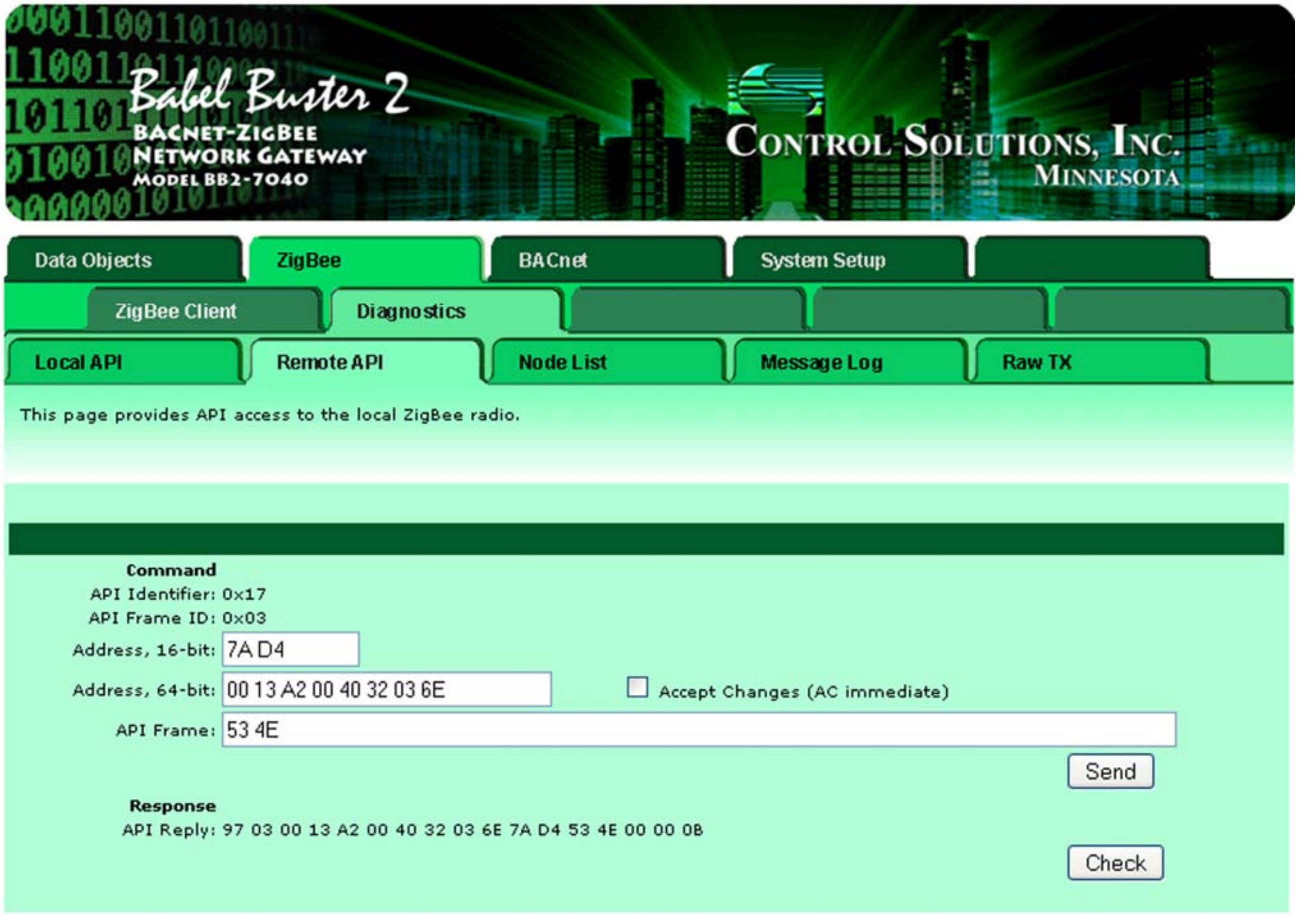
The local API frame input window above provides interpretation and translation of ASCII strings for the construction of raw API packets for transmission. This provides a simple way to create the API version of AT commands.

You may perform a very simple test on the local radio to see if the XBee PRO is responding to commands. the "SL" command should always succeed since this is simply returning the local device address. Enter "/SL" in the API Frame window and click Send. The response that appears shortly for API Reply should be something like: 88 03 53 4C 00 40 30 9E C3 where 88 will always be 88, the 03 in this example will be whatever your next frame ID happened to be, the 53 4C should always be 53 4C, and the next 4 bytes will be the low order half of your local 64-bit address. The last byte is the CRC.

Command interpretation is as follows:

'/' -- results in accepting any number of ASCII alphanumeric characters (0-9,A-Z) with ASCII codes placed in transmit packet.

' " ' (quote) -- results in accepting a quote delimited string, required for including blanks or punctuation
NN -- literal number, always interpreted as hexadecimal, delimited by blanks, creates a single byte with this hex code.



This page provides API access to the local ZigBee radio.

Command

API Identifier: 0x17
API Frame ID: 0x03
Address, 16-bit: 7A D4
Address, 64-bit: 00 13 A2 00 40 32 03 6E
□ Accept Changes (AC immediate)
API Frame: 53 4E

Response

API Reply: 97 03 00 13 A2 00 40 32 03 6E 7A D4 53 4E 00 00 0B

Remote API commands are constructed and sent on this page. Remote API commands work exactly like local API commands, except that they are sent to a remote ZigBee device. These API commands are specific to XBee PRO/Ember 250. Commands sent to ZigBee devices using a stack other than Ember will require construction of raw packets on the Raw Tx page.

The remote API frame input window above provides interpretation and translation of ASCII strings for the construction of raw API packets for transmission. This provides a simple way to create the API version of AT commands.

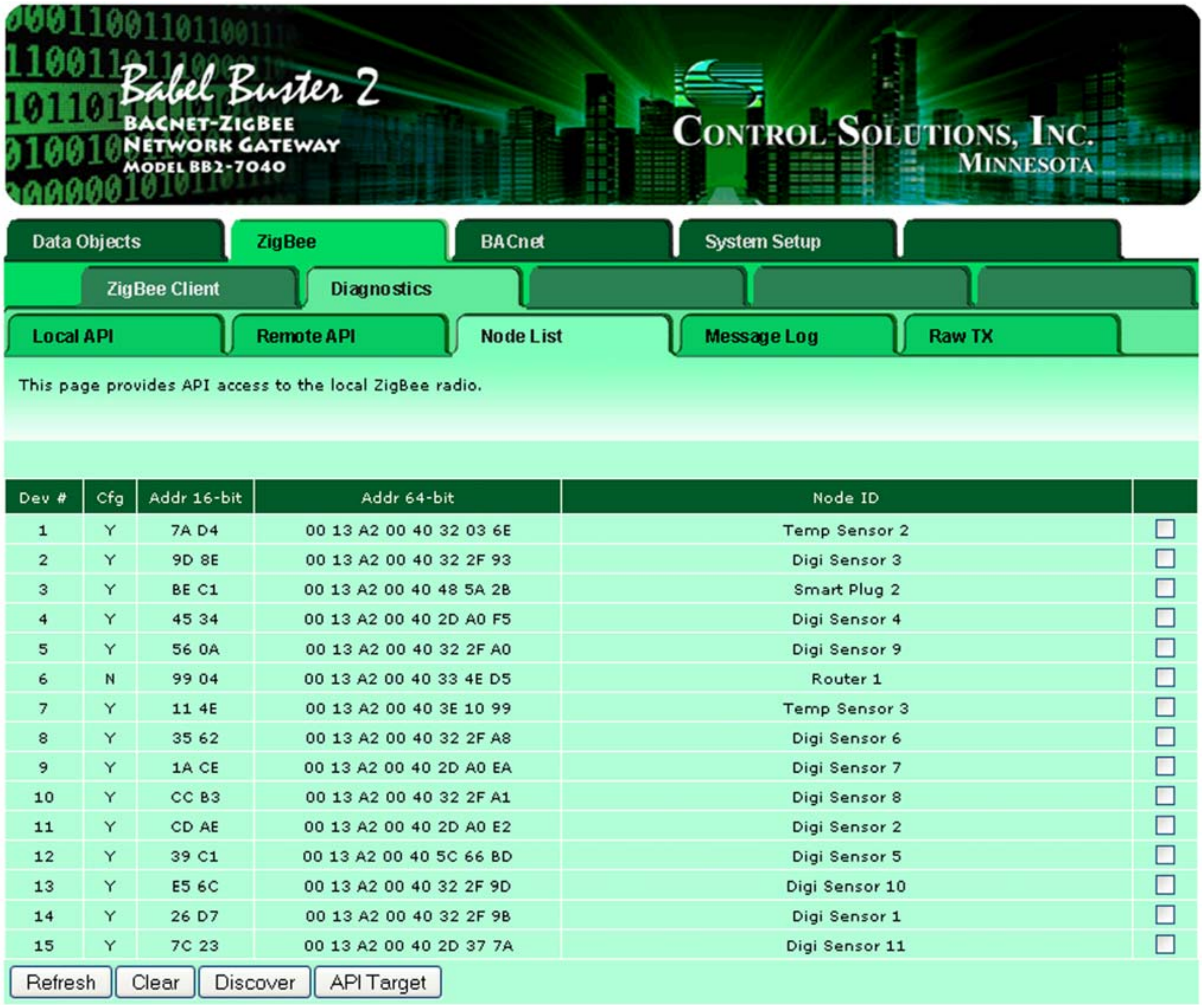
The API command will be sent to the device whose address appears in the address windows. You can enter these manually, or have them filled in automatically by clicking the API Target button on the Node List page.

Command interpretation is as follows:

'/' -- results in accepting any number of ASCII alphanumeric characters (0-9,A-Z) with ASCII codes placed in transmit packet.
' " ' (quote) -- results in accepting a quote delimited string, required for including blanks or punctuation
NN -- literal number, always interpreted as hexadecimal, delimited by blanks, creates a single byte with this hex code.

An example command that sets the Network Interface name of a remote device would be:
/NI "Temp Sensor 1"

Configuration changes at the remote device will not take effect immediately unless you check the Accept Changes box. However, there are circumstances where you do not want to do this. If changing a number of parameters affecting how the remote device transmits, you may wish to make several changes, followed by sending the /AC command to accept the changes at that point.



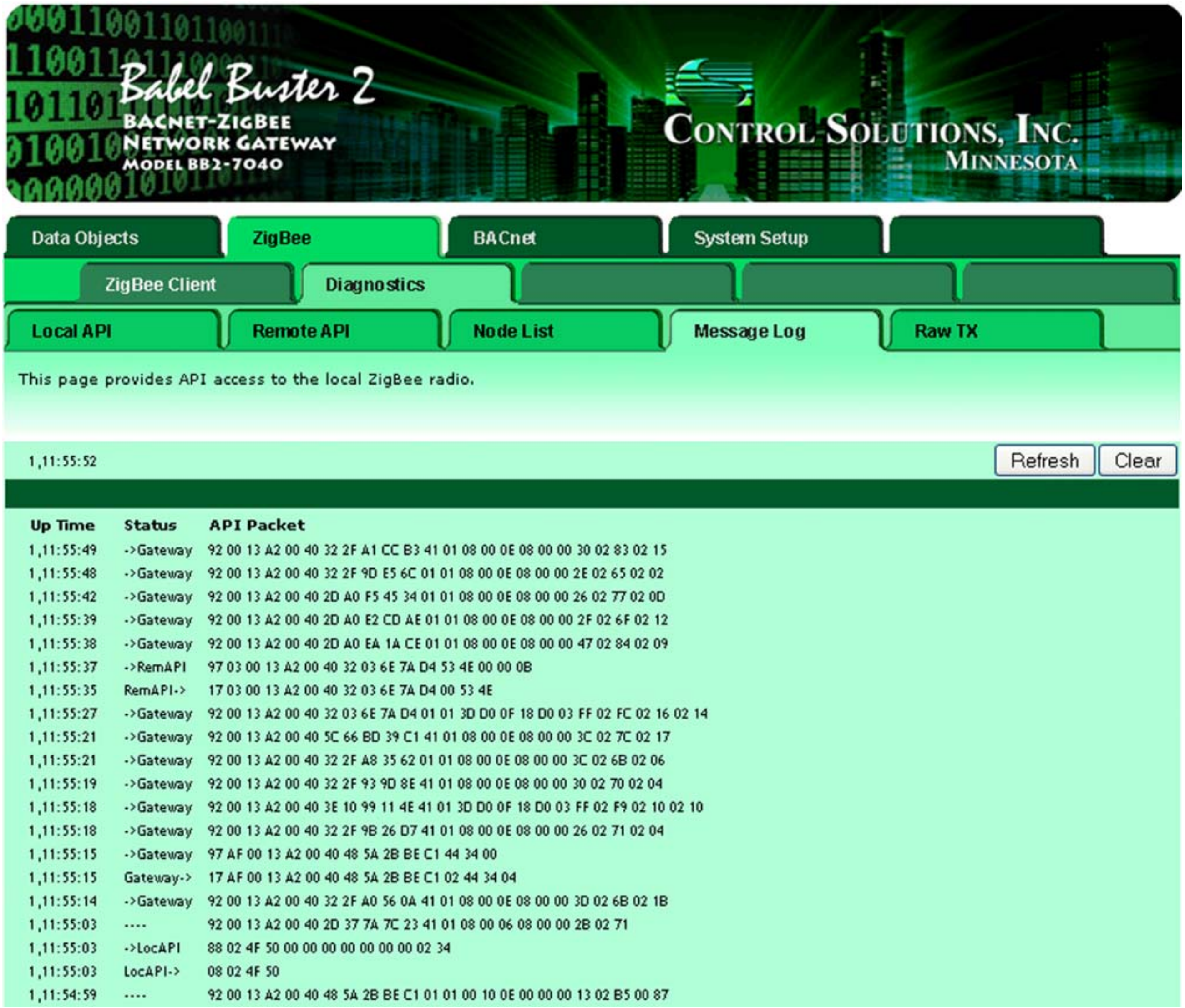
This page provides API access to the local ZigBee radio.

Dev #	Cfg	Addr 16-bit	Addr 64-bit	Node ID	
1	Y	7A D4	00 13 A2 00 40 32 03 6E	Temp Sensor 2	☐
2	Y	9D 8E	00 13 A2 00 40 32 2F 93	Digi Sensor 3	☐
3	Y	BE C1	00 13 A2 00 40 48 5A 2B	Smart Plug 2	☐
4	Y	45 34	00 13 A2 00 40 2D A0 F5	Digi Sensor 4	☐
5	Y	56 0A	00 13 A2 00 40 32 2F A0	Digi Sensor 9	☐
6	N	99 04	00 13 A2 00 40 33 4E D5	Router 1	☐
7	Y	11 4E	00 13 A2 00 40 3E 10 99	Temp Sensor 3	☐
8	Y	35 62	00 13 A2 00 40 32 2F A8	Digi Sensor 6	☐
9	Y	1A CE	00 13 A2 00 40 2D A0 EA	Digi Sensor 7	☐
10	Y	CC B3	00 13 A2 00 40 32 2F A1	Digi Sensor 8	☐
11	Y	CD AE	00 13 A2 00 40 2D A0 E2	Digi Sensor 2	☐
12	Y	39 C1	00 13 A2 00 40 5C 66 BD	Digi Sensor 5	☐
13	Y	E5 6C	00 13 A2 00 40 32 2F 9D	Digi Sensor 10	☐
14	Y	26 D7	00 13 A2 00 40 32 2F 9B	Digi Sensor 1	☐
15	Y	7C 23	00 13 A2 00 40 2D 37 7A	Digi Sensor 11	☐

This page shows the list of devices that have associated themselves with the BB2-7040's ZigBee radio functioning as a coordinator. Click Discover to send the /ND command to the local radio. It will respond by sending its list of devices which are then displayed here. Click Refresh after waiting a short time for all of the responses. Click Clear to clear the page and repopulate the list.

Cfg Y/N column indicates whether this device is on our list of target devices for gateway functionality (ZigBee Client Devices page).

To select a target device for Remote API access, check the corresponding box and click the API Target button. Then proceed to the Remote API page to send remote commands to this device.



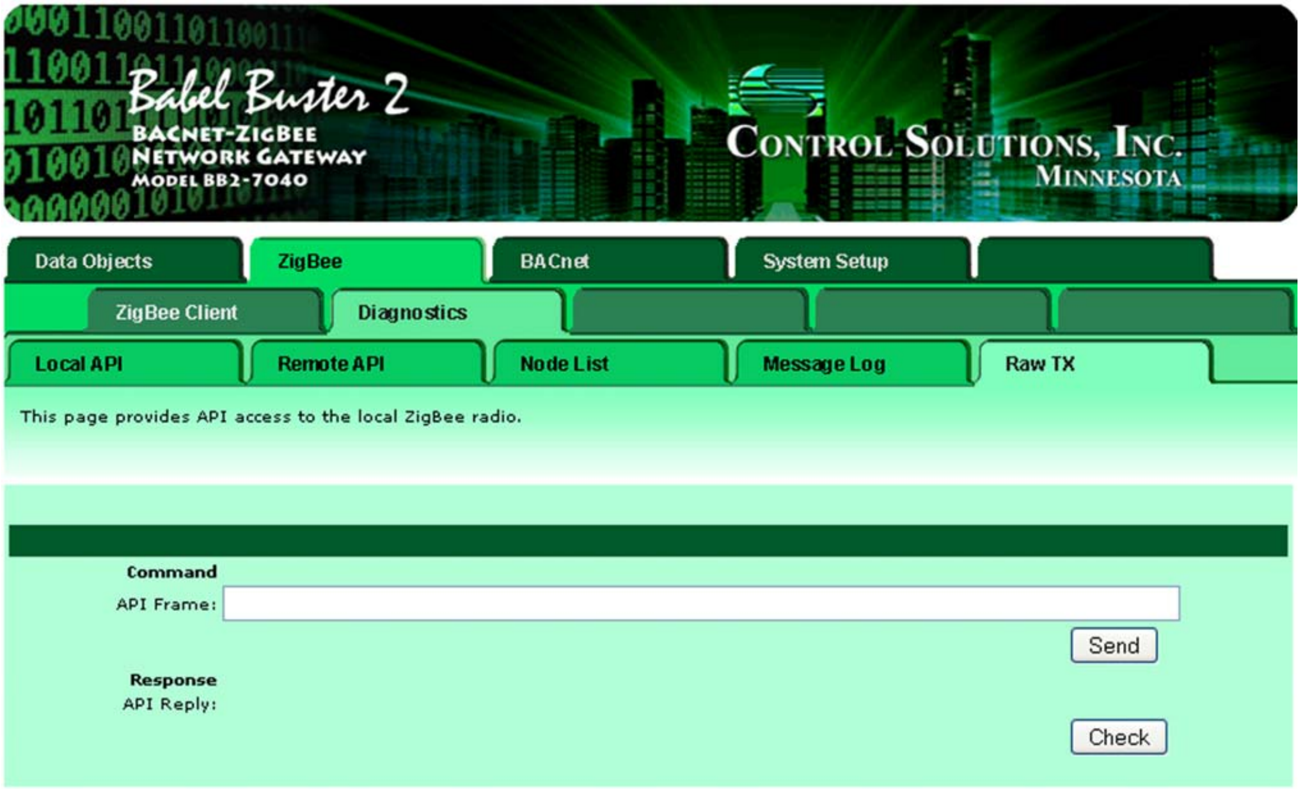
This page provides API access to the local ZigBee radio.

```
1,11:55:52

Up Time     Status      API Packet
1,11:55:49  ->Gateway   92 00 13 A2 00 40 32 2F A1 CC B3 41 01 08 00 0E 08 00 00 30 02 83 02 15
1,11:55:48  ->Gateway   92 00 13 A2 00 40 32 2F 9D E5 6C 01 01 08 00 0E 08 00 00 2E 02 65 02 02
1,11:55:42  ->Gateway   92 00 13 A2 00 40 2D A0 F5 45 34 01 01 08 00 0E 08 00 00 26 02 77 02 0D
1,11:55:39  ->Gateway   92 00 13 A2 00 40 2D A0 E2 CD AE 01 01 08 00 0E 08 00 00 2F 02 6F 02 12
1,11:55:38  ->Gateway   92 00 13 A2 00 40 2D A0 EA 1A CE 01 01 08 00 0E 08 00 00 47 02 84 02 09
1,11:55:37  ->RemAPI    97 03 00 13 A2 00 40 32 03 6E 7A D4 53 4E 00 00 0B
1,11:55:35  RemAPI->    17 03 00 13 A2 00 40 32 03 6E 7A D4 00 53 4E
1,11:55:27  ->Gateway   92 00 13 A2 00 40 32 03 6E 7A D4 01 01 3D D0 0F 18 D0 03 FF 02 FC 02 16 02 14
1,11:55:21  ->Gateway   92 00 13 A2 00 40 5C 66 BD 39 C1 41 01 08 00 0E 08 00 00 3C 02 7C 02 17
1,11:55:21  ->Gateway   92 00 13 A2 00 40 32 2F A8 35 62 01 01 08 00 0E 08 00 00 3C 02 6B 02 06
1,11:55:19  ->Gateway   92 00 13 A2 00 40 32 2F 93 9D 8E 41 01 08 00 0E 08 00 00 30 02 70 02 04
1,11:55:18  ->Gateway   92 00 13 A2 00 40 3E 10 99 11 4E 41 01 3D D0 0F 18 D0 03 FF 02 F9 02 10 02 10
1,11:55:18  ->Gateway   92 00 13 A2 00 40 32 2F 9B 26 D7 41 01 08 00 0E 08 00 00 26 02 71 02 04
1,11:55:15  ->Gateway   97 AF 00 13 A2 00 40 48 5A 2B BE C1 44 34 00
1,11:55:15  Gateway->   17 AF 00 13 A2 00 40 48 5A 2B BE C1 02 44 34 04
1,11:55:14  ->Gateway   92 00 13 A2 00 40 32 2F A0 56 0A 41 01 08 00 0E 08 00 00 3D 02 6B 02 1B
1,11:55:03  ----        92 00 13 A2 00 40 2D 37 7A 7C 23 41 01 08 00 06 08 00 00 2B 02 71
1,11:55:03  ->LocAPI    88 02 4F 50 00 00 00 00 00 00 00 02 34
1,11:55:03  LocAPI->    08 02 4F 50
1,11:54:59  ----        92 00 13 A2 00 40 48 5A 2B BE C1 01 01 00 10 0E 00 00 00 13 02 B5 00 87
```

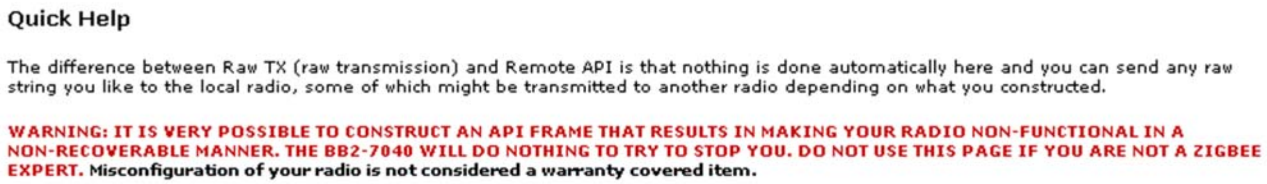
This page provides a log of the last 64 packets (API frames) sent or received by the BB2-7040. It does not show all ZigBee RF traffic in the air, only that which was addressed to or broadcast to the BB2-7040 (which operates as a coordinator).

Packet status indicates how the packet was processed, as follows. The designation XXX-> means outgoing, and ->XXX means incoming.

LocAPI-> ->LocAPI	Packets processed by the Local API web page
RemAPI-> ->RemAPI	Packets processed by the Remote API web page
Gateway-> ->Gateway	Packets processed by gateway functionality defined in ZigBee Client Read Map and Write Map pages.
Basic-> ->Basic	Packets processed by the user's Script Basic program.
----	Packet received but not recognized for processing for any of the above.



This page provides API access to the local ZigBee radio.



Quick Help

The difference between Raw TX (raw transmission) and Remote API is that nothing is done automatically here and you can send any raw string you like to the local radio, some of which might be transmitted to another radio depending on what you constructed.

WARNING: IT IS VERY POSSIBLE TO CONSTRUCT AN API FRAME THAT RESULTS IN MAKING YOUR RADIO NON-FUNCTIONAL IN A NON-RECOVERABLE MANNER. THE BB2-7040 WILL DO NOTHING TO TRY TO STOP YOU. DO NOT USE THIS PAGE IF YOU ARE NOT A ZIGBEE EXPERT. **Misconfiguration of your radio is not considered a warranty covered item.**

The difference between Raw TX (raw transmission) and Remote API is that nothing is done automatically here and you can send any raw string you like to the local radio, some of which might be transmitted to another radio depending on what you constructed. Do note the warning about Raw TX.

API frames created here require that you explicitly provide the API frame type code, the 64-bit and 16-bit addresses of the remote device as applicable, and any packet data as applicable. Only the header, length, and checksum bytes used to move the packet from the BB2-7040 to the ZigBee radio are automatically generated here. All bytes given here are sent to the radio exactly as-is. By using raw transmit (API codes 0x10 and 0x11), it is possible to send packets to non-Ember ZigBee devices.

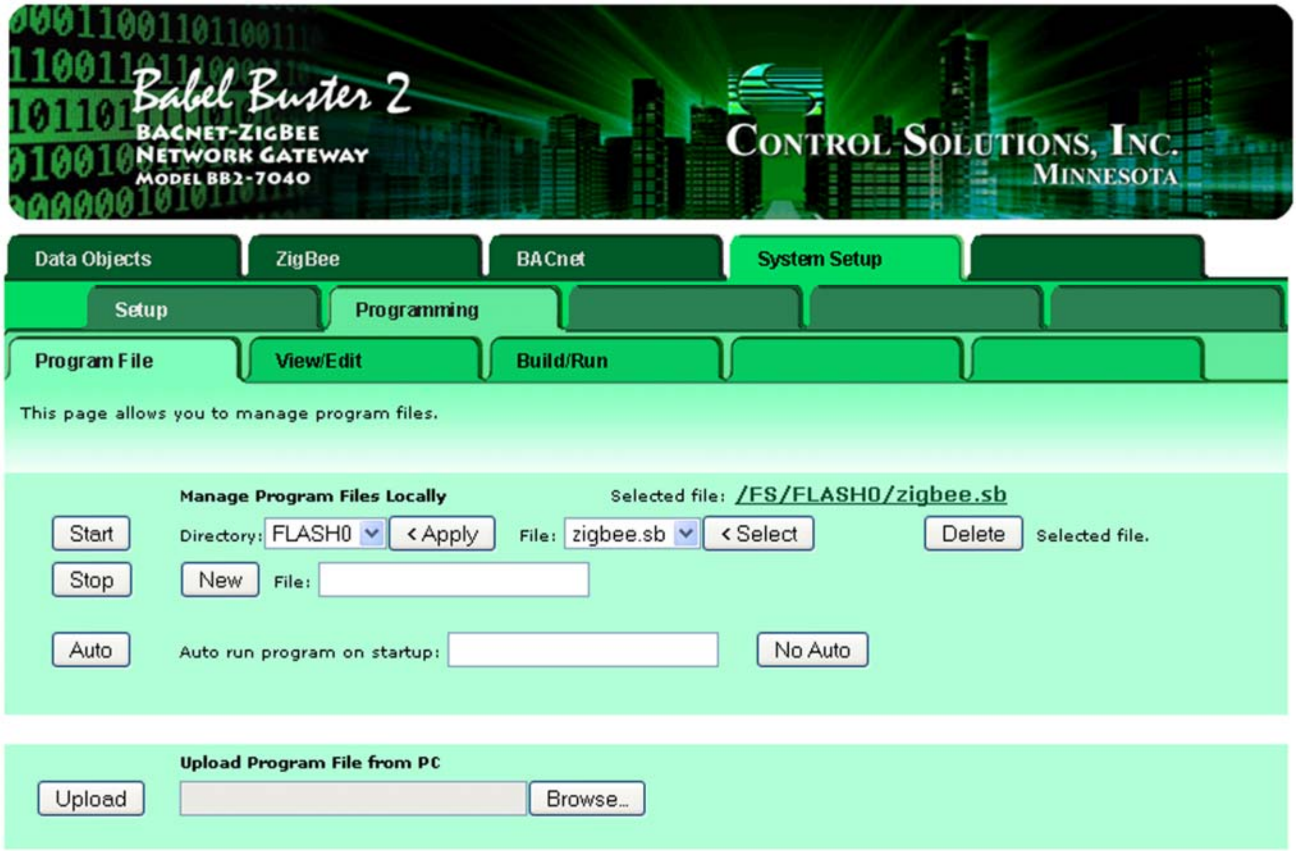
Generally, you should use whatever tools were provided by the sensor manufacturer to configure that sensor and get it to start sending data. Once data starts showing up at the BB2-7040, you only then need concern yourself with with creating a data parsing mask on the ZigBee Client Read Map page.

Command interpretation is as follows:

'/' -- results in accepting any number of ASCII alphanumeric characters (0-9,A-Z) with ASCII codes placed in transmit packet.
' " ' (quote) -- results in accepting a quote delimited string, required for including blanks or punctuation
NN -- literal number, always interpreted as hexadecimal, delimited by blanks, creates a single byte with this hex code.

11 Programming ZigBee I/O in Basic

11.1 Web Interface Pages for Programming



This page allows you to manage program files.

Selected file: /FS/FLASH0/zigbee.sb

Directory: FLASH0

File: zigbee.sb

Selected file.

File:

Auto run program on startup:

Click "Apply" to change directory and refresh the file list from that directory. Only files with a suffix of .sb will appear in this list.

Click "Select" to select the program file. This selection will apply to "Start", and also carry over to the edit and virtual terminal pages.

Click "New" to create a new file having the name entered in the window.

Click "Start" to run the selected program in the background. Click "Stop" to terminate this program if it is an endless loop that will not terminate on its own. If you are implementing a control loop, you will want the program to be an endless loop.

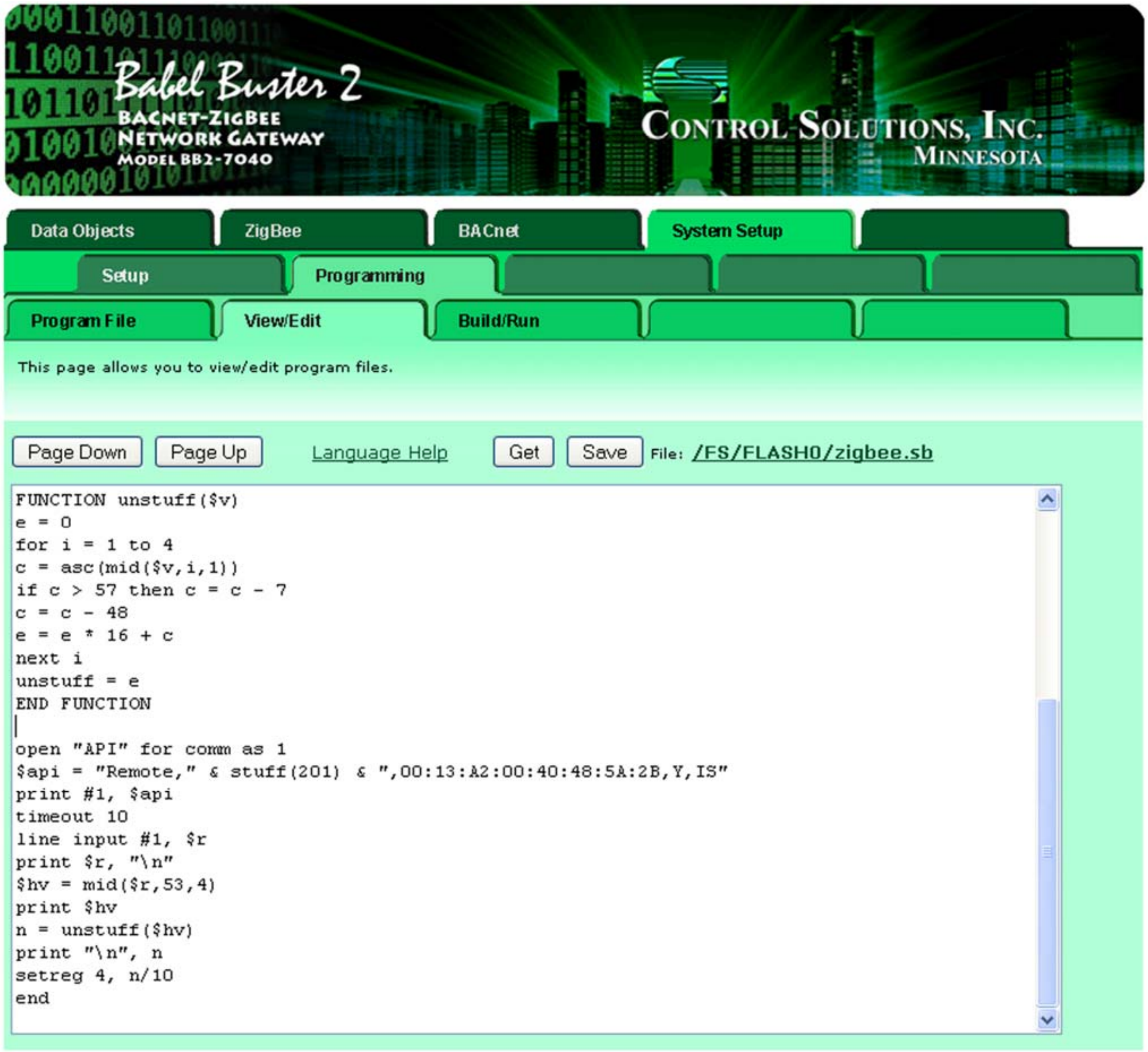
Click "Auto" to select the file that will be started automatically at boot-up. Clear the name window and click "Auto" to disable any previously selected auto-run program. This file name is saved in your boot configuration file, therefore you must save your configuration file in order to retain this auto run file name through power-up.

Click "Delete" to remove a file from the local file directory.

Click "Browse" to select a file for upload from your PC. Once selected, then click "Upload" to complete the process.

To save a copy of the selected program back to your PC, right click on the selected file link (upper right), and then select "save link as" or "save target as" (varies by browser).

Note: There may be a delay of several seconds while Flash memory is updated by some of the above actions.



This page allows you to view/edit program files.

File: /FS/FLASH0/zigbee.sb

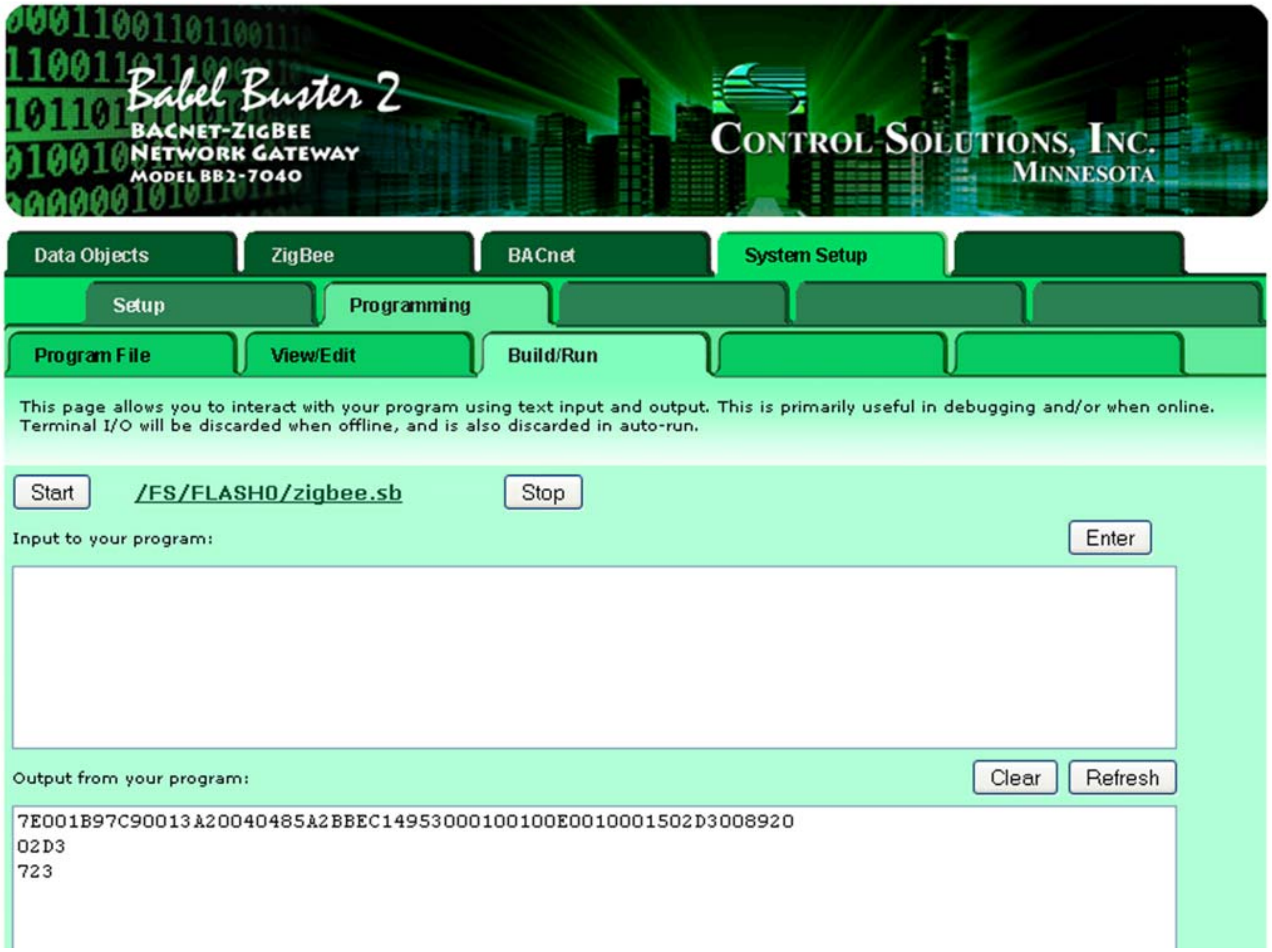
```
FUNCTION unstuff($v)
e = 0
for i = 1 to 4
c = asc(mid($v,i,1))
if c > 57 then c = c - 7
c = c - 48
e = e * 16 + c
next i
unstuff = e
END FUNCTION

open "API" for comm as 1
$api = "Remote," & stuff(201) & ",00:13:A2:00:40:48:5A:2B,Y,IS"
print #1, $api
timeout 10
line input #1, $r
print $r, "\n"
$hv = mid($r,53,4)
print $hv
n = unstuff($hv)
print "\n", n
setreg 4, n/10
end
```

Click "Get" to get the source code for the program shown. Edit using standard text editing (cut, paste, etc). Click "Save" to save changes back to selected file shown.

To create a new file, start by creating a new file in the Program File page. This file will initially be empty, but is needed as a place holder so you have somewhere to save the text on this page when you click "Save".

To erase the scratch pad (shown here), check the erase box and click "Clear".

HTML cannot post large text buffers, therefore larger text files will be displayed for editing in text "pages" within this web page. Use the Page Down and Page Up buttons to scroll through the file. Within each text page, use the scroll bars to navigate through the current text page.



This page allows you to interact with your program using text input and output. This is primarily useful in debugging and/or when online. Terminal I/O will be discarded when offline, and is also discarded in auto-run.

Input to your program:

Output from your program:

```
7E001B97C90013A20040485A2BBEC14953000100100E0010001502D3008920
02D3
723
```

This page emulates terminal I/O interaction with your Basic program as best an HTML page can.

Start the program by clicking "Start". While running, input you type in the input window will be made available to your program each time you click "Enter". Each time you click "Refresh", the output window will be updated with any output generated by your program since the last time you clicked "Refresh". Click "Clear" to discard old output.

Note: The "Enter" and "Refresh" buttons have the same effect as they are both submit buttons. Input present in the input box will be submitted when you click any button.

11.2 Script Basic Enhancements Specific to ZigBee

The ZigBee-specific statements are found only in BB2-7040 version of Script Basic. Communication with a ZigBee device is done via the file interface in Basic. The ZigBee API channel is opened as a file, and ASCII strings are printed to and input from this file.

Opening the ZigBee COM port: Any file number may be used, but file #1 is used in the following example(s).
Program lines:

```
open "API" for comm as 1
open "API+" for comm as 1
```

The first example using "API" as file name will expect API packets to all use the 0x17 remote API command, and will be filtered such that received packets are only returned to Basic when they contain the 0x97 reply code and reference a frame ID generated by Basic. The second example using "API+" will result in Basic receiving unqualified packets, including 0x92 packets generated voluntarily by sensors, etc.

A program code snippet that will send the IS command to a remote device and print its reply is as follows:

```
    $api = "Remote,C9,00:13:A2:00:40:2D:37:7A,Y,IS"
    print #1, $api
    timeout 10
    line input #1, $r
    print $r, "\n"
```

The format for the API string is as follows:

```
"Local,FF,CC,hhhhhh..hh"
"Remote,FF,Vnn,Y,CC,hhhhhh..hh"
"Remote,FF,00:00:00:00:00:00:00:00,Y,CC,hhhhhh..hh"
"Raw,hhhhhh..hh"
```

The first comma separated field must be one of Local, Remote, or Raw. The FF must be a frame number, in hexadecimal, in the range of 201-255. Frame numbers outside this range will not be routed back to Basic. The next field, for Remote API only, is a device designation, either a 64-bit device address formatted as illustrated, or a device number Vnn to look up on the ZigBee Client Devices page where nn is 1 to 50. Following the device, again in Remote API only, is a field which must be either Y or N. If 'Y', then the AC immediate bit is sent (accept changes). For Local or Remote API, the CC field is the 2-character AT command to be sent. In all cases, the last field(s) represent the data payload (which can be omitted for API commands with no parameters).

The hhhhhh.hhh may actually be one or more fields. For each comma separated field following CC, the value will be interpreted as one or more bytes of data in ASCII hexadecimal form, or be interpreted as multiple bytes of an ASCII string with one byte per character. If the first character of the string is not a hexadecimal digit, it will be taken as an ASCII string. If the first character in the comma separated field is a hexadecimal digit, the string will be parsed as hexadecimal data with one packet byte per two ASCII characters. Note that this interpretation differs from the API web page behavior.

An example of a string that will set the Network Interface name of a remote device is as follows:

```
    $api = "Remote,c9,00:13:a0:00:4d:45:fa:af,Y,NI,Temp Sensor 1
```

For a complete discussion of the API command set, refer to the Digi International/Maxstream XBee PRO documentation.

The following program example sends an on-demand sample command to a Digi SmartPlug and parses a data value from the reply.

```
FUNCTION stuff(a)
if (a > 255) then
$h = "FF"
else
$k = hex(a)
if len($k) = 1 then
$h = "0" & $k
else
$h = $k
end if
end if
stuff = $h
END FUNCTION

FUNCTION unstuff($v)
e = 0
for i = 1 to 4
c = asc(mid($v,i,1))
if c > 57 then c = c - 7
```

```
c = c - 48
e = e * 16 + c
next i
unstuff = e
END FUNCTION

open "API" for comm as 1
$api = "Remote," & stuff(201) & ",00:13:A2:00:40:48:5A:2B,Y,IS"
print #1, $api
timeout 10
line input #1, $r
print $r, "\n"
$hv = mid($r,53,4)
print $hv
n = unstuff($hv)
print "\n", n
setreg 4, n/10

end
```

12 Object Properties

12.1 Data Object Properties (Analog, Binary, Multi-state)

The following properties are found in the Analog, Binary, and Multi-state types of Input, Output, and Value objects. Some properties apply only to certain object types as noted where applicable.

Property	Encoding
Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77) (W)	CharacterString “Analog Input *n*”
Object_Description (28) (W)	Character String Same as Object_Name, is only alias for Object_Name
Object_Type (79)	BACnetObjectType ENUMERATED: analog-input (0) analog-output (1) analog-value (2) binary-input (3) binary-output (4) binary-value (5) device (8) multi-state-input (13) multi-state-output (14) multi-state-value (19)
Present_Value (85) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objets) (no index) (priority required when writing commandable objects) (input objects writeable only when out of service)
Status_Flags (111)	BACnetStatusFlags BIT STRING: fault(1), out-of-service(3)
Event_State (36)	BACnetEventState ENUMERATED: normal(0), fault(1)
Reliability (103)	BACnetReliability ENUMERATED: normal(0) *Vendor specific:* BACnet device timeout (82) BACnet error returned by server/slave BACnet device (83) ZigBee device timeout (84) ZigBee device error (85) ZigBee reported invalid command (86) ZigBee reported invalid parameter (87) Other (99)
Out_Of_Service (81) (W)	BOOLEAN
COV_Increment (22) (W)	REAL (analog objects only)
COV_Period (180) (W)	Unsigned

Priority_Array (87)	BACnetPriorityArray (commandable objects only) SEQUENCE SIZE (16) OF BACnetPriorityValue REAL (each element, analog output objects) ENUMERATED (each element, binary output objects) Unsigned (each element, multi-state output objects)
Relinquish_Default (104) (W)	REAL (analog objects) ENUMERATED (binary objects) Unsigned (multi-state objets)
Polarity (84)	BACnetPolarity (binary objects only) ENUMERATED: normal(0)
Number_Of_States (74)	Unsigned (multi-state objects only)
Units (117)	BACnetEngineeringUnits (analog objects only)

12.2 Device Object Properties

The following properties are found in the Device object of the BB2-7040. In addition to standard Device properties, configuration properties that apply at a system level to the whole device are included here.

Property	Encoding
Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString
Object_Type (79)	BACnetObjectType
System_Status (112)	BACnetDeviceStatus
Vendor_Name (121)	CharacterString
Vendor_Identifier (120)	Unsigned16 (should always return 208)
Model_Name (70)	CharacterString
Firmware_Revision (44)	CharacterString
Application_Software_Version (12)	CharacterString
Protocol_Version (98)	Unsigned
Protocol_Revision (139)	Unsigned
Protocol_Services_Supported (97)	BACnetServicesSupported
Protocol_Object_Types_Supported (96)	BACnetObjectTypesSupported
Object_List (76)	BACnetARRAY[N] of BACnetObjectIdentifier
Max_APDU_Length_Accepted (62)	Unsigned
Segmentation_Supported (107)	BACnetSegmentation

APDU_Timeout (11)	Unsigned
Number_Of_APDU_Retries (73)	Unsigned
Device_Address_Binding (30)	List of BACnetAddressBinding
Database_Revision (155)	Unsigned

13 Trouble Shooting

13.1 Error Codes, Reliability Codes

Error codes returned by a Modbus device are encoded into reliability codes presented by the respective object. Reliability codes (read via the Reliability property) are as follows:

BACnet device timeout (82)
BACnet error returned by server/slave BACnet device (83)
ZigBee device timeout (84)
ZigBee device error (85)
ZigBee reported invalid command (86)
ZigBee reported invalid parameter (87)
Other (99)

13.2 Auto-Reset Errors

Reliability codes will “latch” by default and require that you read the Reliability property in order to reset it to zero, assuming the problem has gone away. Once the non-zero reliability code has been read (by reading the Reliability property), it will reset to zero the next time the object is updated, provided the problem has been resolved.

Since many systems do not automatically read Reliability codes, but do automatically respond to the Fault Status Flag associated with the non-zero reliability code, an auto-reset option is available. When set, reliability codes will return to zero as soon as the problem has been resolved, regardless of whether the non-zero reliability code was ever acknowledged by reading it.

Appendix A Connecting Control Solutions Wireless Devices

Control Solutions uses Digi XBee Series 2 ZigBee OEM modules with standard Digi endpoint firmware. Complete details about the XBee modules may be found on Digi's web site. A link to the XBee manual may be found on Control Solutions web site, on the BB2-7040 support page.

A.1 Configuring the Temperature Sensor

You can use the Network Identifier method of identifying sensors. If you use this method, you will need to use the NI command as follows, where ---name--- is replaced with the name of your choice:

```
/NI “---name---“
```

IMPORTANT: There is a potential battery life problem when using the Network Identifier approach. Please see section 5.1 for more details about this problem.

The ZB25-TH temperature/humidity sensor has a thermistor on input 1, and humidity sensor on input 2. The ZB25-T temperature (only) sensor has a thermistor on input 1. Use the /D1 command for either sensor. Use the /D2 only if you also have humidity. The following is the set of commands that will be needed to configure the sensor:

```
/SN 00 3B
/SP 01 F4
/WH 00 64
/D1 02
/D2 02
/IR FF FF
/ST 00 96
/WR
```

The data parse mask for temperature & humidity are as follows (respectively):

```
92----------------------------06TTTT----
92----------------------------06----XXXX
```

The data parse mask for temperature only (no /D2 or /D2 0) is as follows:

```
92----------------------------02TTTT
```

Temperature is returned as degrees Celsius x 100. Use scale of 0.01 and offset 0 for degrees C, or scale 0.018 and offset 32 to convert to degrees F. Humidity is a raw count between 0 and 1023 representing 0-100% RH. For humidity, use scale 0.097752 and offset 0.

Note: The “TT” parse field is unique and specific to Control Solutions ZB25 series sensors. It linearizes a 3K type 2 thermistor reading with 1.2V excitation connected directly to the A/D input of the XBee radio.

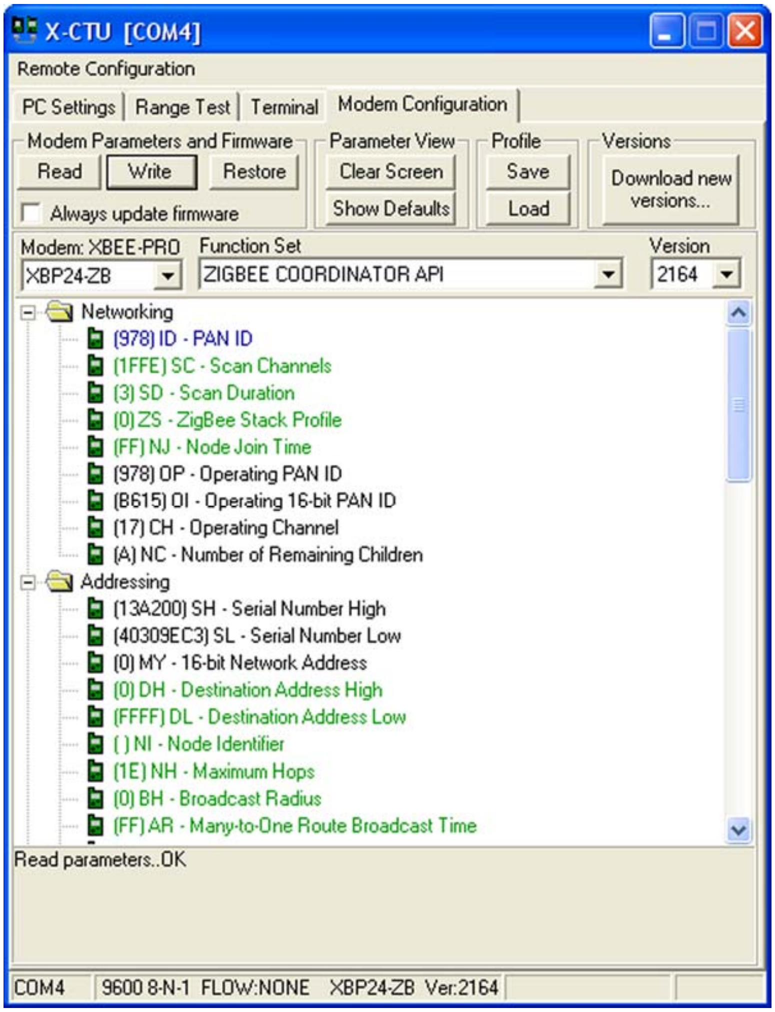
A.2 Configuring the Router

No configuration is needed other than setting the PAN ID and sleep period. Use the /SP command to set the sleep period to a value at least as large as the longest period set in any sensor (or endpoint). The router never sleeps, but needs to know the sleep period used by endpoints so that the router knows how long to wait for a reply from the endpoint.

IMPORTANT: The most frequent configuration related cause of failure to communicate with a sensor is having the sensor's SP longer than the router. If you are having trouble, check the router's SP parameter first.

A.3 Generic Configuration and the X-CTU Profile

If you are configuring a generic XBee device using the X-CTU, pictured in the screen shot below, you should select the appropriate firmware version (XB24-ZB ZIGBEE ENDPOINT). Beyond that, you can use the default settings that come with that version of firmware. You can pre-program the above commands via X-CTU if you wish.



□ Always update firmware

Modem: XBEE-PRO XBP24-ZB

Function Set ZIGBEE COORDINATOR API

Version 2164

Networking
- (978) ID - PAN ID
- (1FFE) SC - Scan Channels
- (3) SD - Scan Duration
- (0) ZS - ZigBee Stack Profile
- (FF) NJ - Node Join Time
- (978) OP - Operating PAN ID
- (B615) OI - Operating 16-bit PAN ID
- (17) CH - Operating Channel
- (A) NC - Number of Remaining Children

Addressing
- (13A200) SH - Serial Number High
- (40309EC3) SL - Serial Number Low
- (0) MY - 16-bit Network Address
- (0) DH - Destination Address High
- (FFFF) DL - Destination Address Low
- ( ) NI - Node Identifier
- (1E) NH - Maximum Hops
- (0) BH - Broadcast Radius
- (FF) AR - Many-to-One Route Broadcast Time

Read parameters..OK

COM4 | 9600 8-N-1 FLOW:NONE | XBP24-ZB Ver:2164

Appendix B Connecting Digi International Wireless Devices

The Digi devices mentioned here may be purchased from Digi International.

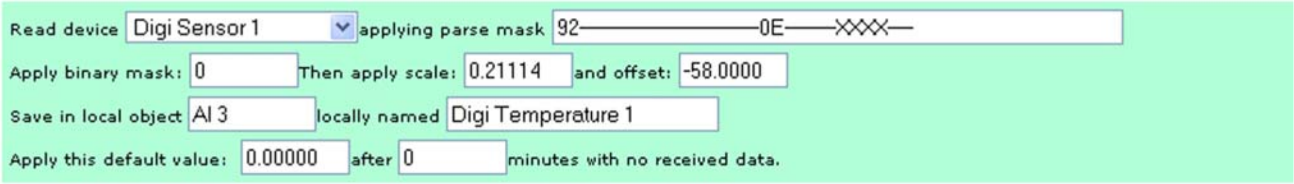
B.1 Configuring the Digi XBee Sensor

Send the following commands to the Digi Sensor, substituting your own PAN ID for the example shown.

```
/ID 00 00 00 00 00 00 02 34
/NI “---name---“
/SN 00 3B
/SP 01 F4
/D1 02
/D2 02
/D3 02
/IR FF FF
/ST 03 E8
/WR
```

The data parse masks will be as follows:

```
92----------------------------0E----XXXX--------  Light
92----------------------------0E--------XXXX----  Temperature
92----------------------------0E------------XXXX  Humidity
```

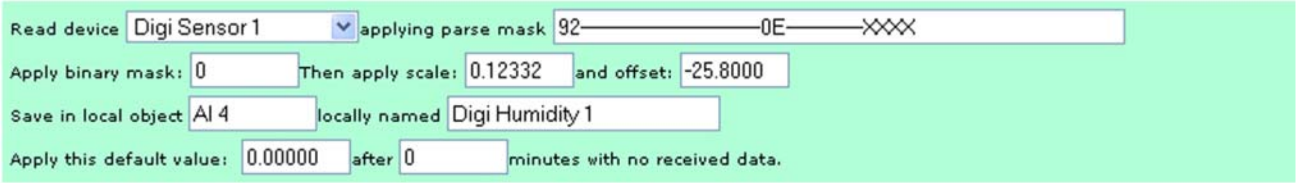


Read device Digi Sensor 1 applying parse mask 92————————0E———XXXX——

Apply binary mask: 0 Then apply scale: 0.21114 and offset: -58.0000

Save in local object AI 3 locally named Digi Temperature 1

Apply this default value: 0.00000 after 0 minutes with no received data.



Read device Digi Sensor 1 applying parse mask 92————————0E————XXXX

Apply binary mask: 0 Then apply scale: 0.12332 and offset: -25.8000

Save in local object AI 4 locally named Digi Humidity 1

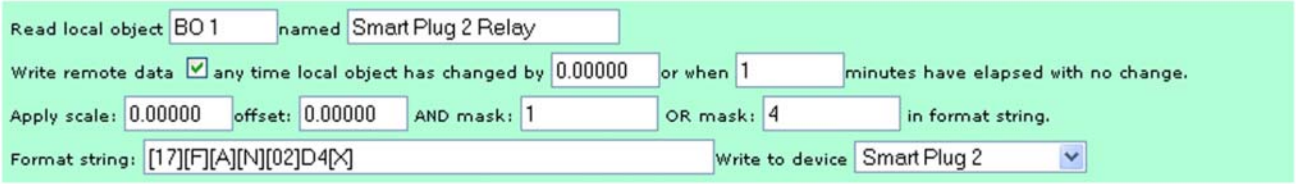
Apply this default value: 0.00000 after 0 minutes with no received data.

B.2 Configuring the Digi XBee SmartPlug

Set the PAN ID and Network Identifier. The write map’s Data Format Mask will be as follows:

[17][F][A][N][02]D4[X]

This assumes the BACnet object mapped will have a value of 0 or 1, and further assumes an AND mask of 1 and an OR mask of 4.



Read local object BO 1 named Smart Plug 2 Relay

Write remote data ☑ any time local object has changed by 0.00000 or when 1 minutes have elapsed with no change.

Apply scale: 0.00000 offset: 0.00000 AND mask: 1 OR mask: 4 in format string.

Format string: [17][F][A][N][02]D4[X] Write to device Smart Plug 2

B.3 Configuring the Digi Wall Router

You will only need to set the PAN ID and sleep period in the router. Use the /SP command to set the sleep period to a value at least as large as the longest period set in any sensor (or endpoint). The router never sleeps, but needs to know the sleep period used by endpoints so that the router knows how long to wait for a reply from the endpoint.

Appendix C Connecting Point Six Wireless Devices

ZigBee sensors from Point Six Wireless (Point Six, Inc.) are compatible with the BB2-7040 but require some special handling. The Point Six devices use pin sleep mode under control of a second small microprocessor, and therefore will not recognize the remote API commands the BB2-7040 is capable of sending. You must make sure the Point Six sensor is configured properly at the factory, and/or be able to use the Point Six configuration tools provided by Point Six. The standard ZigBee BB2-7040 is unable to configure the Point Six ZigBee devices. (If you are a ZigBee/XBee expert and have a Digi USB/XBee programming board, you can remove the XBee module from the Point Six unit and reprogram its parameters that way.) (OEM versions of the BB2-7040 for Point Six Wireless devices, including XSC 900 MHz, may be available. Contact sales@csimn.com.)

C.1 Changes Required in the BB2-7040

As of this writing, Point Six ZigBee sensors were being shipped configured for stack profile 2, while as of this writing Digi sensors are shipped configured for stack profile 0, and XBee modules in general default to stack profile 0. The BB2-7040 is currently shipped configured for profile 0.

Since reconfiguring the Point Six sensor might not be convenient, you can reconfigure the BB2-7040 to match it instead. To configure the BB2-7040 for stack profile 2, use the following commands from the Local API web page of the BB2-7040:

```
/ZS 2
/WR
```

Then restart the gateway. You may also want to reconfigure the PAN ID before restarting. To set the PAN ID of the gateway, include the following command before the /WR command:

```
/ID 00 00 00 00 00 00 12 34
```

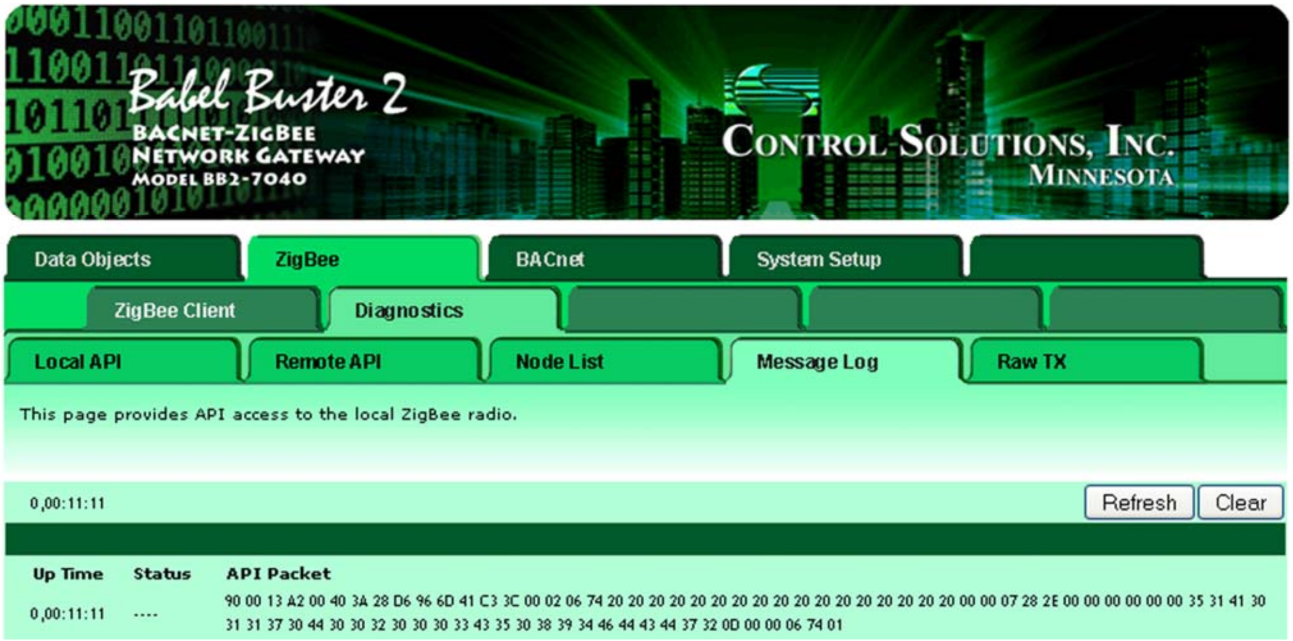
This example will set the PAN ID to 1234 (hex).

C.2 Compatibility Between Point Six and Digi Sensors

Using both Point Six and Digi sensors on the same network, or any other generic XBee product, will require that you reconfigure one or the other for the same stack profile. All stack profiles must be the same on the same network in order for devices to communicate. You can use the /ZS command in the Remote API page to set the stack profile of remote devices. Be sure to follow it with /WR to save the change past the next power cycle.

C.3 Configuring BB2-7040 to Recognize Point Six Sensors

Once the BB2-7040 is configured to the same stack profile and PAN ID as the Point Six sensor, you will see a data packet as illustrated below appear, after the sleep period has expired. The Point Six sensor data packet includes the sensor's serial number in ASCII code. The search string method of device recognition may be used to identify these sensors.

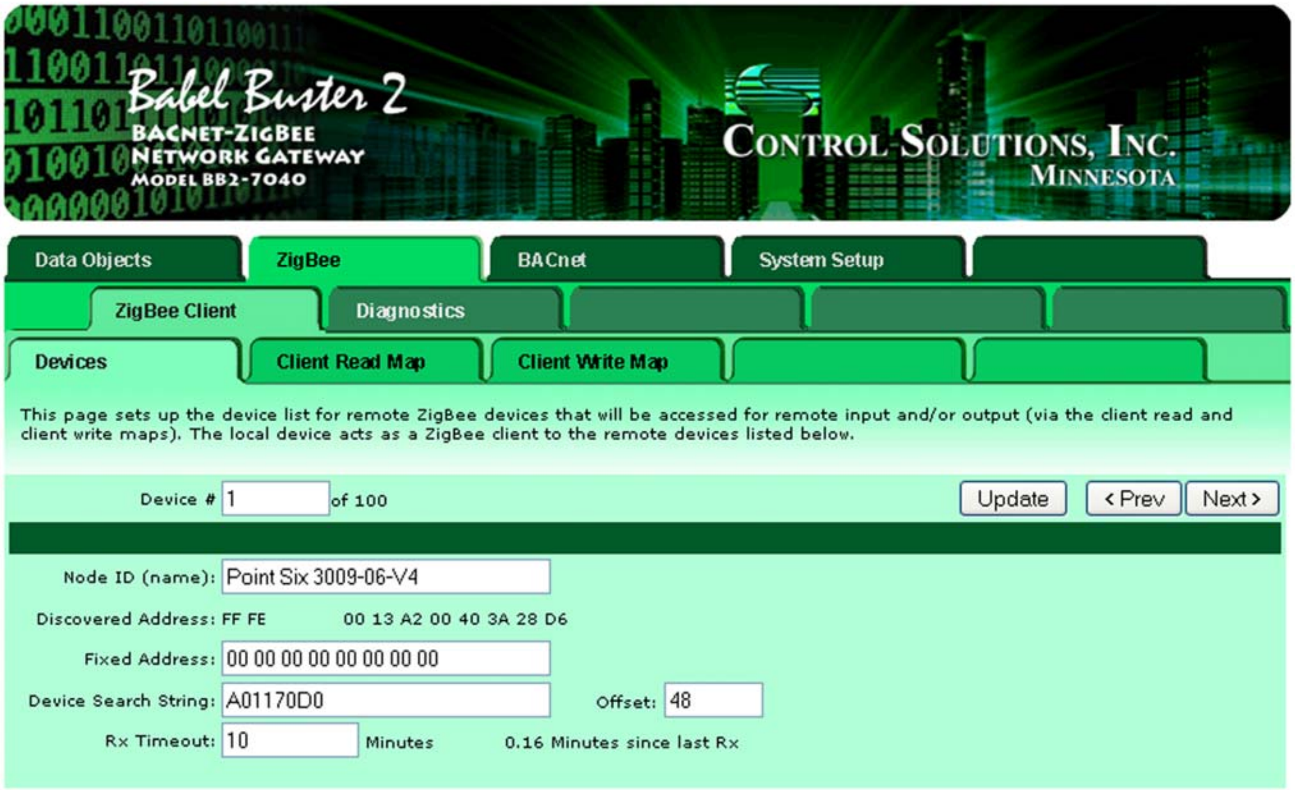


This page provides API access to the local ZigBee radio.

0,00:11:11

Up Time	Status	API Packet
0,00:11:11	....	90 00 13 A2 00 40 3A 28 D6 96 6D 41 C3 3C 00 02 06 74 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 00 00 07 28 2E 00 00 00 00 00 00 35 31 41 30 31 31 37 30 44 30 30 32 30 30 30 33 43 35 30 38 39 34 46 44 43 44 37 32 0D 00 00 06 74 01

The serial number taken from the label on the back of the Point Six sensor may be entered as the device search string (with offset 48) in the device list. Once this string is recognized, the device's 64-bit address will be listed here.



This page sets up the device list for remote ZigBee devices that will be accessed for remote input and/or output (via the client read and client write maps). The local device acts as a ZigBee client to the remote devices listed below.

Device # 1 of 100

Node ID (name): Point Six 3009-06-V4

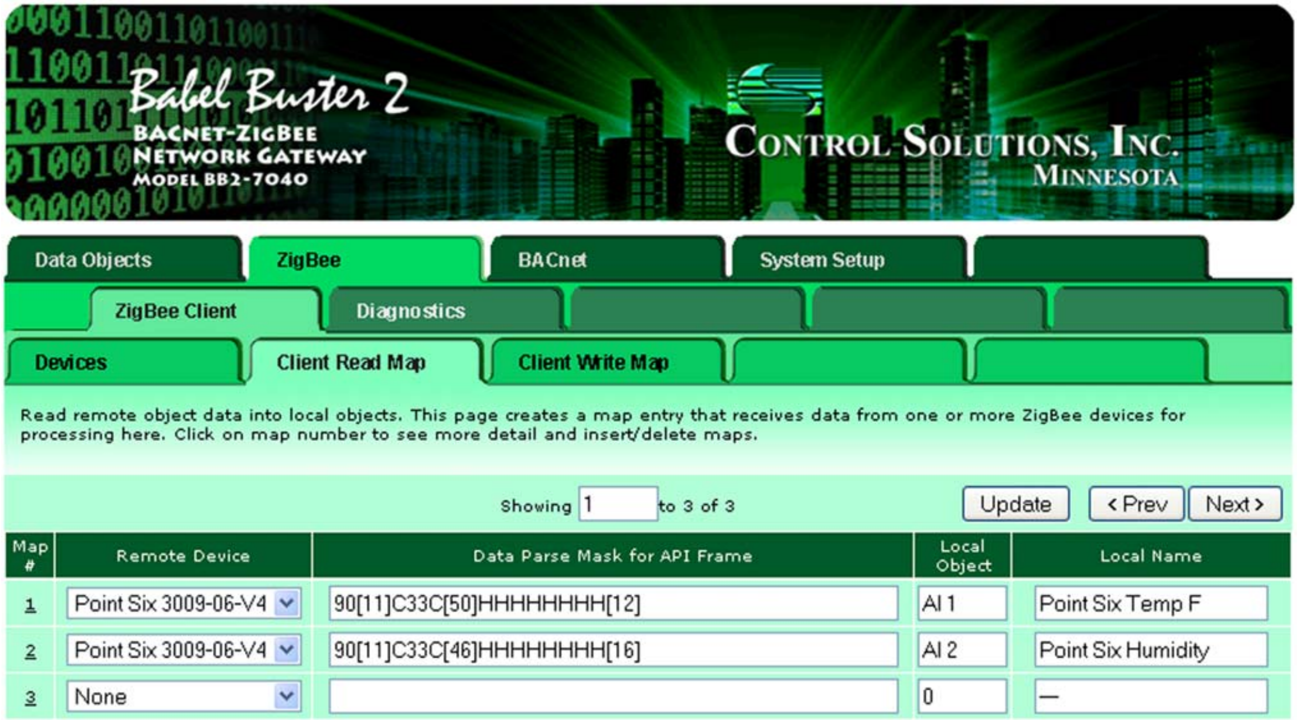
Discovered Address: FF FE 00 13 A2 00 40 3A 28 D6

Fixed Address: 00 00 00 00 00 00 00 00

Device Search String: A01170D0 Offset: 48

Rx Timeout: 10 Minutes 0.16 Minutes since last Rx

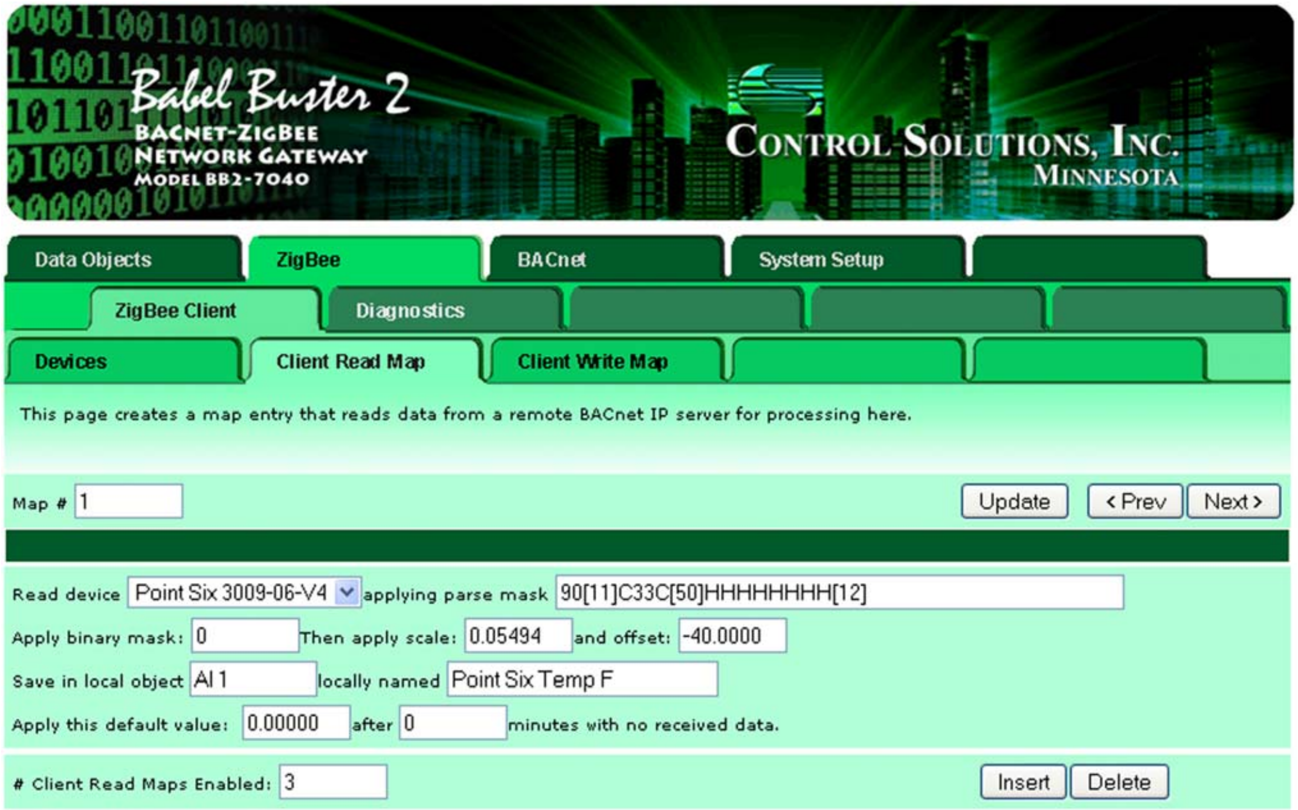
The screen shots below illustrate the data parse mask for temperature and humidity data fields from the Point Six sensor. The expanded view also includes the slope and intercept for obtaining relative humidity as a percent, and temperature in degrees Fahrenheit.



Read remote object data into local objects. This page creates a map entry that receives data from one or more ZigBee devices for processing here. Click on map number to see more detail and insert/delete maps.

Showing 1 to 3 of 3

Map #	Remote Device	Data Parse Mask for API Frame	Local Object	Local Name
1	Point Six 3009-06-V4	90[11]C33C[50]HHHHHHHH[12]	AI 1	Point Six Temp F
2	Point Six 3009-06-V4	90[11]C33C[46]HHHHHHHH[16]	AI 2	Point Six Humidity
3	None		0	—



This page creates a map entry that reads data from a remote BACnet IP server for processing here.

Map # 1

Read device Point Six 3009-06-V4 applying parse mask 90[11]C33C[50]HHHHHHHH[12]

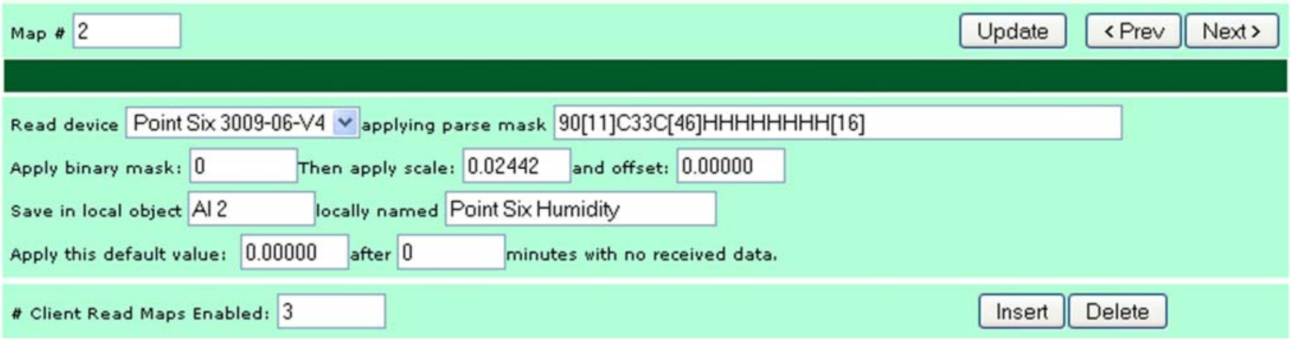
Apply binary mask: 0 Then apply scale: 0.05494 and offset: -40.0000

Save in local object AI 1 locally named Point Six Temp F

Apply this default value: 0.00000 after 0 minutes with no received data.

Client Read Maps Enabled: 3

The conversion factor for degrees Celsius is scale (slope): 0.03053, offset (intercept): -40.00.



Map # 2

Read device Point Six 3009-06-V4 applying parse mask 90[11]C33C[46]HHHHHHHH[16]

Apply binary mask: 0 Then apply scale: 0.02442 and offset: 0.00000

Save in local object AI 2 locally named Point Six Humidity

Apply this default value: 0.00000 after 0 minutes with no received data.

Client Read Maps Enabled: 3

Appendix D Gateway Setup Quick Start

There are just a couple of settings that need to be made to the ZigBee radio in the BB2-7040 to get started. One is to set the PAN ID, and the other is to set the sleep period.

The BB2-7040 is shipped with its ZigBee radio set as a coordinator, and with PAN ID set to 0, which means it will automatically pick a random PAN ID at power-up.

To set the PAN ID to a chosen fixed ID, go to the ZigBee Diagnostics tab set, and select Local API. Enter the command:

```
/ID 00 00 00 00 00 00 12 34
```

if you want the PAN ID to be hex 1234 (for example). The PAN ID is 8 bytes, and these are entered in the API command window as 1 or 2 digit hex numbers separated by spaces.

After setting the PAN ID, send the /WR command to save this ID to non-volatile memory.

You will never see actually "OK" come back as noted in the command set document. What you will get is the API equivalent, which will be the ASCII code for the command, followed by a byte of 00 if all is well. The last byte will be an error code if it did not execute right. You need to click the Check button to see the reply at least some of the time (most of the time on the Remote API page).

Status codes for XBee endpoint devices using API mode are as follows:

00	No error
01	Error (unspecified)
02	Invalid Command
03	Invalid Parameter
04	Remote command transmission failed.

ZigBee devices that are set to the same PAN ID as your BB2-7040, or are set to "join any" PAN, will associate themselves with the the BB2-7040. To see what devices are known to the BB2-7040, go to the Node List tab, click Discover, followed by Refresh a time or two (after brief delay). The addresses of the nodes will be listed (after some longer delay if sleep periods are set long).

Important note about sleep period:

Use the /SP command to set the sleep period to a value at least as large as the longest period set in any sensor (or endpoint). The coordinator never sleeps, but needs to know the sleep period used by endpoints so that the router knows how long to wait for a reply from the endpoint. If you get errors, in particular error code 04 in a reply, it may indicate the /SP setting is shorter than the endpoint's /SP setting. If the suggested setting of /SP 01 F4 has been used on an endpoint, then /SP 01 F4 should be issued to the local radio via the Local API page. After using /SP, use the /WR command to make the setting persistent.

Revision History

1.0	March 2010	Initial Release
1.1	April 2010	Added Appendix D, notes about use of /SP in routers, Appendix A and B.
1.2	February 2011	Updated configuration information for Control Solutions ZB25 series sensors.