User Guide

Babel Buster Pro

Model BBPRO-V230
BACnet to SNMP Gateway
Rev. 1.1 – Dec. 2016

BBPRO-V230 User Guide Contents

1. Introduction

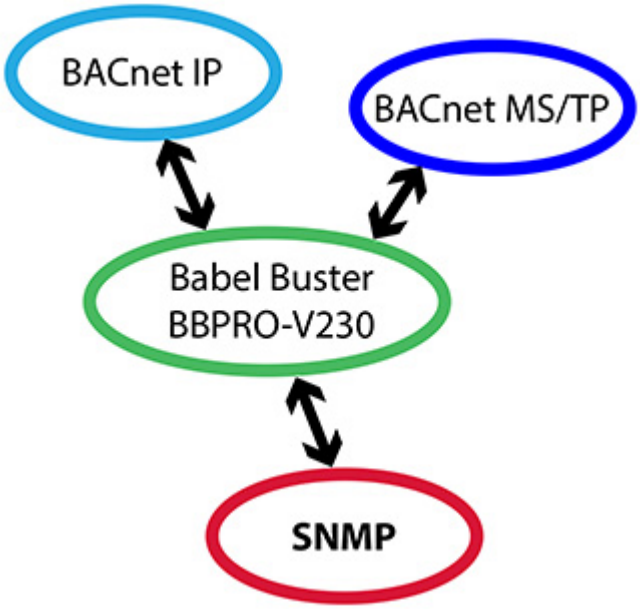
1.1 How to Use This Guide

This user guide provides background information on how the gateway works, and an overview of the configuration process. There are several sections for groups of tabs found in the web interface in the gateway which is accessed by opening a web browser and browsing to the IP address of the device.

You should at least read Sections 2 and 3, and other sections specific to your intended use. There is a "Quick Help" section at the bottom of each web page in the gateway which is generally sufficient for quick reference in setting up the gateway.

1.2 Overview of Gateway Device

The Babel Buster BBPRO-V230 is a BACnet gateway used primarily to interface between BACnet devices and SNMP devices. The gateway may be used as BACnet IP client and server, BACnet MS/TP master and slave, SNMP server (agent), and SNMP client (manager). The BBPRO-V230 includes additional special features allowing it to Walk SNMP tables, and receive traps from other SNMP devices. The BBPRO-V230 includes Script Basic for local programming to interpret trap messages that do not otherwise fall into a trap receive rule template.



The most common application for these gateways is interfacing a BACnet device to an SNMP network, or vice versa. Another common application is remapping objects from multiple slave devices to appear as a single device on the alternate network, mapping MS/TP to IP or vice versa.

1.3 What is New in Pro Version of Gateway

The most significant enhancements in the Pro gateway are the addition of extremely powerful trap receiver capabilities, and the MIB Walk capability. The MIB walker can traverse straight contiguous tables or sparse tables with wildcard fields in the OID to act upon for capturing data.

The BBPRO-V230 is useful for connecting SNMP enabled UPS systems that implement RFC 1628 or equivalent. These devices have alarm tables that cannot be simply polled in the traditional manner. The BBPRO-V230 includes intelligent algorithms for interpreting the alarm tables of such devices.

The BBPRO-V230 includes built-in Script Basic for user programming. Script Basic has access to the SNMP trap receiver data as well as BACnet objects (which inherently includes access to SNMP Client data). The primary applications for Script Basic are trap interpretation when the trap does not fit the trap rule template, and intepretation of non-standard Octet String SNMP data. Since Basic is general purpose in nature, one could potentially use it to implement control algorithms across multiple BACnet and/or SNMP devices.

1.4 Important Safety Notice

Proper system design is required for reliable and safe operation of distributed control systems incorporating any Control Solutions product. 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 ANY 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.

1.5 Warranty

This software and documentation 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 documentation or in the product(s) and/or the program(s) described in this documentation at any time. This product could include software bugs, technical inaccuracies, typographical errors, and the like. Changes are periodically made to the information herein; these changes may be incorporated in

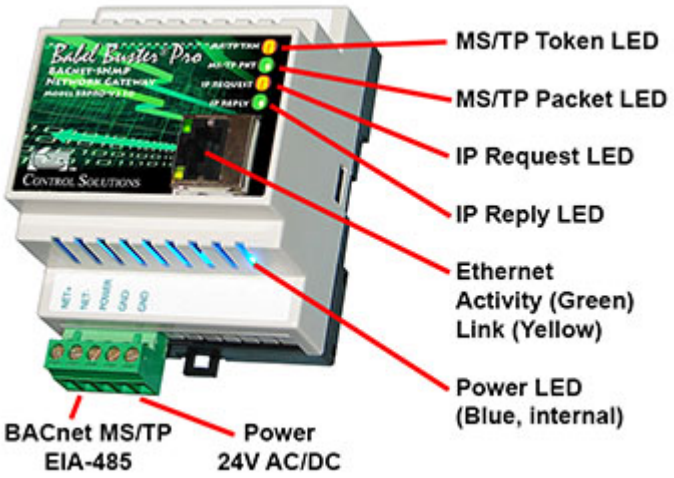
new editions of the software.

2. Connecting Gateway for the First Time

2.1 Connectors and Indicators

Follow these steps to make the initial connection to the BBPRO-V230.

(a) Connect power. Apply +12 to +24VDC or 24VAC to the terminal marked “POWER”, and common or ground to one of the terminals marked “GND”.



(b) Connect a CAT5 cable between the RJ-45 jack on the gateway, and your network switch or hub. You cannot connect directly to your PC unless you use a “crossover” cable.

(c) Apply power.

A blue LED inside the case should light indicating power is present.

If the link LED on the RJ45 jack is not on, check your Ethernet cable connections. Both link and activity LEDs on the RJ45 jack will be on solid for a short time during boot-up. The entire bootup process will take 1-2 minutes, during which time you will not be able to connect with a browser.

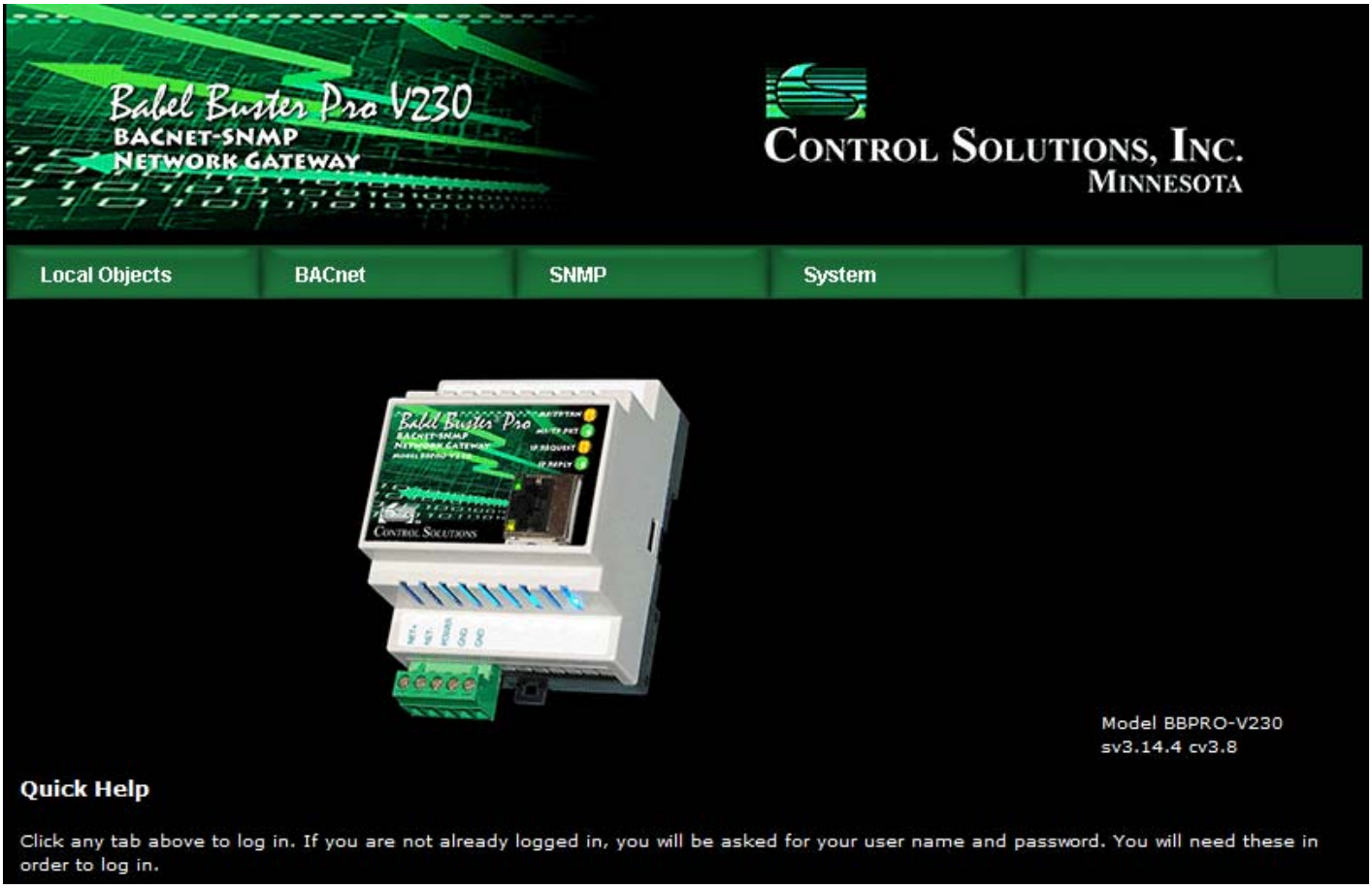
Ethernet link LED is the yellow LED integrated into the CAT5 connector. Ethernet activity LED is the green LED integrated into the CAT5 connector.

Refer to Appendix A for additional detail pertaining to connections and indicators as well as optional internal jumper settings.

2.2 Open Web User Interface

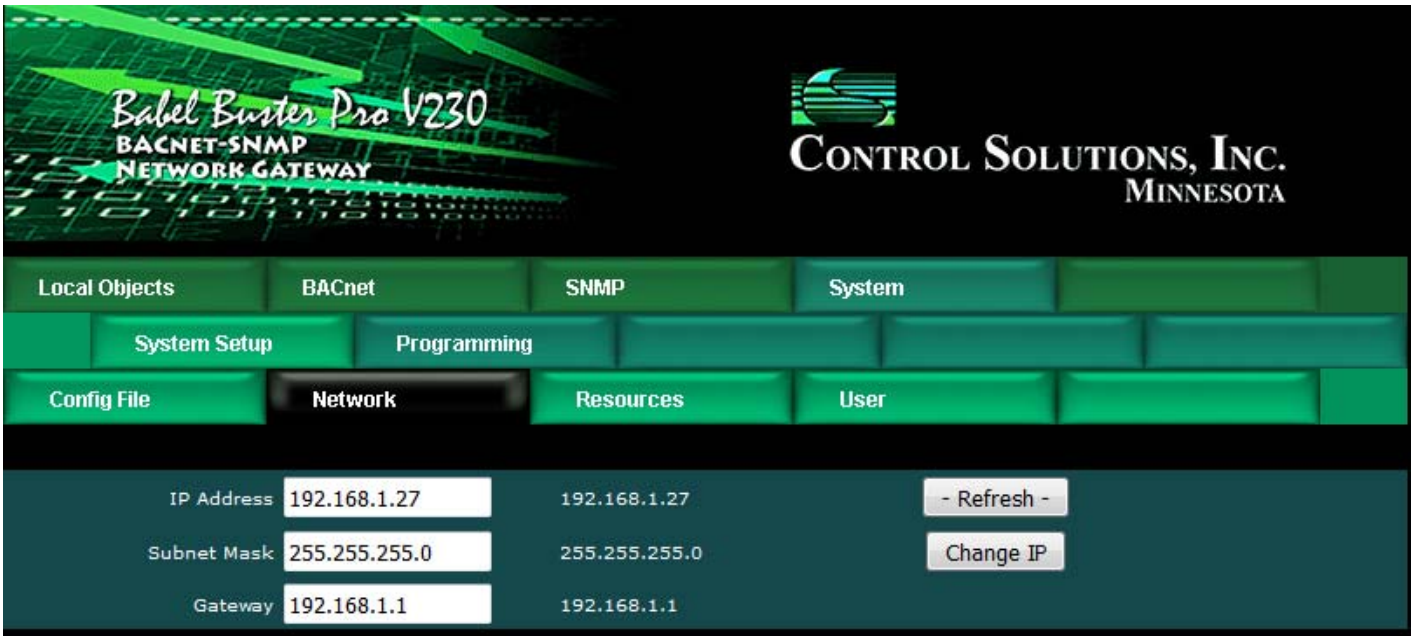
The default IP address as shipped is 10.0.0.101. Open your browser, and enter "http://10.0.0.101/" in the address window. You should see a page with the "Babel Buster Pro V230" header shown below. From this point, you will find help on each page in the web site contained within the product.

If your PC is not already on the 10.0.0.0 domain, and you are unable to connect, you may need to temporarily change your computer's IP address to a static IP address that starts with 10.0.0. and ends with anything but 101.



When you click on any of the page tabs such as System, you will be asked for a user name and password. The default login is user name "system" with password "admin". You can also log in as "root" using password "buster". You must be logged in as the root user to add other users.

To change the IP address of the gateway, go to the Network page under System :: System Setup. The following page should appear (only top portion illustrated here). Change the IP address, and subnet mask and gateway if applicable. Click Change IP to save the changes. The process of programming this into Flash takes around half a minute. The new IP address only takes effect following the next system restart or power cycle.



Most changes are stored in an XML configuration file in the device's Flash file system. Only a few are stored differently, and the IP address is one of those. Normally, clicking Update on any configuration page only stores that configuration information to a temporary RAM copy of the configuration file. To make your changes other than IP address permanent, you must click Save on the Config File page (System :: System Setup :: Config File). Refer also to section 14.1.

3. Configuring Local Objects

Babel Buster gateways do not come with a predefined set of BACnet objects. The gateway will initially have a handful of objects, but it is up to the user to allocate the number needed, up to the maximum of 1,000 total combined objects.

3.1 Behavior of Input vs Output Objects

The easiest way to keep track of input versus output is to think about a BACnet device's role in the system. The system will receive input from the BACnet device, and provide output to the BACnet device. Inside the BACnet device, hardware will physically associate BACnet Input Objects with sensor inputs such as temperature or pressure sensors, etc. The system then receives the sensor input information via BACnet Input Objects. When the system wants to control an actuator, it will send setpoints to the actuator via BACnet Output Objects. Hardware inside the BACnet device will physically associate the Output Object with a physical actuator such as valve position servo or motor speed controller.

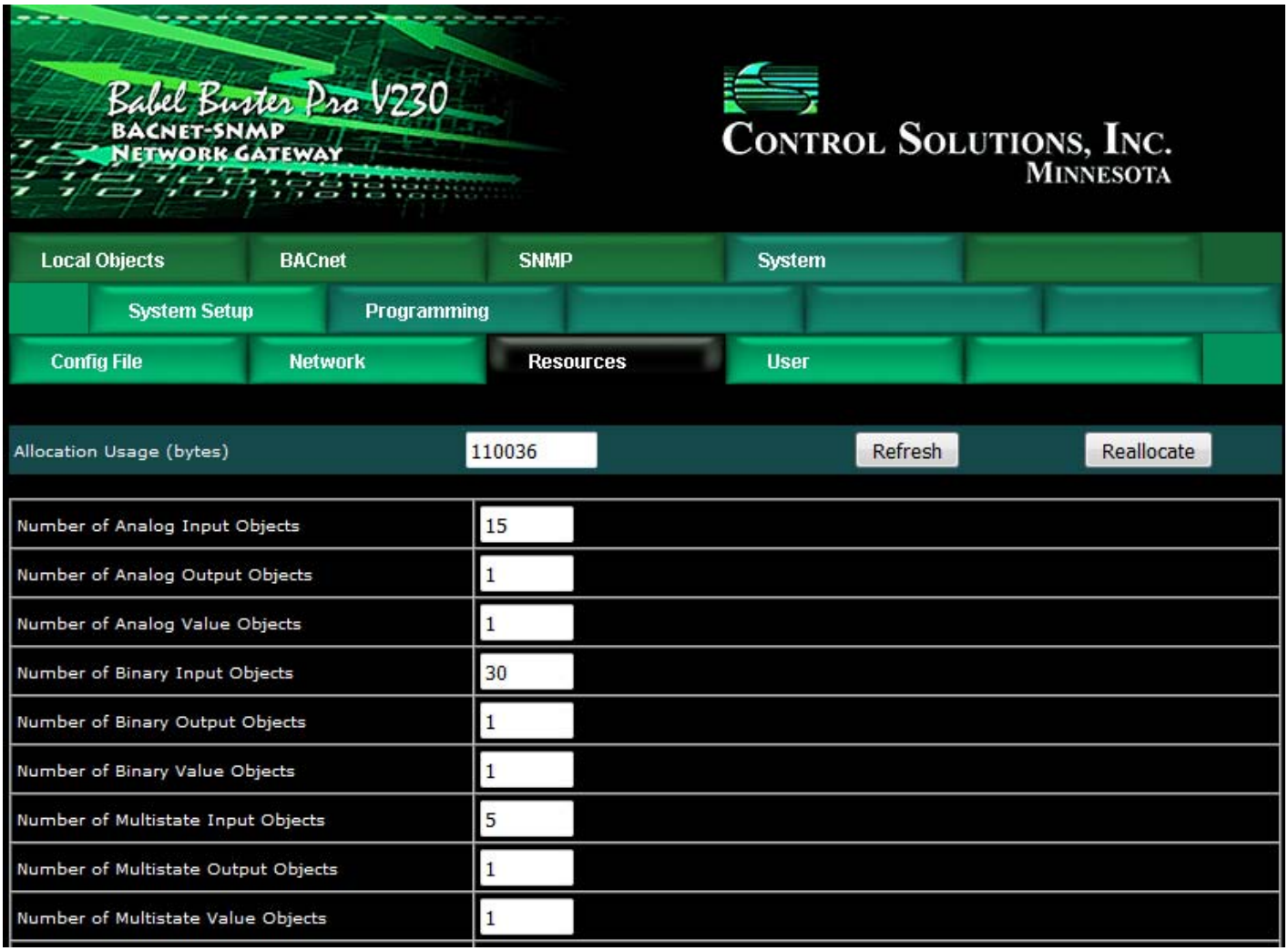
Keeping track of input versus output in a gateway can be a bit trickier; however, the choice of input versus output does not change from the BACnet perspective. Only the nature of the physical sensor and actuator hardware changes. In the case of the BBPRO-V230, sensors and actuators both consist of SNMP devices (from BACnet's perspective). Therefore, use a BACnet Output Object to send data to an SNMP device, and use a BACnet Input Object to receive data from an SNMP device.

The BACnet system can also interact with user written programs written in Basic, running inside the BBPRO-V230. The BACnet system would receive data from the program using Input Objects. The BACnet system would provide data to the program using Output Objects. Therefore, from the programmer's perspective, the program will write to Input Objects, and read from Output Objects.

We have not mentioned BACnet Value Objects yet just to avoid confusing the discussion. A Value Object can be input or output, or both at the same time. If you are familiar with Modbus, the BACnet Value Object is most synonymous with the holding register that you can both read and write. When using a Value Object, it is best to think about its role as input or output when deciding how to apply maps or rules in the gateway.

3.2 Allocating Local Objects

The resource allocation page is where you set the number of each type of available BACnet object that you will use. It is a good idea to determine ahead of time how many objects you will need, then allocate that number, possibly including a spare object or two. It is not a good idea to allocate a large number of objects that will remain unused since this simply clutters the screen when a front end system auto-discovers all objects in the device.



Clicking Refresh will update the page with the current counts. To change counts, enter new numbers, then click Reallocate. Next, without clicking Refresh, go to the Config File page and click Save to save your configuration file in the selected *.XML file. As soon as it is done saving the file, click Load to reload the file just saved. The new file will contain the new counts, and the resources (object counts) will be reallocated when the configuration file is reloaded.

3.3 Configuring Local Objects

There is a different page for each BACnet object type in the device. Objects are listed in tabular form with name and description, present value, reliability code and status. Additional information as applicable to the object type may also be listed.

Click on the object number in the first column to open the expanded view of that

object and gain access to its configuration.

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**upsEstimatedMinutesRemaining** Estimate of time to battery depleted	N	297.0000	0	0,0,0,0	minutes
2	**upsEstimatedChargeRemaining** Estimate of charge remaining	N	100.0000	0	0,0,0,0	percent
3	**upsOutputPercentLoad** Power capacity being used	N	0.000000	0	0,0,0,0	percent
4	**upsBatteryVoltage** Present battery voltage DC	N	27.10000	0	0,0,0,0	volts

Reliability codes may be any of the following:

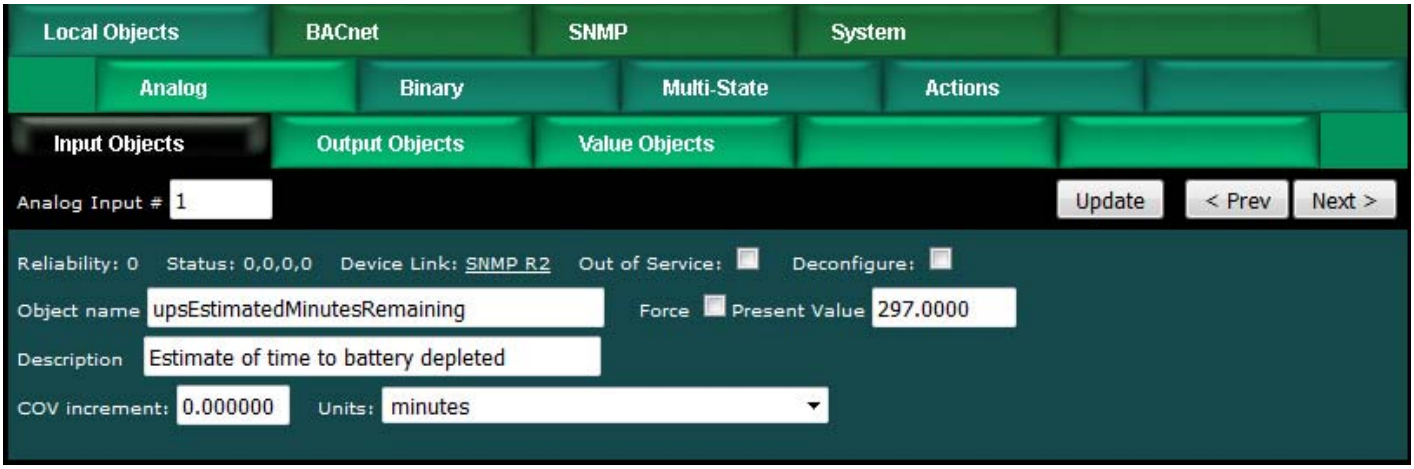
BACnet IP client, device timeout (82)
BACnet IP client, error returned by server (83)
SNMP client, no response from agent (91)
SNMP client, agent returned 'no such name' error (92)
SNMP client, unable to parse data returned (93)
SNMP client, reply did not match request (94)
SNMP client, invalid table walk configuration (95)
SNMP client, unable to parse PDU returned (96)
SNMP client, table walk came up short on data (97)
SNMP client, agent returned error other than 'no such name' (98)

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:

A = in alarm
B = fault
C = overridden
D = out of service

3.3.1 Analog Input Objects

The source of data for an Analog Input object will typically be reading from some other BACnet or SNMP device.



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for an Analog Input object will typically be reading from some other BACnet or SNMP device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

Out of Service means polling of the mapped remote 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 mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

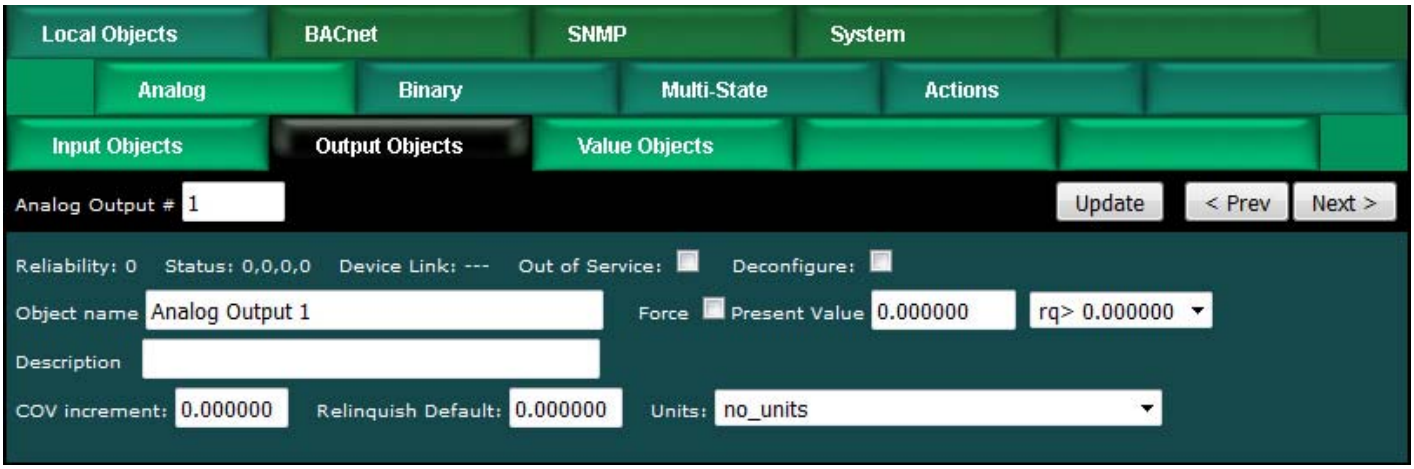
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.2 Analog Output Objects

The destination of data for an Analog Output object will typically be some other BACnet or SNMP device.



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The destination of data for an Analog Output object will be writing the remote BACnet or SNMP device via the map indicated by the Device Link. The remote device will be updated upon change of source data and/or periodically as defined by the Write Map.

The Analog Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm

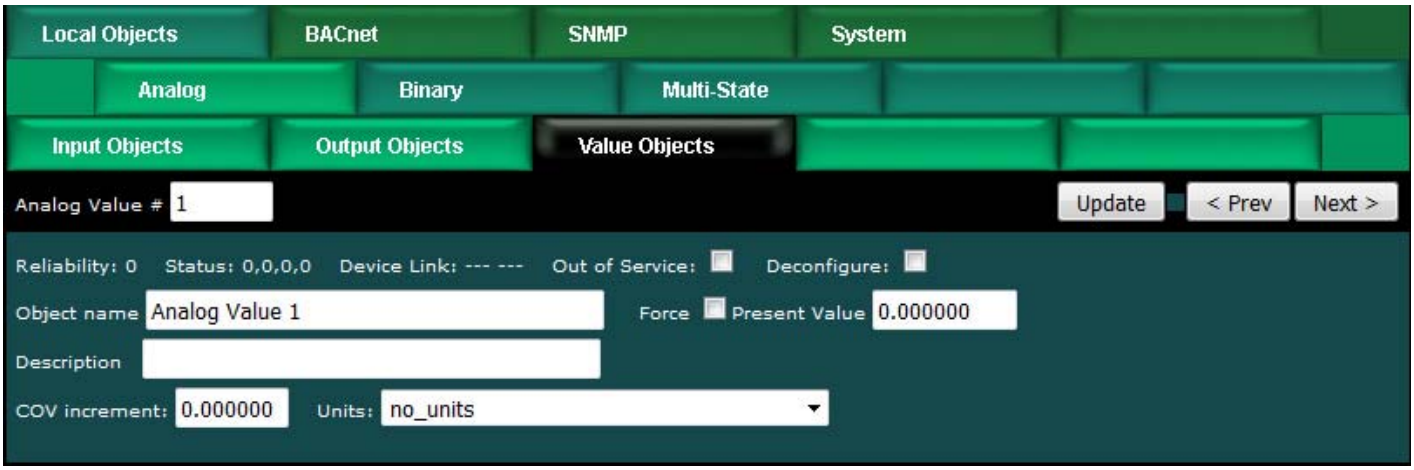
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.3 Analog Value Objects

Analog Value objects may be both a source and destination for some other BACnet or SNMP device.



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

Analog Value objects may be both a source and destination for some other BACnet or SNMP device. The source of data for the Value object will be reading from a remote device when associated with a Read Map. The destination of data for the Value object will be writing to a remote device when associated with a Write Map. If a remote device is mapped, the device links are displayed above. You may click on either link to view the respective mapping.

The Value object may be simultaneously associated with both Read and Write maps pointing to the same remote device object. When this Value object receives new data (from any source), this data will be written to the mapped remote device before any subsequent read from the same device. Thus the Value data is not discarded by the read operation before the new data can be written.

Out of Service means polling of the remote device will stop. While out of service, the present value may be written by an external BACnet client but it will not be written to any mapped remote device. Data may be forced via this web page at any time, but will be overwritten by the next read from a remote device unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

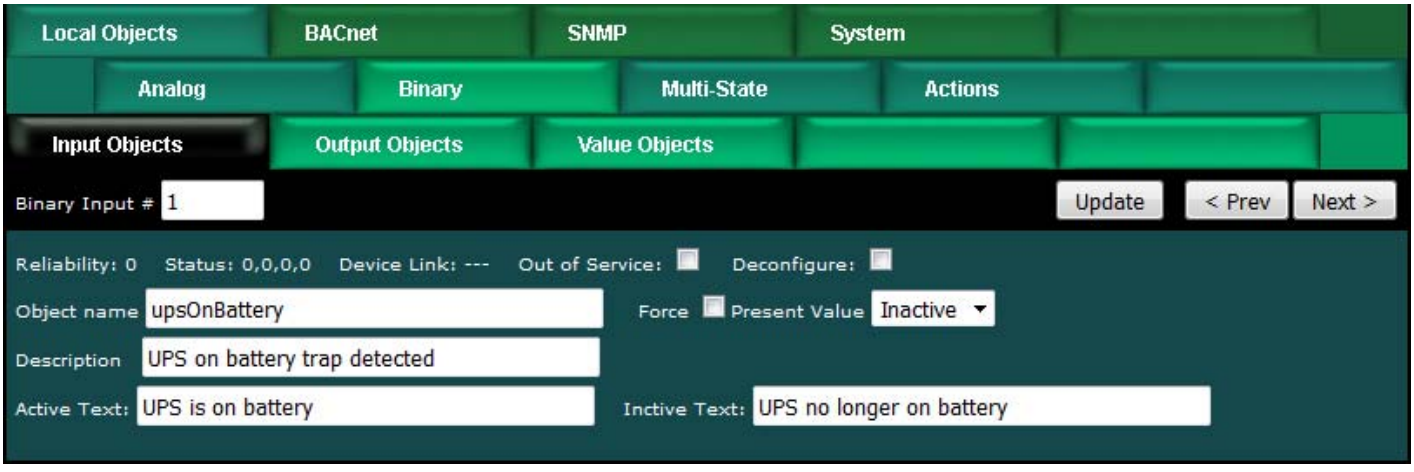
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.4 Binary Input Objects

The source of data for a Binary Input object will typically be reading from some other BACnet or SNMP device.



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for an Binary Input object will typically be reading from some other BACnet or SNMP device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

Out of Service means polling of the mapped remote 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 mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

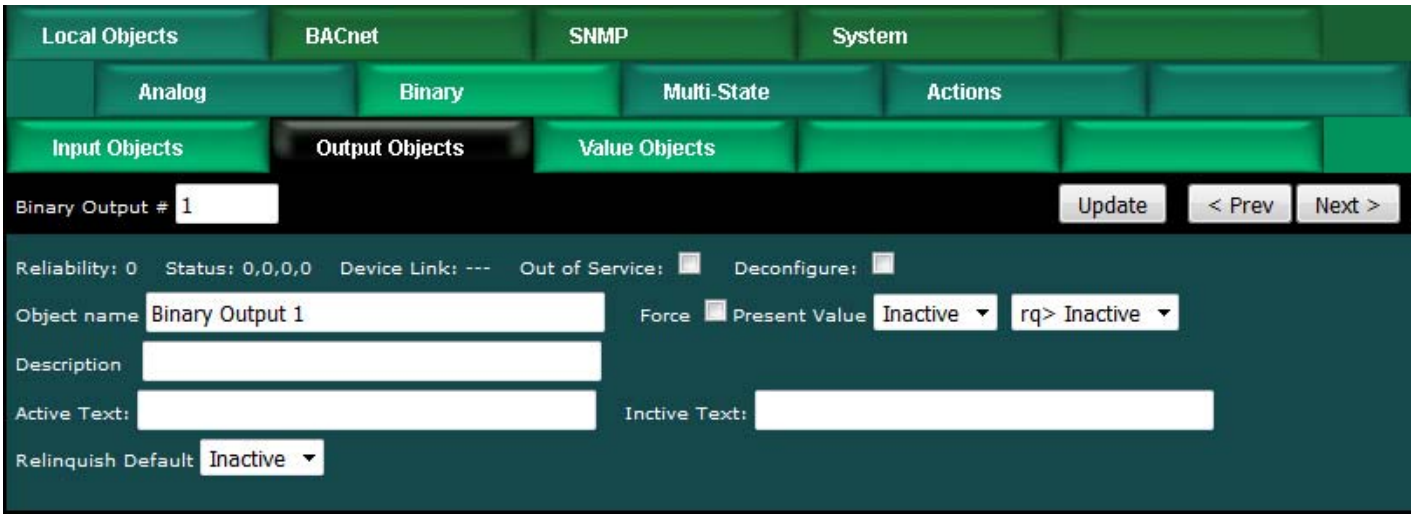
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.5 Binary Output Objects

The destination of data for a Binary Output object will typically be some other BACnet or SNMP device.



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The destination of data for a Binary Output object will be writing the remote BACnet or SNMP device via the map indicated by the Device Link. The remote device will be updated upon change of source data and/or periodically as defined by the Write Map.

The Binary Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

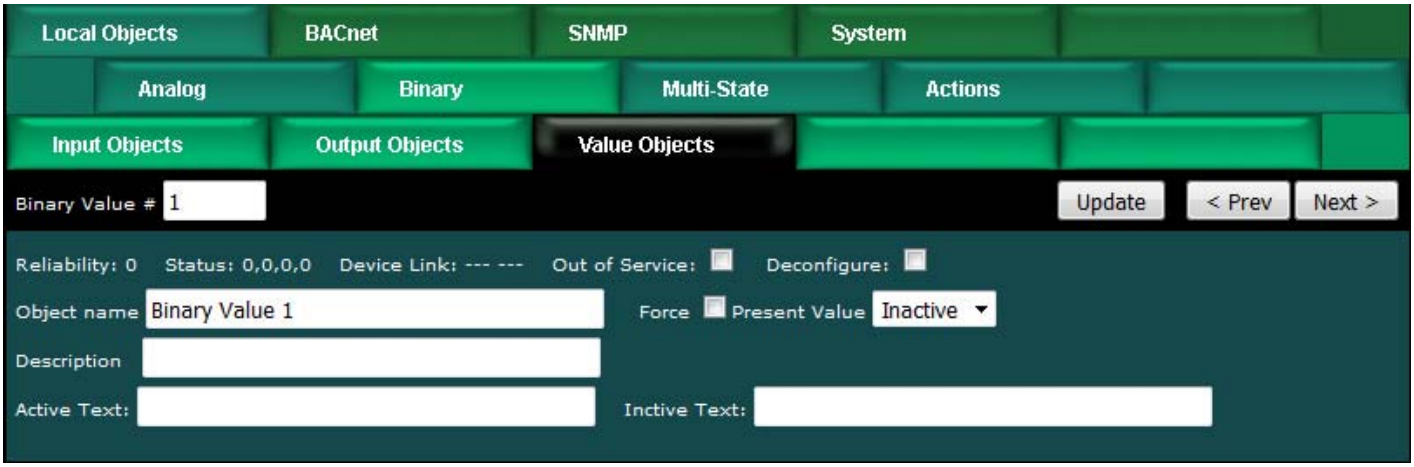
Device link will indicate BIP, SNMP, TRAP, or WALK, followed by R for read or W for

write, and a number which is the rule number in the table of read or write rules for mapping to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.6 Binary Value Objects

Binary Value objects may be both a source and destination for some other BACnet or SNMP device.



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

Binary Value objects may be both a source and destination for some other BACnet or SNMP device. The source of data for the Value object will be reading from a remote device when associated with a Read Map. The destination of data for the Value object will be writing to a remote device when associated with a Write Map. If a remote device is mapped, the device links are displayed above. You may click on either link to view the respective mapping.

The Value object may be simultaneously associated with both Read and Write maps pointing to the same remote device object. When this Value object receives new data (from any source), this data will be written to the mapped remote device before any subsequent read from the same device. Thus the Value data is not discarded by the read operation before the new data can be written.

Out of Service means polling of the remote device will stop. While out of service, the present value may be written by an external BACnet client but it will not be written to any mapped remote device. Data may be forced via this web page at any time, but will be overwritten by the next read from a remote device unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

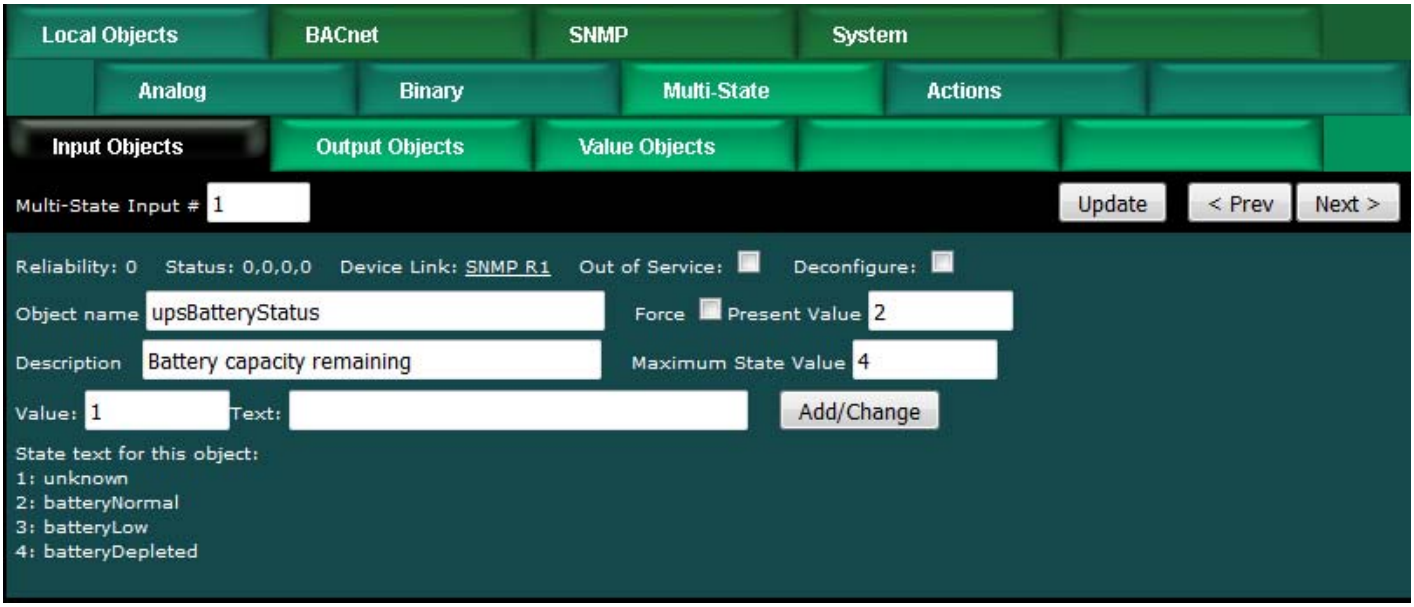
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.7 Multistate Input Objects

The source of data for a Multistate Input object will typically be reading from some other BACnet or SNMP device.



The object name, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. When changed, be sure to save the updated configuration by clicking Save on the Config File page under

System Setup.

State text may be added. Before adding text, set the maximum state value for this object. Then add text strings corresponding to each of the number of states allocated by entering the value, corresponding text, and clicking Add/Change. When changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for a Multistate Input object will typically be reading from some other BACnet or SNMP device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

Out of Service means polling of the mapped remote 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 mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

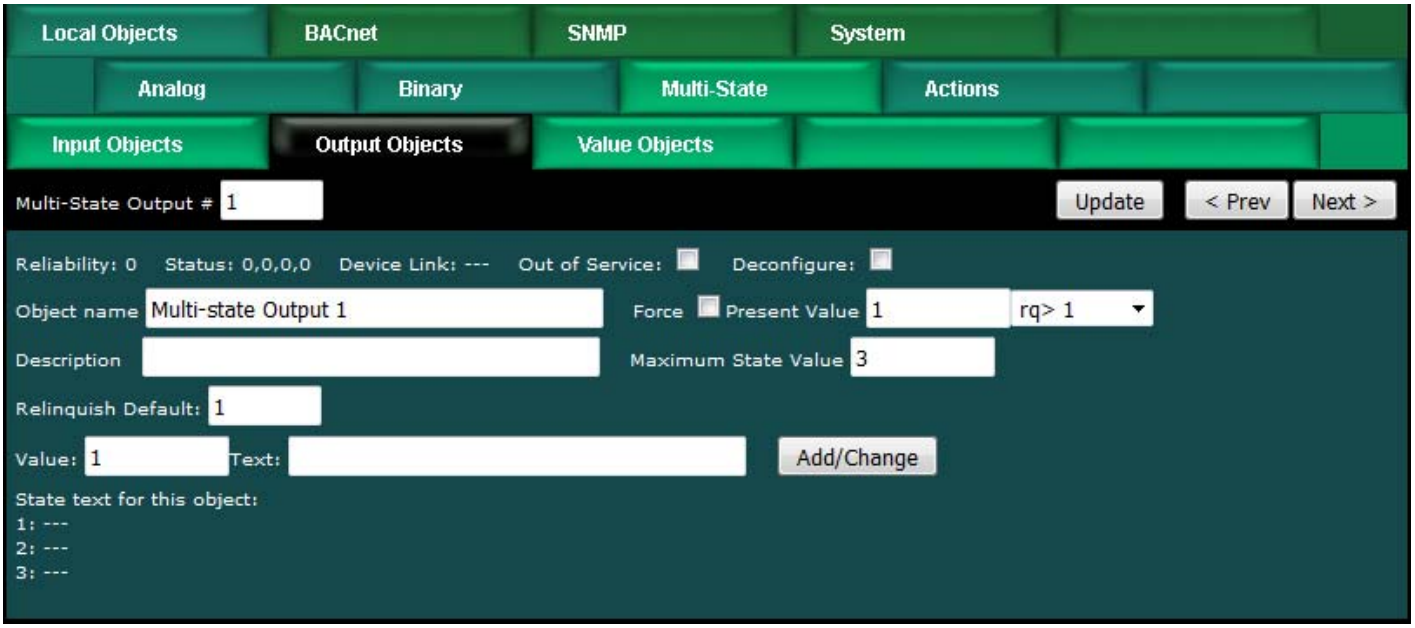
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.8 Multistate Output Objects

The destination of data for a Multistate Output object will typically be some other BACnet or SNMP device.



The object name, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. When changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

State text may be added. Before adding text, set the maximum state value for this object. Then add text strings corresponding to each of the number of states allocated by entering the value, corresponding text, and clicking Add/Change. When changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The Multistate Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

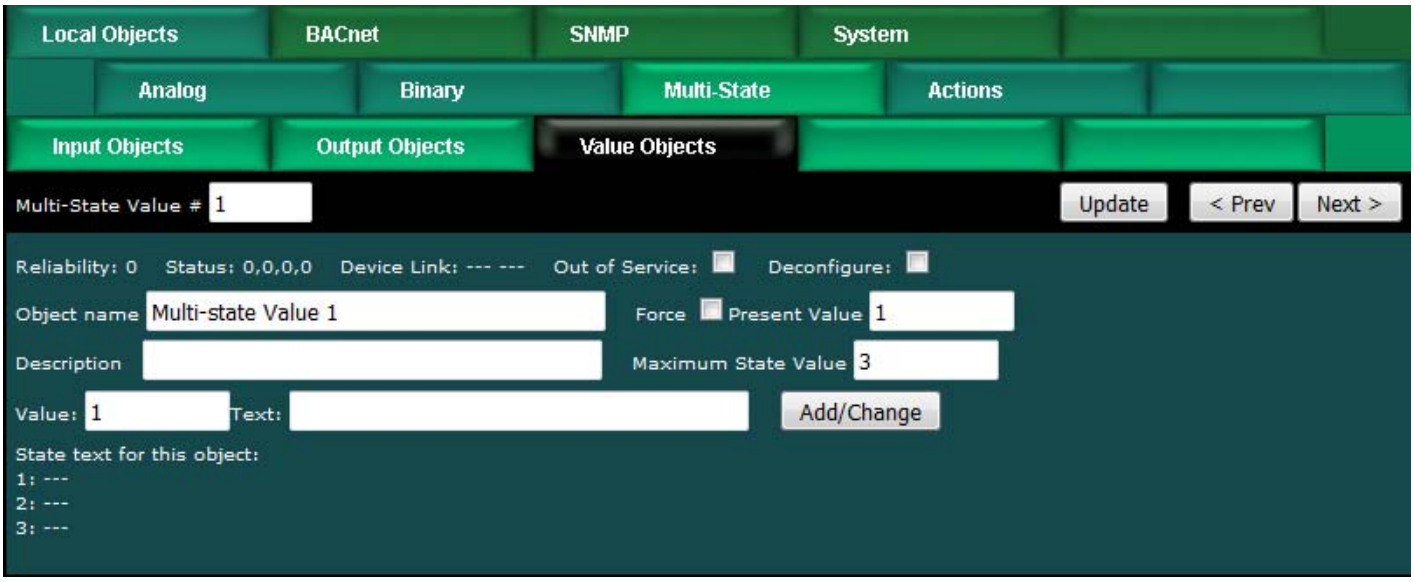
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

3.3.9 Multistate Value Objects

Multistate Value objects may be both a source and destination for some other BACnet or SNMP device.



The object name, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. When changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

State text may be added. Before adding text, set the maximum state value for this object. Then add text strings corresponding to each of the number of states allocated by entering the value, corresponding text, and clicking Add/Change. When changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

Multistate Value objects may be both a source and destination for some other BACnet or SNMP device. The source of data for the Value object will be reading from a remote device when associated with a Read Map. The destination of data for the Value object will be writing to a remote device when associated with a Write Map. If a remote device is mapped, the device links are displayed above. You may click on either link to view the respective mapping.

The Value object may be simultaneously associated with both Read and Write maps pointing to the same remote device object. When this Value object receives new data (from any source), this data will be written to the mapped remote device before any subsequent read from the same device. Thus the Value data is not discarded by the read operation before the new data can be written.

Out of Service means polling of the remote device will stop. While out of service, the present value may be written by an external BACnet client but it will not be written to any mapped remote device. Data may be forced via this web page at any time, but will be overwritten by the next read from a remote device unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

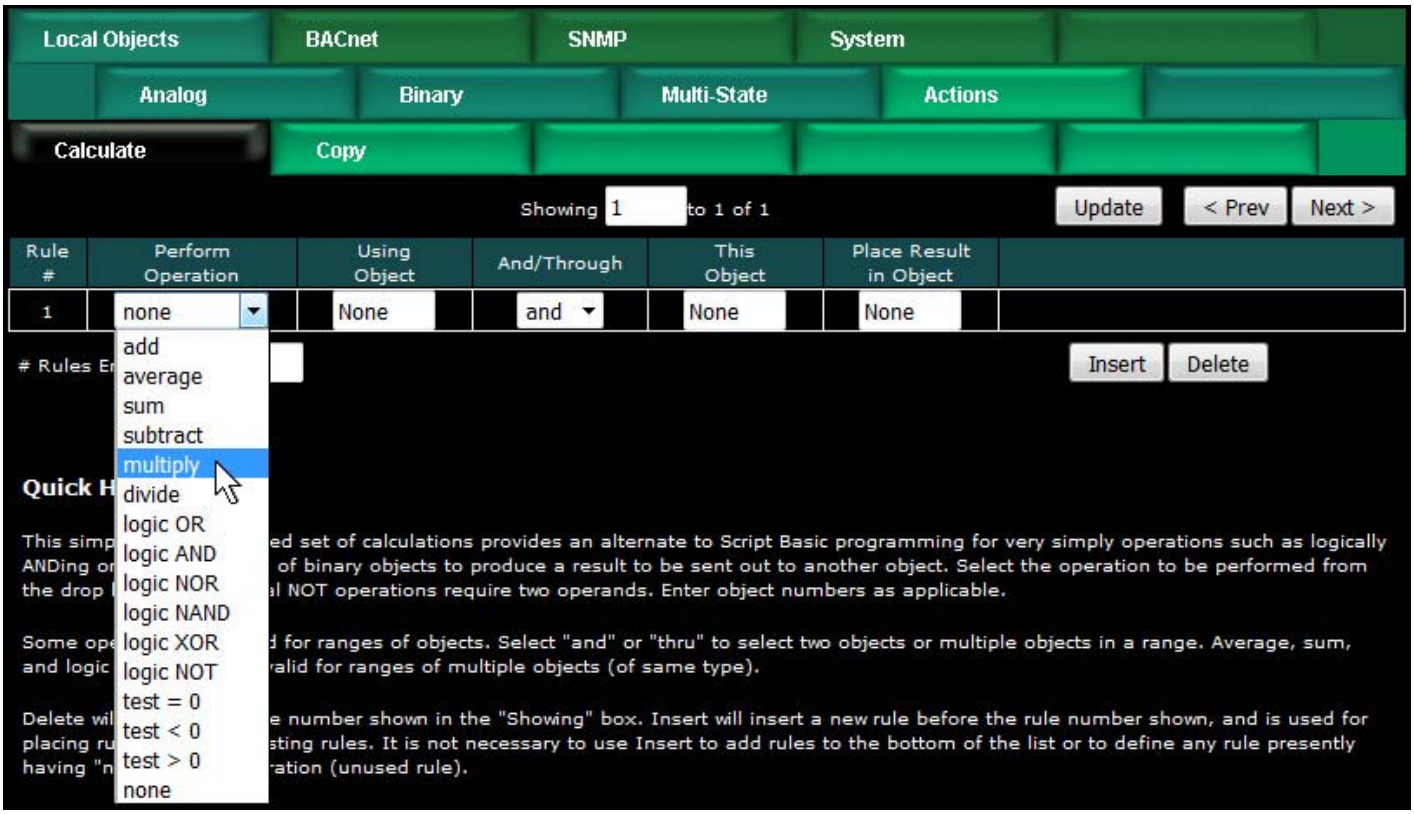
Check 'Deconfigure' and click Update to erase configuration for this object.

3.4 Local Object Calculate Rules

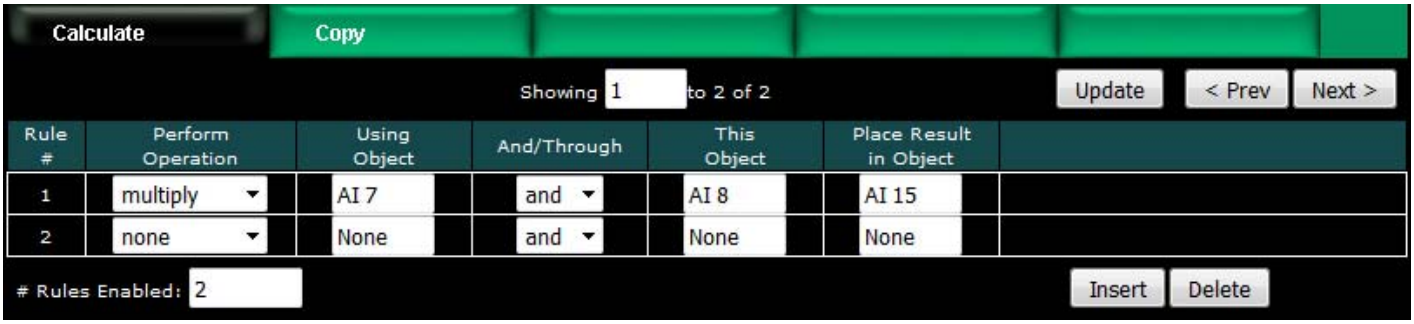
The Babel Buster Pro includes the ability to do simple calculations based on simple template rules. Select the operation, one or two operands as applicable, and a object to place the result in. Operations like "multiply" will use objects A "and" B. Operations like "sum" can add up the contents of a series of objects by selecting "thru" instead of "and".

These template rules can be useful for doing minor data manipulation or testing for purposes of enabling rules, or for generating SNMP traps. If you need to do something

more complex than what these simple rules will accomplish, you have the option of writing a program in Script Basic that can get as complicated as you like. Script Basic programs have the ability to both read and write local objects, which means your program can look at data values, and set objects that in turn cause things like SNMP traps to occur.



Here is an example of a template rule that multiplies the value of Analog Input 7 by value of Analog Input 8 and places the result in Analog Input 15. An example of application would be to multiply a voltage reading input by a current reading input to derive a power value presented as an input.



Constants may be introduced into the calculation by reserving a commandable object to hold that constant, and then configuring the relinquish default to be that value. Then reference that object in your calculate rule. If you need a lot of constants or have a lot of calculations, consider writing a short Script Basic program instead.

3.5 Local Object Copy Rules

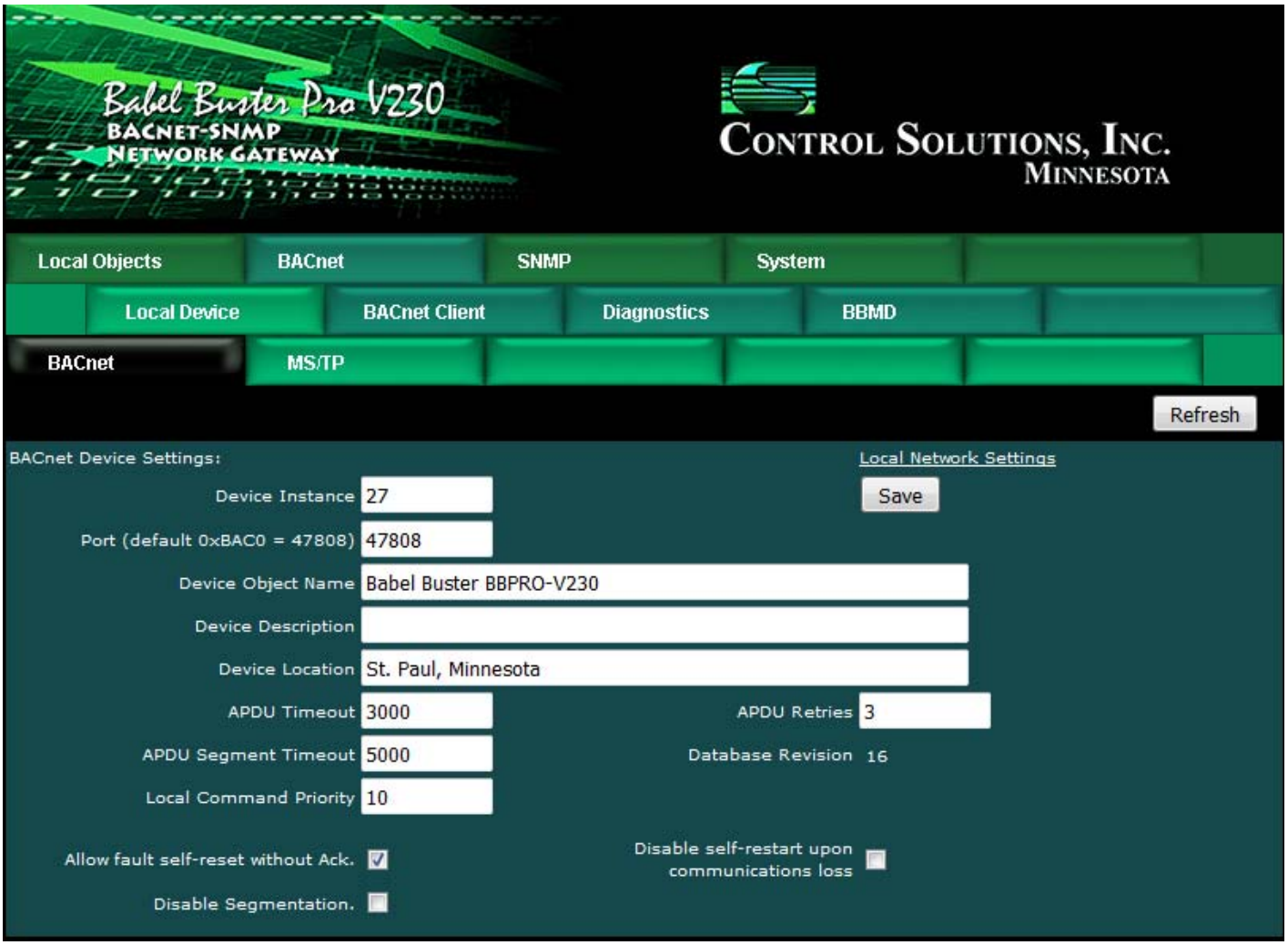
The copy rules provide a means of simply copying the content of one object to another.

The above rule would cause the value of AI 7 to be placed in AI 14. If a floating point (Analog) value is copied to a Binary object, the Binary object will be set Active if the value was nonzero, or cleared to Inactive if zero. Analog values copied to a Multistate object will be not only truncated, but bounded to the maximum number of states (not a recommended use of Copy).

4. Configuring Gateway as a BACnet Device

4.1 Device Object Parameters

The identity of the gateway as a BACnet device is entered on this page, along with other device object parameters.



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 device object name, description, and location are entered here. The device object name is expected to be unique to the entire BACnet network. Standard BACnet timeout and retry values are also entered on this page. These values are stored in a

special area of non-volatile memory rather than the XML configuration file.

Local command priority is used when Script Basic writes to a commandable object, or when the result of a Calculate or Copy rule is written to a commandable object. It is also used if the result of a client read map is saved to a local commandable object, although this would not be recommended. Output objects are commandable. Client read maps should store results in input or value objects, while client write maps take their data from value or output objects. In other words, output objects should not be used for input.

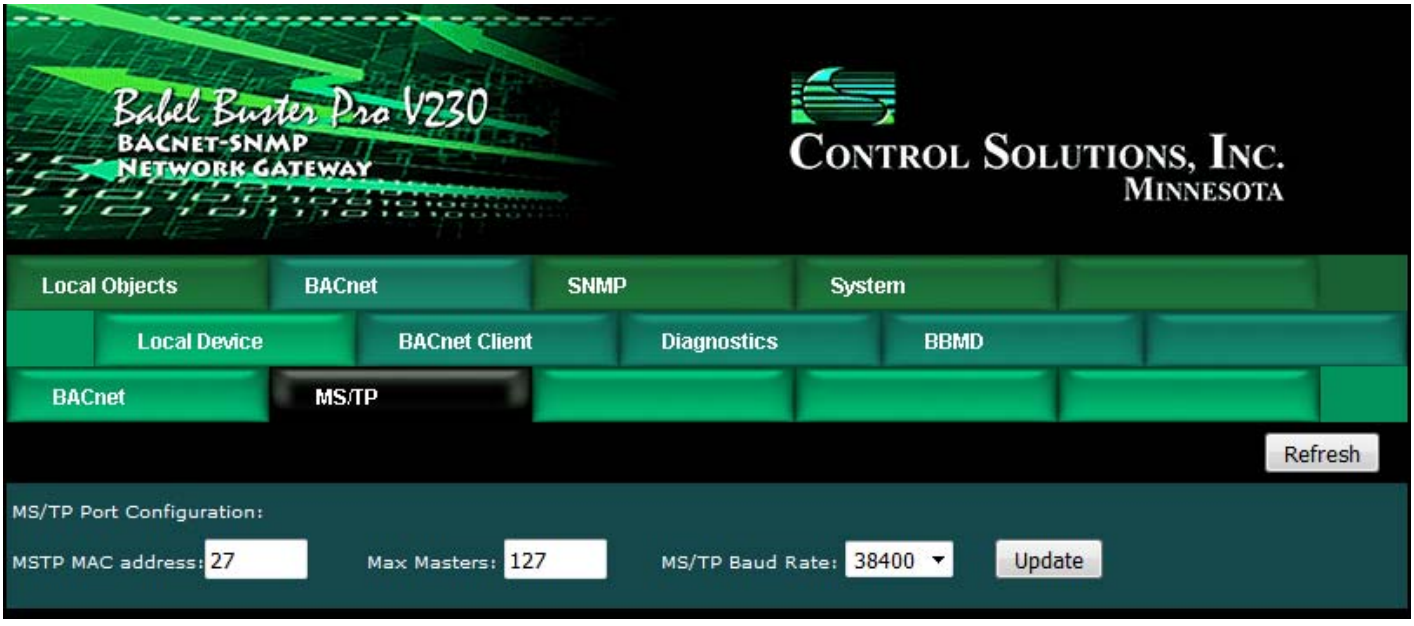
Check the "allow" check box to allow faults to self-reset. These faults are those conditions indicated by a non-zero reliability code in any of the data objects (see list on data objects pages). Normally an external client needs to read the realiability code to acknowledge the fault before it will automatically 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.

Check the "disable self-restart" box to disable self restart upon communication loss. If this box is not checked, this gateway will restart itself in an attempt to auto-recover if communications with devices has started and then stopped.

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.

4.2 MS/TP Settings

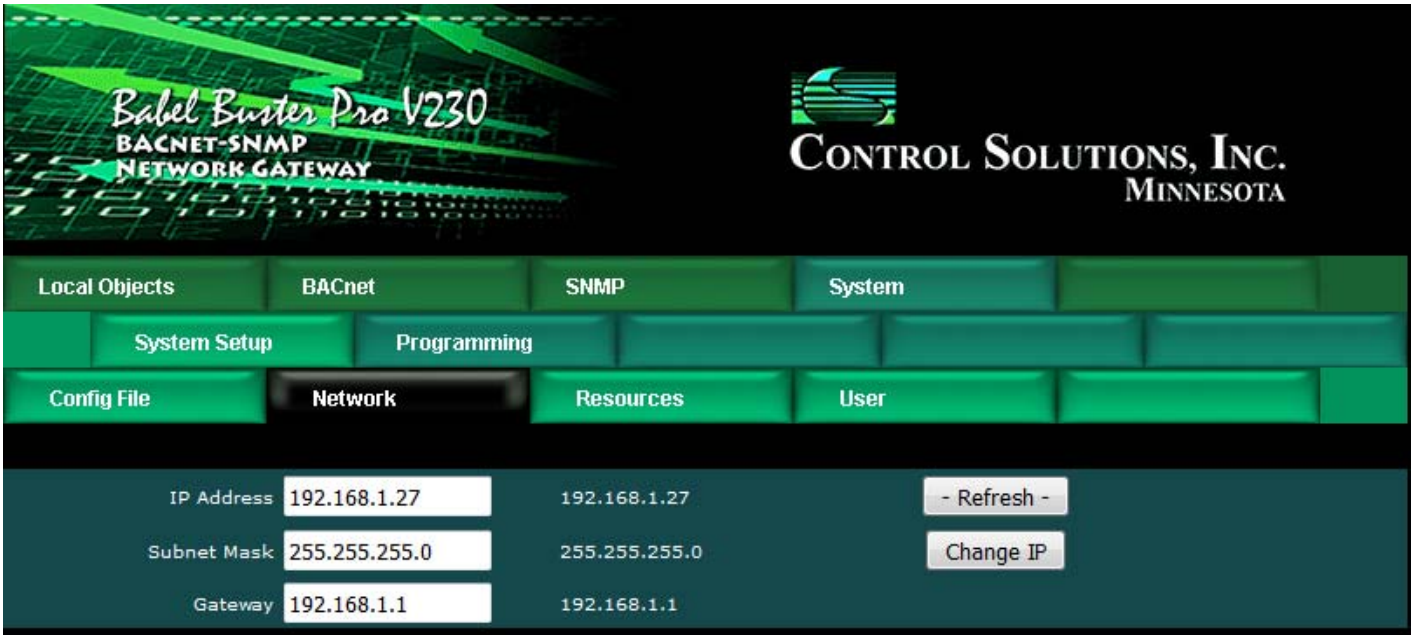
The MS/TP port settings are entered on the MS/TP page. This page is not present in the -SP version of the BBPRO-V230. Baud rate for the serial port in the -SP is set by the Basic program that opens the comm port.



Enter MS/TP port parameters as applicable. The Max Masters count must be equal to or higher than the highest MAC address on the MSTP link. MAC address must be unique on the MS/TP link to avoid communication problems. Select the baud rate that matches other devices on the link. Click Update to save changes. These changes are stored in configuration memory, therefore you do not need to re-save the XML configuration file to retain these settings. Wait at least 30 seconds before power cycling or restarting the device after clicking Update to allow the configuration memory update to complete.

4.3 Network Settings

The two most important things that must be unique on the BACnet IP network are device instance, and IP address. The IP address is set on the Network page.



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.

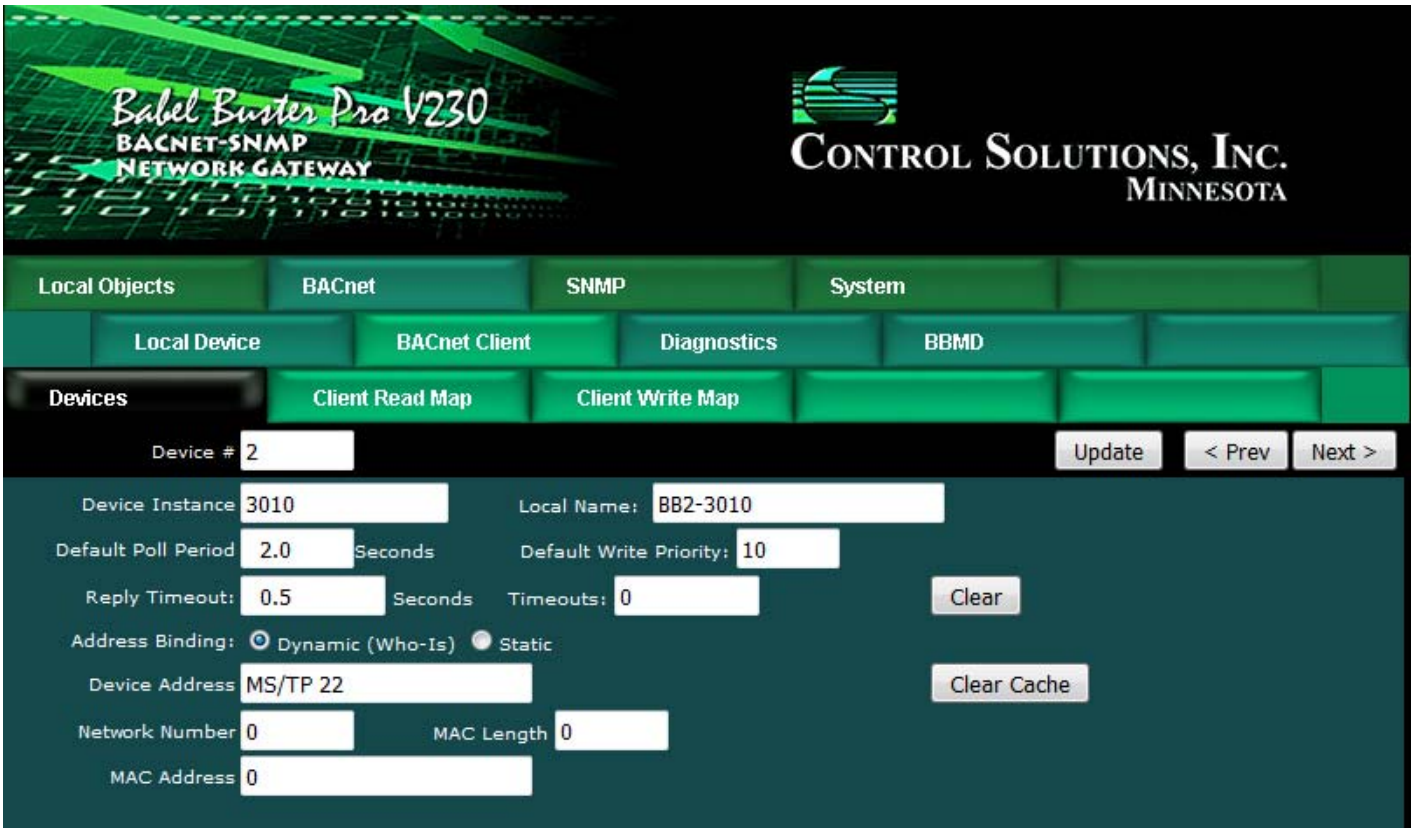
The above screen shot is only a portion of the Network setup page, and is the only part of the Network page that is required for BACnet IP. The remainder of the Network page is discussed in Section 14.2.

5. Configuring Gateway as a BACnet Client

The BACnet client is used to query other BACnet devices, obtain their Present Value data, and store a copy of that data in the BBPRO-V230's own local objects. From there, the data may be accessed by SNMP devices, or other BACnet devices when application specific reasons make this approach more preferred than direct routing.

5.1 BACnet Device List

Setting up the BACnet client consists of identifying one or more BACnet devices, then listing the objects that should be queried (whether read or written). The client configuration pages are illustrated below.



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

Remote BACnet IP 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, and a poll

rate. Then click "update".

Select dynamic or static address binding. Dynamic binding is used most often, and simply means the gatewill will send out a "Who-Is" request asking for the device instance to respond, at which time the gateway learns its IP address (or MS/TP address) automatically.

If static binding must be used and the device is BACnet IP, enter the fixed IP address you know the device instance to be found at. If no port is given, it will default to 0xBAC0 (47808). Enter IP as a.b.c.d or IP with port as a.b.c.d:p, for example 192.168.1.99:47808. If the device is MS/TP, enter "MS/TP n" where 'n' is the MS/TP MAC address. Be sure to type MS/TP in upper case, followed by a blank, followed by a number.

Include network number, mac length, and mac address ONLY if static binding to a device on the other side of a BACnet router.

When dynamic address binding is used (default), the gateway broadcasts a "Who-Is" looking for this device instance 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. When dynamic binding is used, the IP address is read-only and any changes entered will be ignored.

5.2 BACnet Client Read Maps

Getting the gateway to read objects from another BACnet device requires setting up a "Read Map" as shown here.

Map #	Remote Type	Remote Object #	Remote Device	Scale	Local Object #	Name
1	Analog Input	1	BB2-7010	0.000000	AV 1	Analog Value 1
2	Analog Input	2	BB2-7010	0.000000	AV 2	Analog Value 2
3	Analog Input	3	BB2-7010	0.000000	AV 3	Analog Value 3
4	None	0	None	0.000000	None	---

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 object instance and type, and location (device). The names in the device list are defined in the Devices page. The property read will default to Present Value. If you wish to read a different property, click on the Map # in the first column for the expanded view of the map and enter the property number.

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.

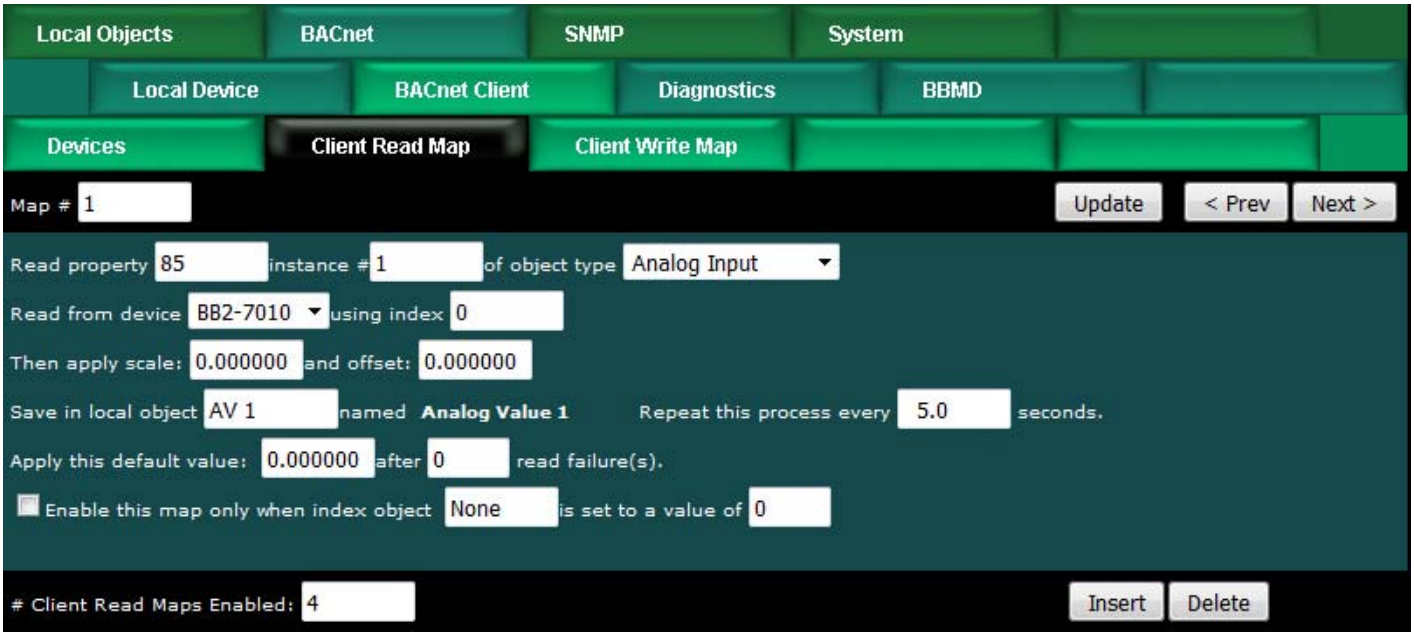
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

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 numbers will appear as "register numbers" in XML configuration files. 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 are set on the Resources page (under System).

Click on a Map # in the first column of maps to get the expanded view of that map as follows:



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 "Map #" box, then click Update.

For each remote object to be read, enter the property number, 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.

The most commonly read property will be Present Value, which is property number 85. For other property numbers, refer to Appendix F, BACnet Codes.

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.

You have the option of enabling this map only when a selected object contains a given value. Any local object may be used as the index object. As the name implies, you could have the same local object contain different values based on different maps as indexed by the index object.

Delete will remove the map 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

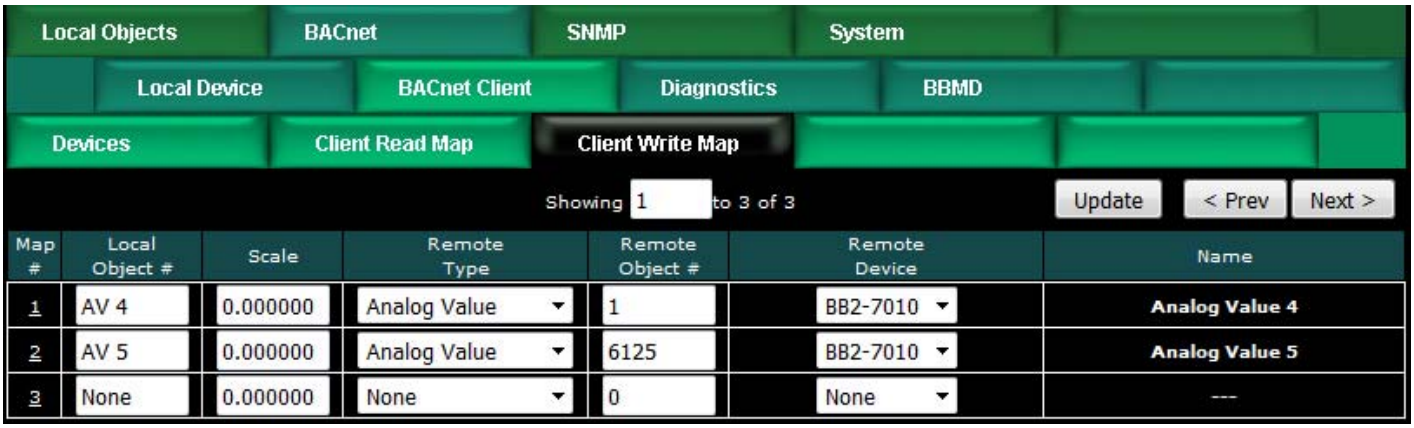
define any map presently having zero for a source object or "none" for remote type.

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.

5.3 BACnet Client Write Maps

Getting the gateway to write objects to another BACnet device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.



Map #	Local Object #	Scale	Remote Type	Remote Object #	Remote Device	Name
1	AV 4	0.000000	Analog Value	1	BB2-7010	Analog Value 4
2	AV 5	0.000000	Analog Value	6125	BB2-7010	Analog Value 5
3	None	0.000000	None	0	None	---

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 object instance and type, and location (device). The names in the device list are defined in the Devices page. The property written will default to Present Value. If you wish to write a different property, click on the Map # in the first column for the expanded view of the map and enter the property number.

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.

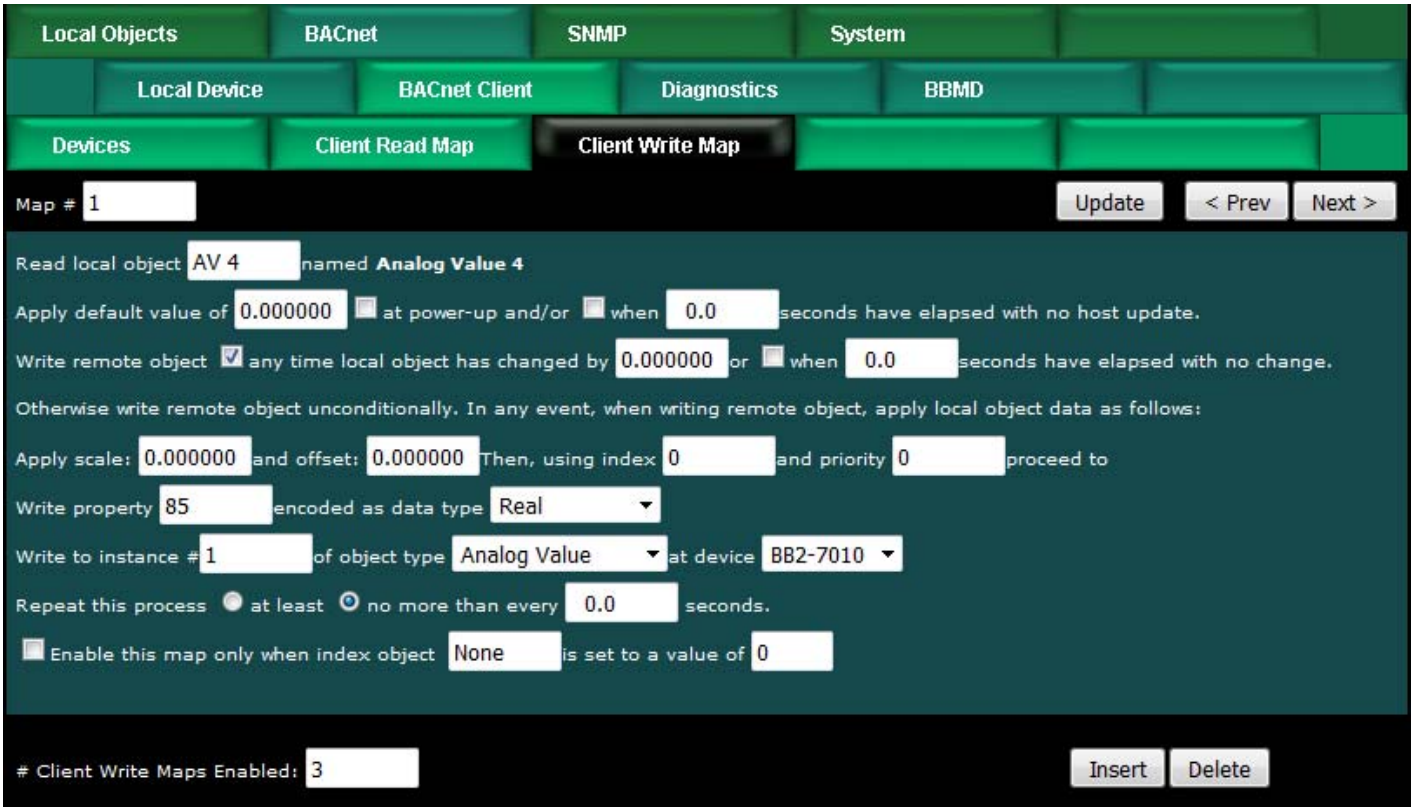
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type.

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 numbers will appear as "register numbers" in XML configuration files. 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 are set on the Resources page (under System).

Click on a Map # in the first column of maps to get the expanded view of that map as follows:



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 "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 object. 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 property number, 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 most commonly written property will be Present Value, which is property number 85. For other property numbers, refer to Appendix F, BACnet Codes.

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.

You have the option of enabling this map only when a selected object contains a given value. Any local object may be used as the index object. As the name implies, you can write different values to the remote object based on different maps as indexed by the index object.

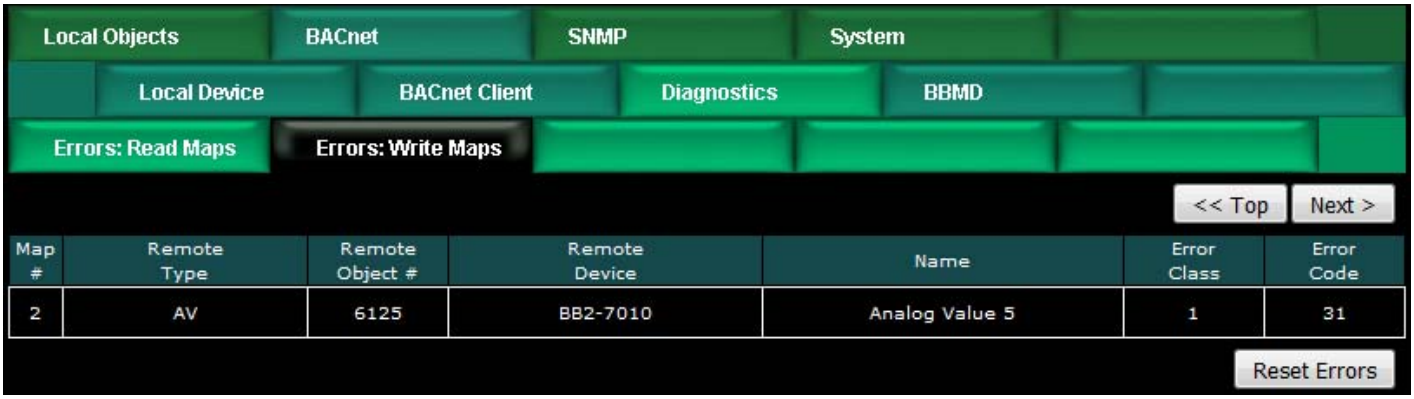
Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object 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.

5.4 BACnet Client Diagnostics

If errors are detected in the course of reading or writing other BACnet objects via the client's maps, they will be indicated on the errors pages.



Errors for BACnet client read/write maps are shown on these pages. Only those maps with errors to report are listed. Refer to the code and class lists below for interpretation. In the illustration above, error class 1 says the error refers to "object" and the code says "unknown object". In other words, AV 6125 does not exist in the device shown.

Proprietary class 82, code 0, is generated locally indicating a timeout, no response received from remote server. All other codes listed below are returned by the remote server.

0 = ERROR_CLASS_DEVICE

```
1 = ERROR_CLASS_OBJECT
2 = ERROR_CLASS_PROPERTY
3 = ERROR_CLASS_RESOURCES
4 = ERROR_CLASS_SECURITY
5 = ERROR_CLASS_SERVICES

/* valid for all classes */
0 = ERROR_CODE_OTHER

/* Error Class - Device */
2 = ERROR_CODE_CONFIGURATION_IN_PROGRESS
3 = ERROR_CODE_DEVICE_BUSY
25 = ERROR_CODE_OPERATIONAL_PROBLEM

/* Error Class - Object */
4 = ERROR_CODE_DYNAMIC_CREATION_NOT_SUPPORTED
17 = ERROR_CODE_NO_OBJECTS_OF_SPECIFIED_TYPE
23 = ERROR_CODE_OBJECT_DELETION_NOT_PERMITTED
24 = ERROR_CODE_OBJECT_IDENTIFIER_ALREADY_EXISTS
27 = ERROR_CODE_READ_ACCESS_DENIED
31 = ERROR_CODE_UNKNOWN_OBJECT
36 = ERROR_CODE_UNSUPPORTED_OBJECT_TYPE

/* Error Class - Property */
8 = ERROR_CODE_INCONSISTENT_SELECTION_CRITERION
9 = ERROR_CODE_INVALID_DATA_TYPE
32 = ERROR_CODE_UNKNOWN_PROPERTY
37 = ERROR_CODE_VALUE_OUT_OF_RANGE
40 = ERROR_CODE_WRITE_ACCESS_DENIED
41 = ERROR_CODE_CHARACTER_SET_NOT_SUPPORTED
42 = ERROR_CODE_INVALID_ARRAY_INDEX
44 = ERROR_CODE_NOT_COV_PROPERTY
45 = ERROR_CODE_OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
47 = ERROR_CODE_DATATYPE_NOT_SUPPORTED
50 = ERROR_CODE_PROPERTY_IS_NOT_AN_ARRAY

/* Error Class - Resources */
18 = ERROR_CODE_NO_SPACE_FOR_OBJECT
19 = ERROR_CODE_NO_SPACE_TO_ADD_LIST_ELEMENT
20 = ERROR_CODE_NO_SPACE_TO_WRITE_PROPERTY

/* Error Class - Security */
1 = ERROR_CODE_AUTHENTICATION_FAILED
6 = ERROR_CODE_INCOMPATIBLE_SECURITY_LEVELS
12 = ERROR_CODE_INVALID_OPERATOR_NAME
15 = ERROR_CODE_KEY_GENERATION_ERROR
```

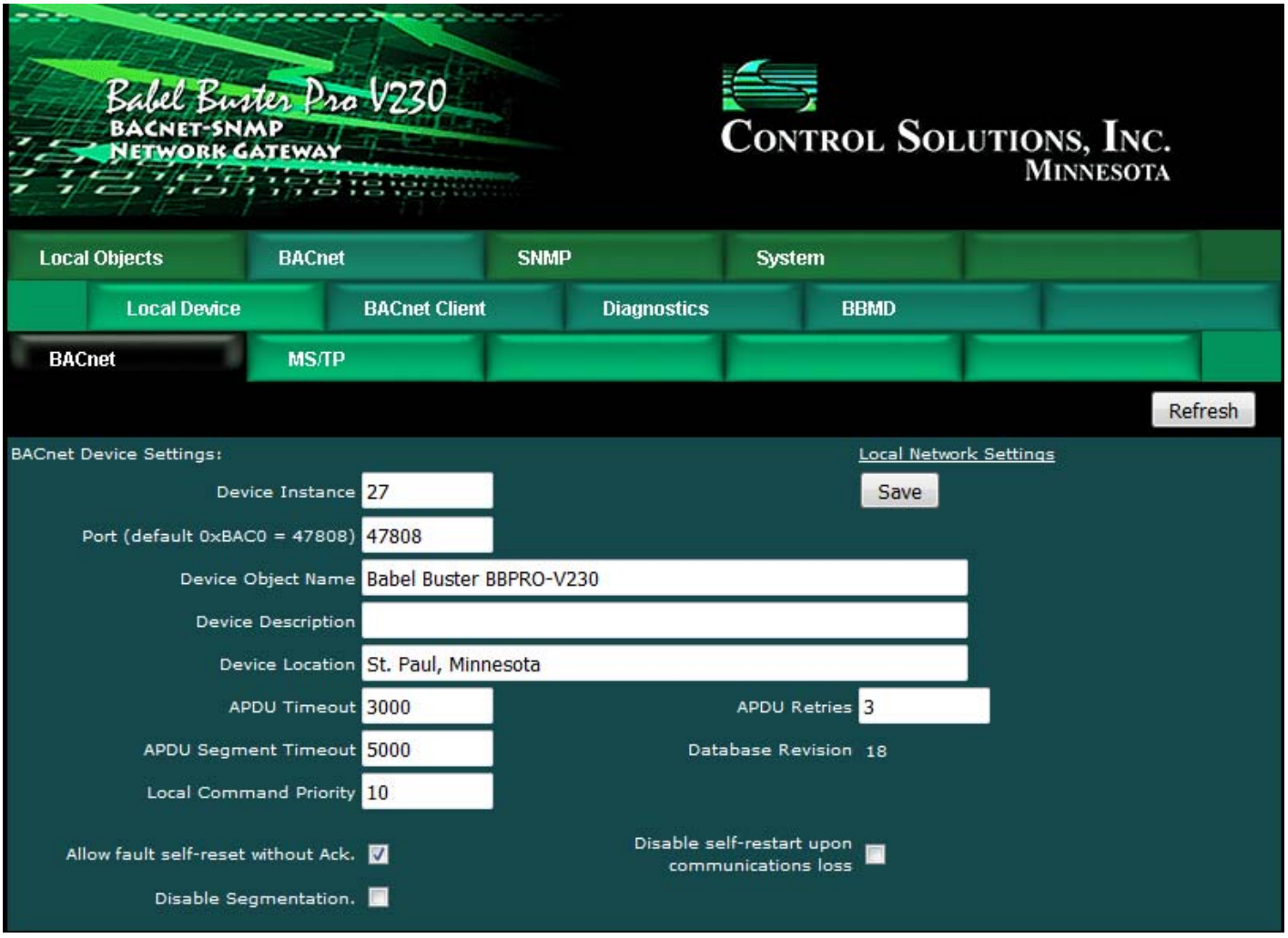
```
26 = ERROR_CODE_PASSWORD_FAILURE
28 = ERROR_CODE_SECURITY_NOT_SUPPORTED
30 = ERROR_CODE_TIMEOUT

/* Error Class - Services */
5 = ERROR_CODE_FILE_ACCESS_DENIED
7 = ERROR_CODE_INCONSISTENT_PARAMETERS
10 = ERROR_CODE_INVALID_FILE_ACCESS_METHOD
11 = ERROR_CODE_ERROR_CODE_INVALID_FILE_START_POSITION
13 = ERROR_CODE_INVALID_PARAMETER_DATA_TYPE
14 = ERROR_CODE_INVALID_TIME_STAMP
16 = ERROR_CODE_MISSING_REQUIRED_PARAMETER
22 = ERROR_CODE_PROPERTY_IS_NOT_A_LIST
29 = ERROR_CODE_SERVICE_REQUEST_DENIED
43 = ERROR_CODE_COV_SUBSCRIPTION_FAILED
46 = ERROR_CODE_INVALID_CONFIGURATION_DATA
48 = ERROR_CODE_DUPLICATE_NAME
49 = ERROR_CODE_DUPLICATE_OBJECT_ID
```

6. Configuring Gateway as a BACnet Server

6.1 Server Configuration

The BBPRO-V230 contains a set of BACnet objects whose only purpose is to store copies of data obtained from other devices. This copy of data may then be queried by different devices.

The only configuration needed to use the BBPRO-V230 as a BACnet server is to set the Device instance on the BACnet page. The device should also be given an object name that will be unique on the entire network. Configuring the gateway as a BACnet Device is described in more detail in Section 4.



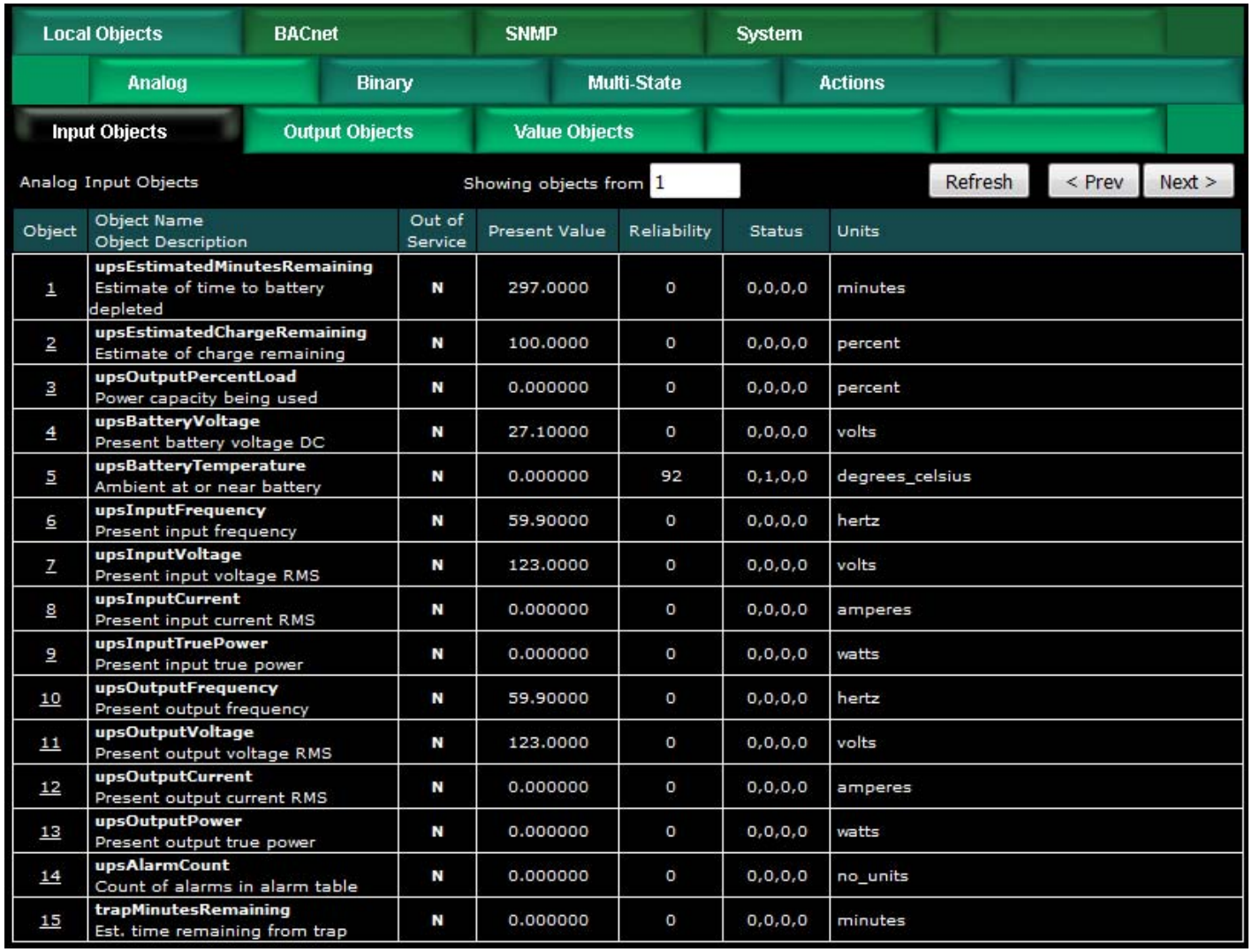
6.2 Accessing Local Objects

The collection of objects includes Analog, Binary, and Multi-State types of objects, and includes Input, (commandable) Output, and (writeable) Value types of each of those objects. The BBPRO-V230 also contains a Device object which is configured in the above screen.

Data may be placed in the local objects by other devices writing to the BBPRO-V230, or by the BBPRO-V230 querying other devices. When the BBPRO-V230 is configured to query other devices, these operations are defined by “read maps” and “write maps” associated with the respective client function (BACnet client or SNMP client).

The following pages illustrate the Analog Input object pages and the Binary Output object pages. The remaining object pages found in the BBPRO-V230 are virtually identical, and are not replicated here. (See also Configuring Local Objects, Section 3.)

Each object page initially comes up as a table of object data. Click on the object number in the left-hand column to expand the view of that object and access the windows that let you locall force values, assign units or names, etc.



Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**upsEstimatedMinutesRemaining** Estimate of time to battery depleted	N	297.0000	0	0,0,0,0	minutes
2	**upsEstimatedChargeRemaining** Estimate of charge remaining	N	100.0000	0	0,0,0,0	percent
3	**upsOutputPercentLoad** Power capacity being used	N	0.000000	0	0,0,0,0	percent
4	**upsBatteryVoltage** Present battery voltage DC	N	27.10000	0	0,0,0,0	volts
5	**upsBatteryTemperature** Ambient at or near battery	N	0.000000	92	0,1,0,0	degrees_celsius
6	**upsInputFrequency** Present input frequency	N	59.90000	0	0,0,0,0	hertz
7	**upsInputVoltage** Present input voltage RMS	N	123.0000	0	0,0,0,0	volts
8	**upsInputCurrent** Present input current RMS	N	0.000000	0	0,0,0,0	amperes
9	**upsInputTruePower** Present input true power	N	0.000000	0	0,0,0,0	watts
10	**upsOutputFrequency** Present output frequency	N	59.90000	0	0,0,0,0	hertz
11	**upsOutputVoltage** Present output voltage RMS	N	123.0000	0	0,0,0,0	volts
12	**upsOutputCurrent** Present output current RMS	N	0.000000	0	0,0,0,0	amperes
13	**upsOutputPower** Present output true power	N	0.000000	0	0,0,0,0	watts
14	**upsAlarmCount** Count of alarms in alarm table	N	0.000000	0	0,0,0,0	no_units
15	**trapMinutesRemaining** Est. time remaining from trap	N	0.000000	0	0,0,0,0	minutes

The object name, units, value, and status are shown for a list of objects starting with the number entered at the top of the page. Click Prev/Next to scroll through the list. Click on the object number in the first column to change name, units, COV, and out-of-service state.

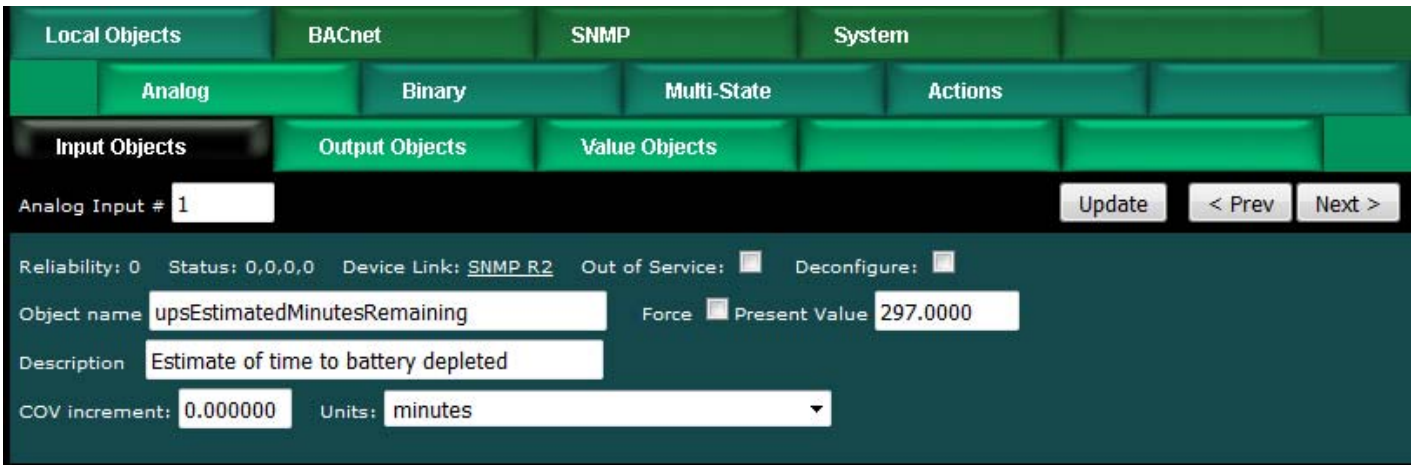
The source of data for an Analog Input object will typically be reading from some other BACnet or SNMP device. Click on the object number in the first column for more detail including the link to any client map providing data to this object.

Out of Service means polling for data 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 mapped client update unless the object is out of service.

Reliability codes may be any of the following:
BACnet IP client, device timeout (82)
BACnet IP client, error returned by server (83)
SNMP client, no response from agent (91)
SNMP client, agent returned 'no such name' error (92)
SNMP client, unable to parse data returned (93)
SNMP client, reply did not match request (94)
SNMP client, invalid table walk configuration (95)
SNMP client, unable to parse PDU returned (96)
SNMP client, table walk came up short on data (97)
SNMP client, agent returned error other than 'no such name' (98)

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Click on an Object number in the first column of maps to get the expanded view of that object as follows:



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. BACnet units may be selected. Initial COV increment may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The object may be set Out of Service by checking that box and clicking Update. The present value may be changed by entering a value, checking Force, and clicking Update.

The source of data for an Analog Input object will typically be reading from some other BACnet or SNMP device via the map indicated by the Device Link. The mapped device will be polled at the rate specified by the Read Map.

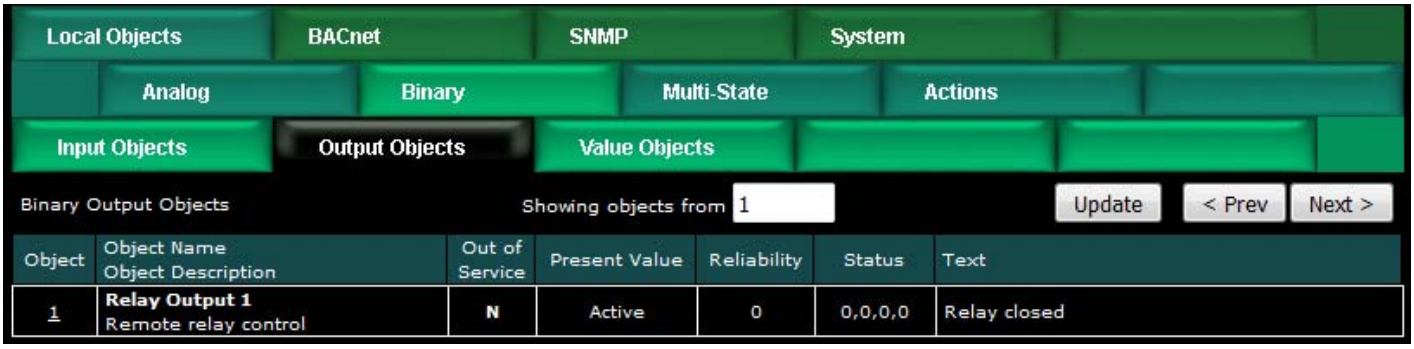
Out of Service means polling of the mapped remote device will stop. While out of service, the present value may be written by the BACnet client. Data may be forced via this web page at any time, but will be overwritten by the next mapped client update unless the object is out of service.

Reliability codes indicate errors as itemized on the tabular object list.

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.



The object name, value, and status are shown for a list of objects starting with the number entered at the top of the page. Click

on the object number in the first column to change name or out-of-service state.

The destination of data for a Binary Output object will typically be some other BACnet or SNMP device. Click on the object number in the first column for more detail including the link to any client map receiving data from this object.

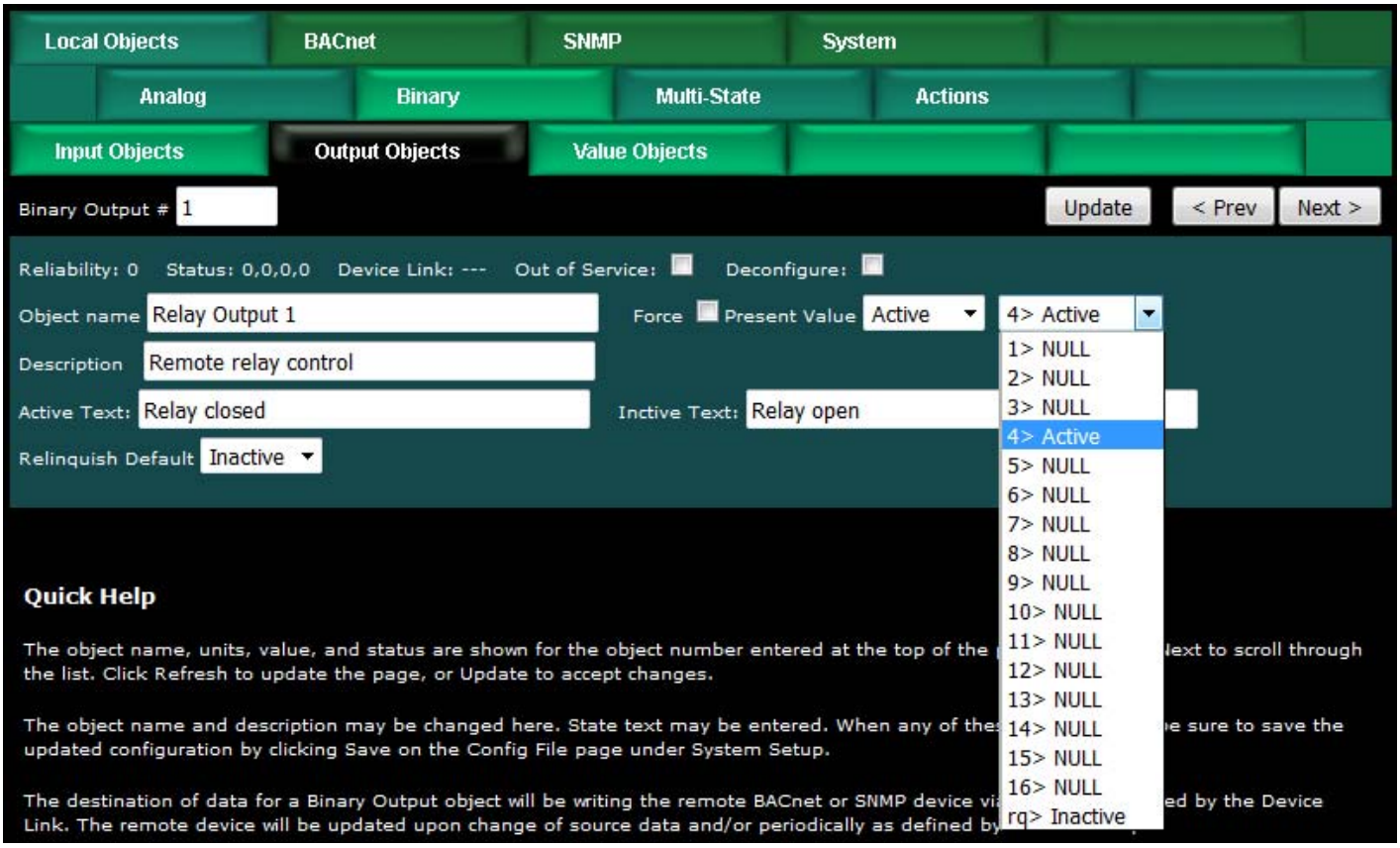
The Binary Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes may be any of the following:
BACnet IP client, device timeout (82)
BACnet IP client, error returned by server (83)
SNMP client, no response from agent (91)
SNMP client, agent returned 'no such name' error (92)
SNMP client, unable to parse data returned (93)
SNMP client, reply did not match request (94)
SNMP client, invalid table walk configuration (95)
SNMP client, unable to parse PDU returned (96)
SNMP client, table walk came up short on data (97)
SNMP client, agent returned error other than 'no such name' (98)

Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Click on an Object number in the first column of maps to get the expanded view of that object as follows:



The object name, units, value, and status are shown for the object number entered at the top of the page. Click Prev/Next to scroll through the list. Click Refresh to update the page, or Update to accept changes.

The object name and description may be changed here. State text may be entered. When any of these are changed, be sure to save the updated configuration by clicking Save on the Config File page under System Setup.

The destination of data for a Binary Output object will be writing the remote BACnet or SNMP device via the map indicated by the Device Link. The remote device will be updated upon change of source data and/or periodically as defined by the Write Map.

The Binary Output object is commandable, meaning the BACnet client must write both a value and a priority level for that value. The highest level value will be the one written to the remote device (if one is mapped). If all values are relinquished, the relinquish default value will be written to the remote device.

To set an output object manually from this page, check the Force box, enter a value in the Present Value window, and select a priority level to assign to your forced value. Then click Update. To return a given priority level to NULL, simply type the word NULL in the Present Value window, check Force, and click Update.

Out of service means the mapped remote device will not be written to. Values written by the BACnet client will be retained, but only applied when this object is placed back in service. At that time, the highest priority value will be written to the remote device.

Reliability codes indicate errors as itemized on the tabular object list.

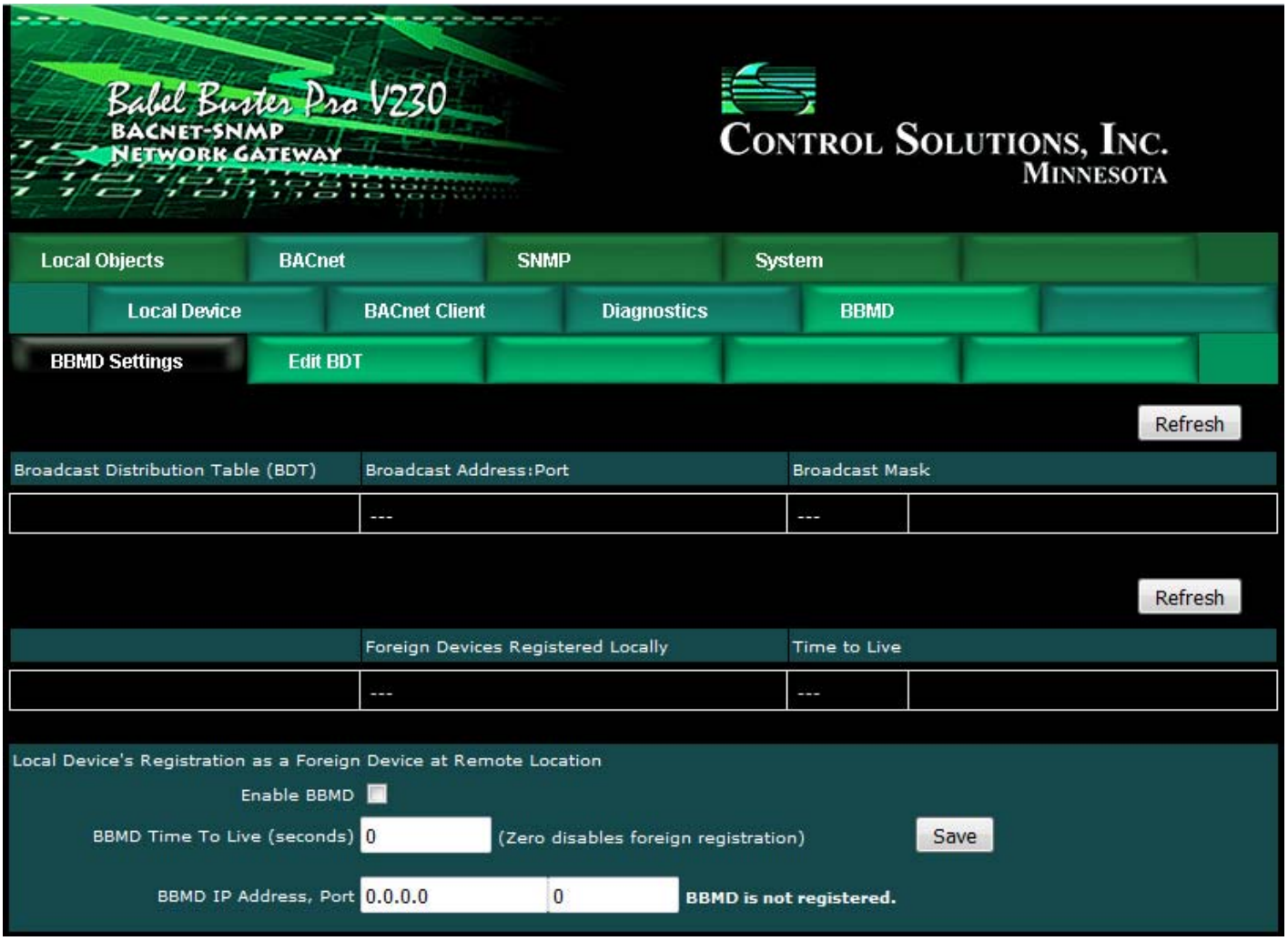
Status flags A,B,C,D indicate the following, 0 meaning not true, 1 meaning true:
A = in alarm
B = fault
C = overridden
D = out of service

Device link will indicate BIP, SNMP, TRAP, or WALK, 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 to this BACnet object. The designation R means read from a remote device, and W means write to a remote device.

Check 'Deconfigure' and click Update to erase configuration for this object.

7. Configuring BBMD

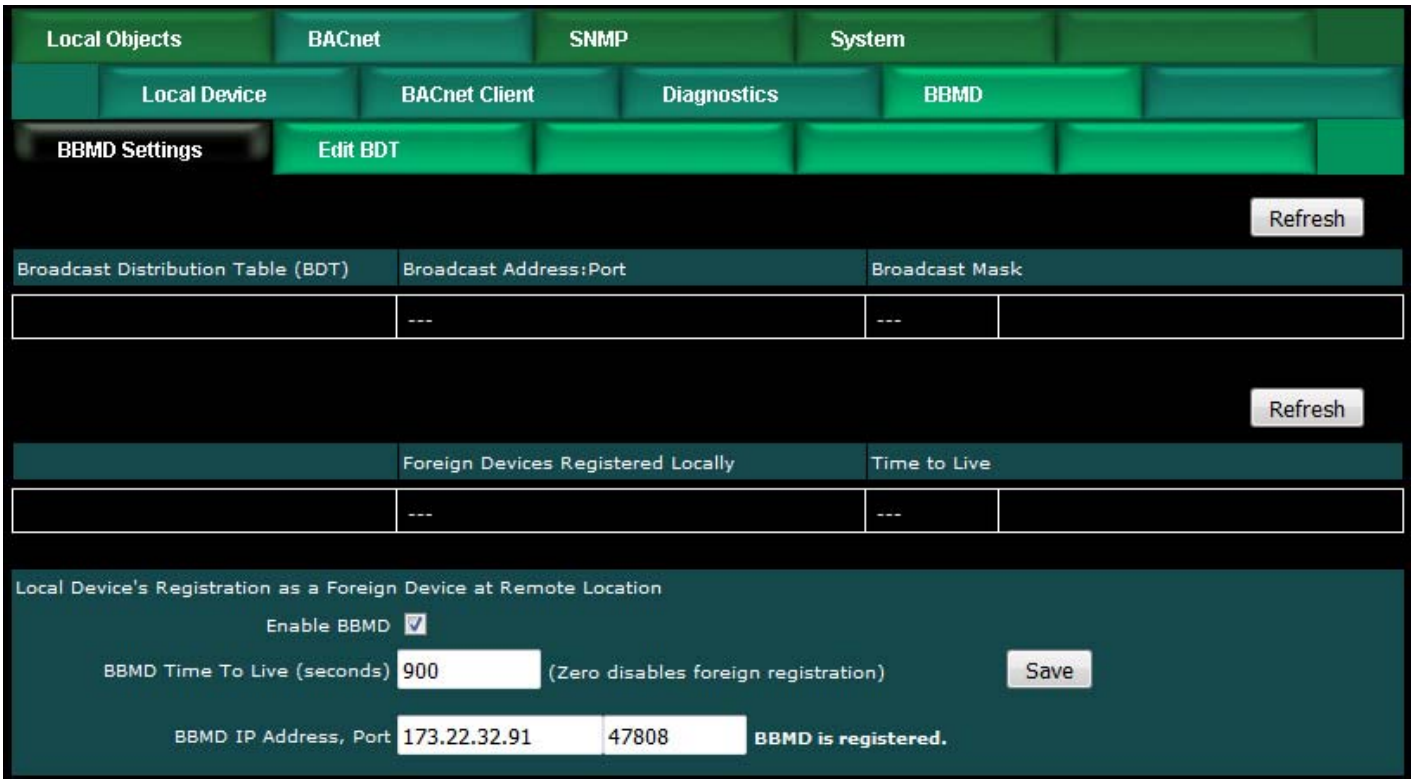
BBMD stands for BACnet Broadcast Management Device. Messages such as "Who-Is" and "I-Am" are broadcast. Most routers, however, to not pass broadcast messages along. The BBMD solves this problem by explicitly directing broadcast messages to a specific IP address.



The BBMD Settings page appears as shown above when no part of BBMD support is enabled, as is the case when shipped. Do not enable BBMD if you are not aware of needing it and/or do not understand how BBMD works. The three elements of BBMD support are discussed in the following sections, and their use is often mutually exclusive, meaning you will often need only one of the three elements.

7.1 Registering as a Foreign Device

If you have a remote BBPRO-V230 that needs to connect via router, including NAT router, to a local network, use Foreign Device Registration. There will typically be a master device, such as operator station or other front end, that includes BBMD. The IP address of this device is the one that should be given as the BBMD address for foreign device registration.



To enable BBMD processing, check the "Enable BBMD" box. This applies to foreign device registration. The broadcast distribution table functions regardless of whether foreign device registration is enabled.

If the BBPRO-V230 should register as a foreign device with another BBMD, then the port, time-to-live, and IP address of the remote BBMD must be given. The local BBMD will attempt to register with the remote BBMD whose address is given.

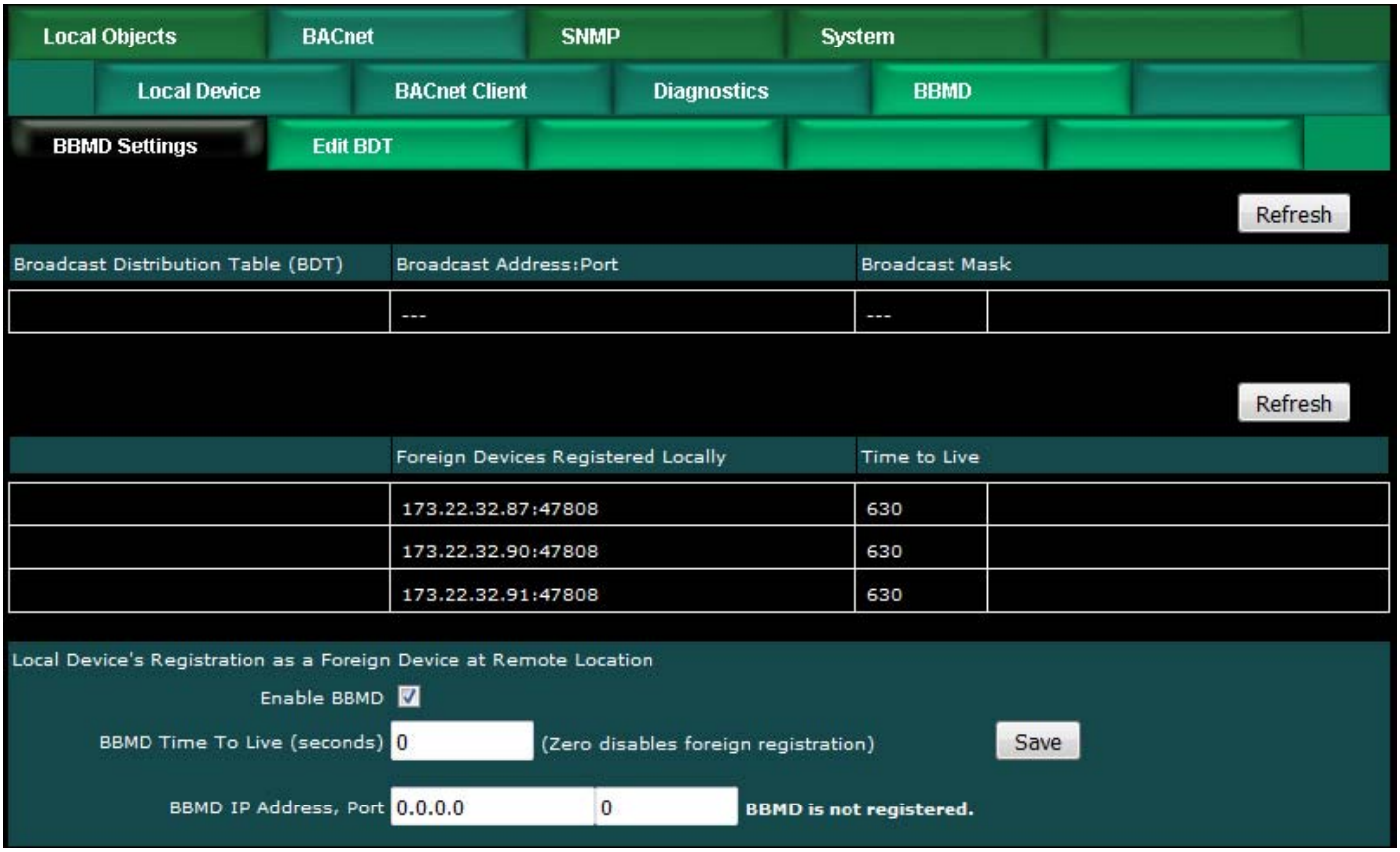
Disable this device's attempts to register elsewhere, but allow other devices to register here, by setting time to live to zero with BBMD enabled.

7.2 Allowing Other Devices to Register Locally

The BBPRO-V230 can be the BBMD that other devices register with. The screen shot below shows that three other devices have registered with this BBMD, and broadcast messages will now be sent explicitly to these locations. In this case, there are NAT routers between this local device and the three remote devices. While they are all on physically separate local networks, they will appear as a single BACnet network even if the local networks are miles apart. The local BACnet client will be able to

communicate with these remote BACnet devices as a result of the foreign registration.

Note that foreign registration only provides communication with a single remote device. If communicating with an entire remote network of BACnet devices is the intent, then a full BACnet router is required, and BBMD would be handled by the BACnet router (disable everything on this BBMD Settings page if connected via a BACnet router).

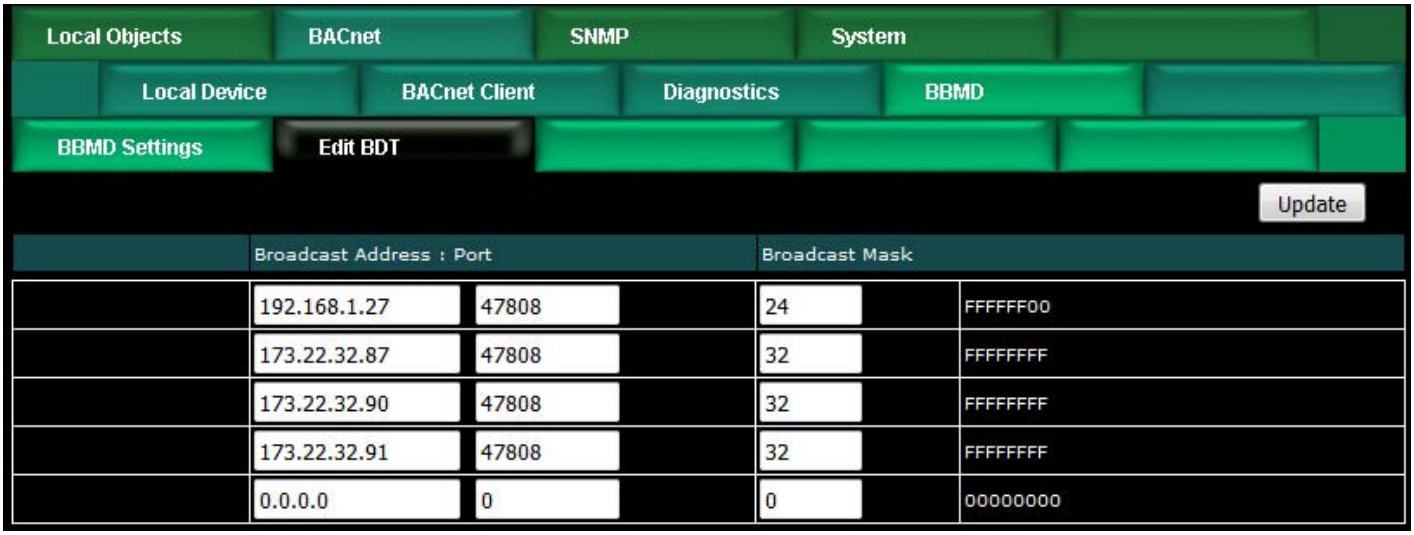


To allow foreign devices to register with this device, but not have this device register elsewhere, check Enable BBMD, but enter zero for BBMD Time To Live. This enables BBMD but disables this device's attempt to register somewhere else.

7.3 Broadcast Distribution Table

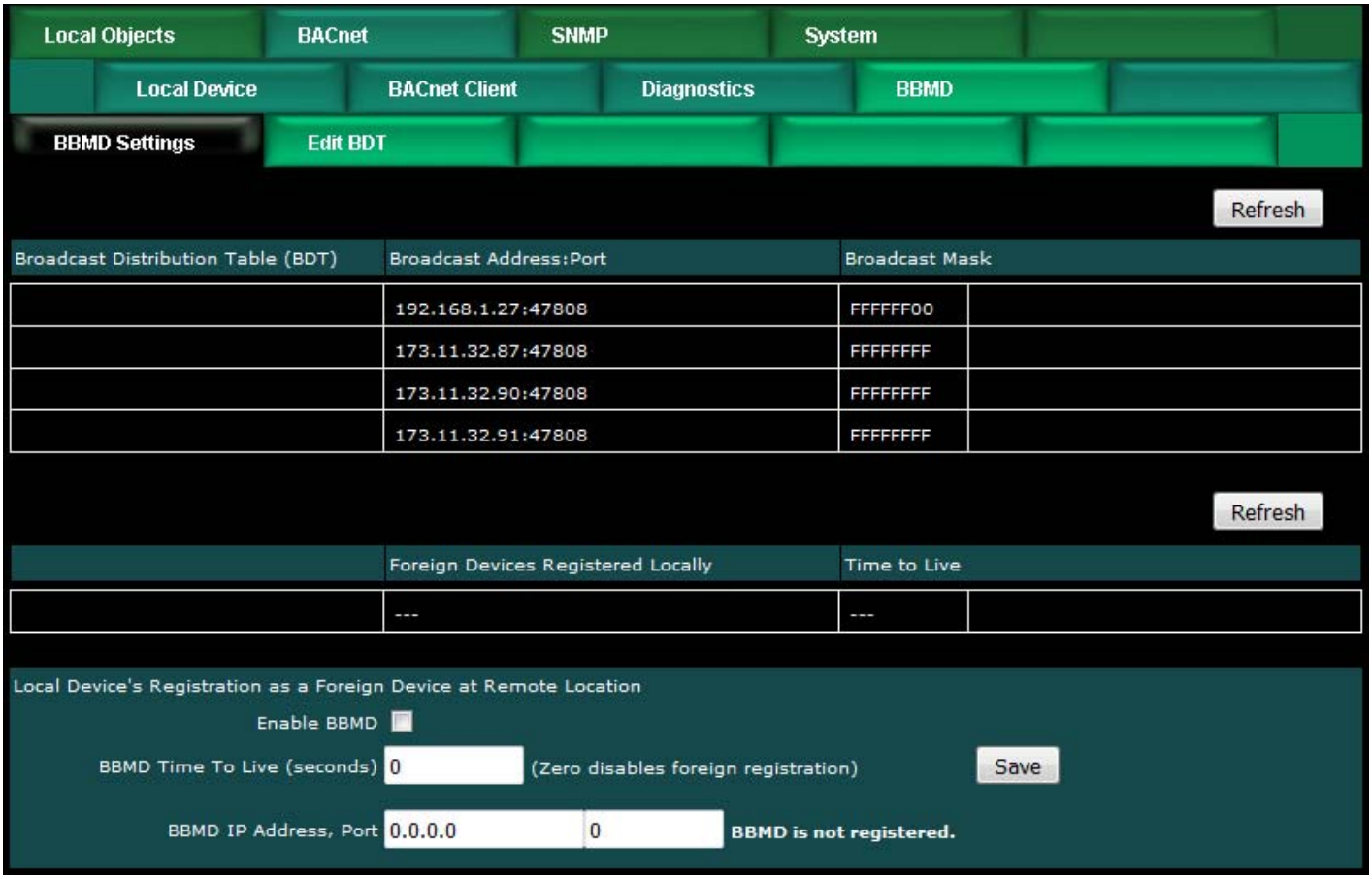
A Broadcast Distribution Table (BDT) defines a list of IP addresses that the BBMD should send broadcast messages to. It is important to note that a BBMD only forwards broadcast messages. It does not do full routing. If you are attempting to connect two networks across a NAT router, you must get a full BACnet Router to accomplish this. For this reason, the BDT has limited usefulness when only BBMD is present. The BBPRO-V230 only includes BBMD, not full routing.

Broadcast distribution will result in device discovery, but you will not be able to read/write properties in the remote device without full routing. Foreign device registration via a router does result in being able to fully communicate with the foreign device from the local network.



Th Edit BDT page allows viewing of the broadcast distribution table that has been provided to the local device by an external network management tool capable of sending the BDT initialize. The BDT may also be edited on this page. Regardless of how the table is filled, it will be saved in the configuration file when saved, and reloaded upon restart.

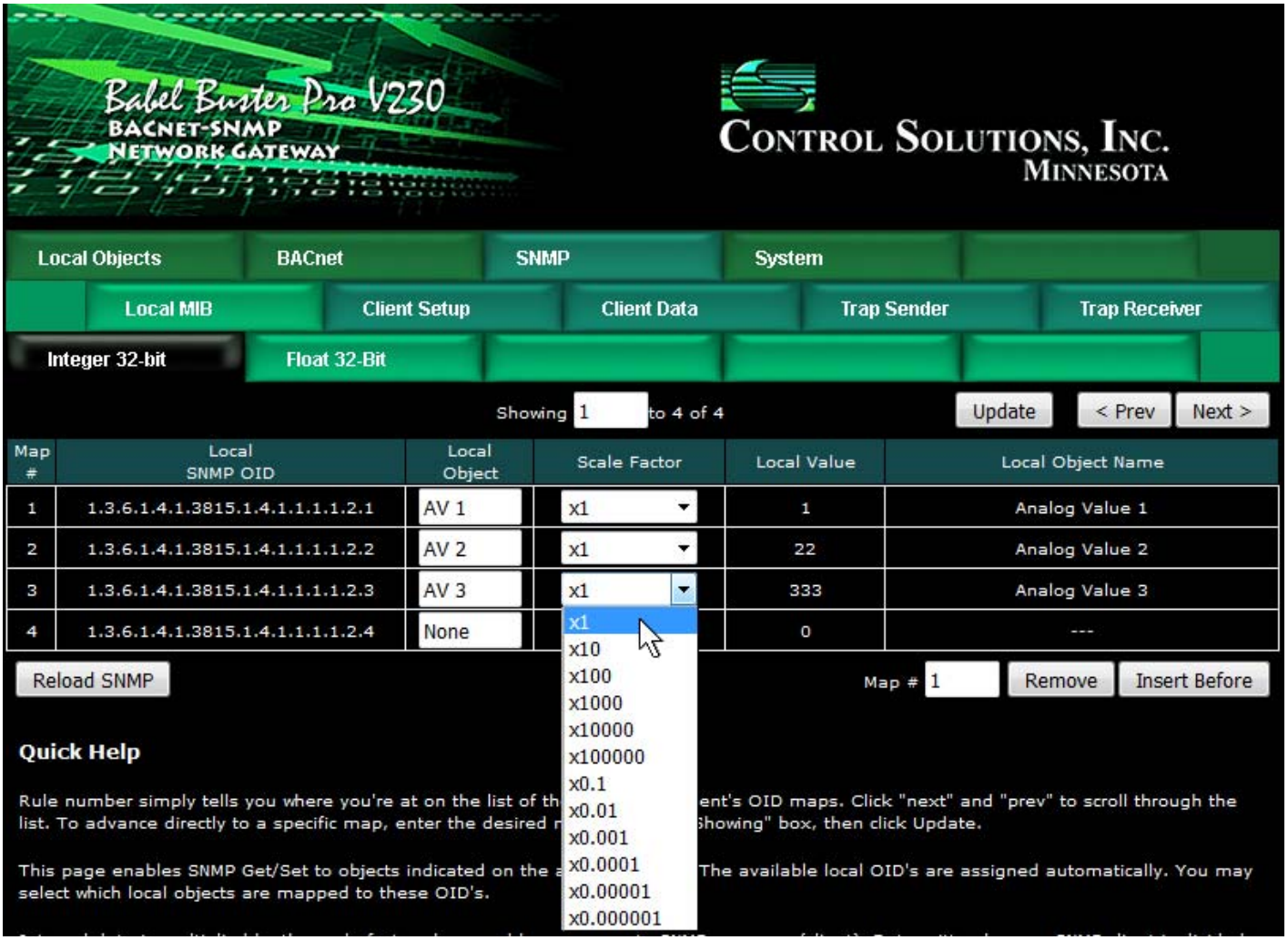
Once the table has been initialized, it will appear on the BBMD Settings page as illustrated below.



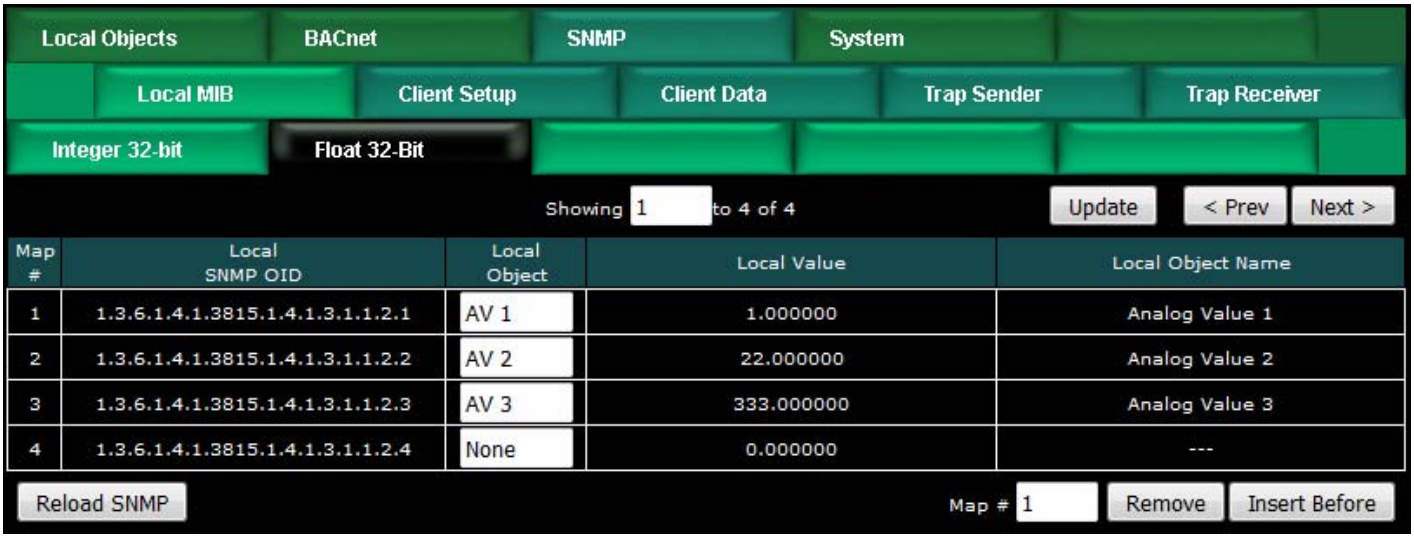
8. Configuring Gateway as an SNMP Server

8.1 Creating Local SNMP MIB

The Babel Buster Pro starts out with no variables in its MIB. When you create local objects, you then have a choice of where to make them show up in the MIB. There are two branches in the BBPRO-V230 MIB, although only the Integer branch is guaranteed to be universally accessible to all other SNMP devices. It is up to you to select which branch of the MIB to place each local object in. You do not need to place all local objects in the MIB, only those that you want externally accessible via SNMP. You must also place local objects in the MIB in order to generate SNMP traps related to those local objects.



To add a local object to a MIB branch, simply enter the local object number at the next available OID which will always automatically be at the bottom of the list. When placing objects in the Integer branch, you also have the option of applying a scale factor. Scaled integer is the most universally recognized means of transmitting non-integer data.You have the option of selecting a scale factor for floating point data provided as integer That means, for example, that a local value of 5.19 with a scale factor of x100 will be transmitted (in a Get response) as integer 519 and it is up to the recipient to know that this is scaled x100.



Local objects added to the Float 32-Bit MIB branch will be provided in IEEE 754 format per RFC 6340.

After adding new members to the MIB, it is necessary to click Reload SNMP before they will become accessible to an external SNMP manager's Get request.

8.2 Supported Data Formats, RFC 6340

SNMP does not have a universally accepted representation for floating point. The one universally known data type is INTEGER. A commonly recomended means of transmitting floating point data is either as a scaled integer or as an ASCII character string. There is an RFC 6340 for representation of floating point based on IEEE 754 encoding. The "Float 32-bit" data type in the Babel Buster Pro refers to RFC 6340 encoding.

Specifically, the data types found in the Babel Buster Pro MIB are encoded with ASN types as follows:

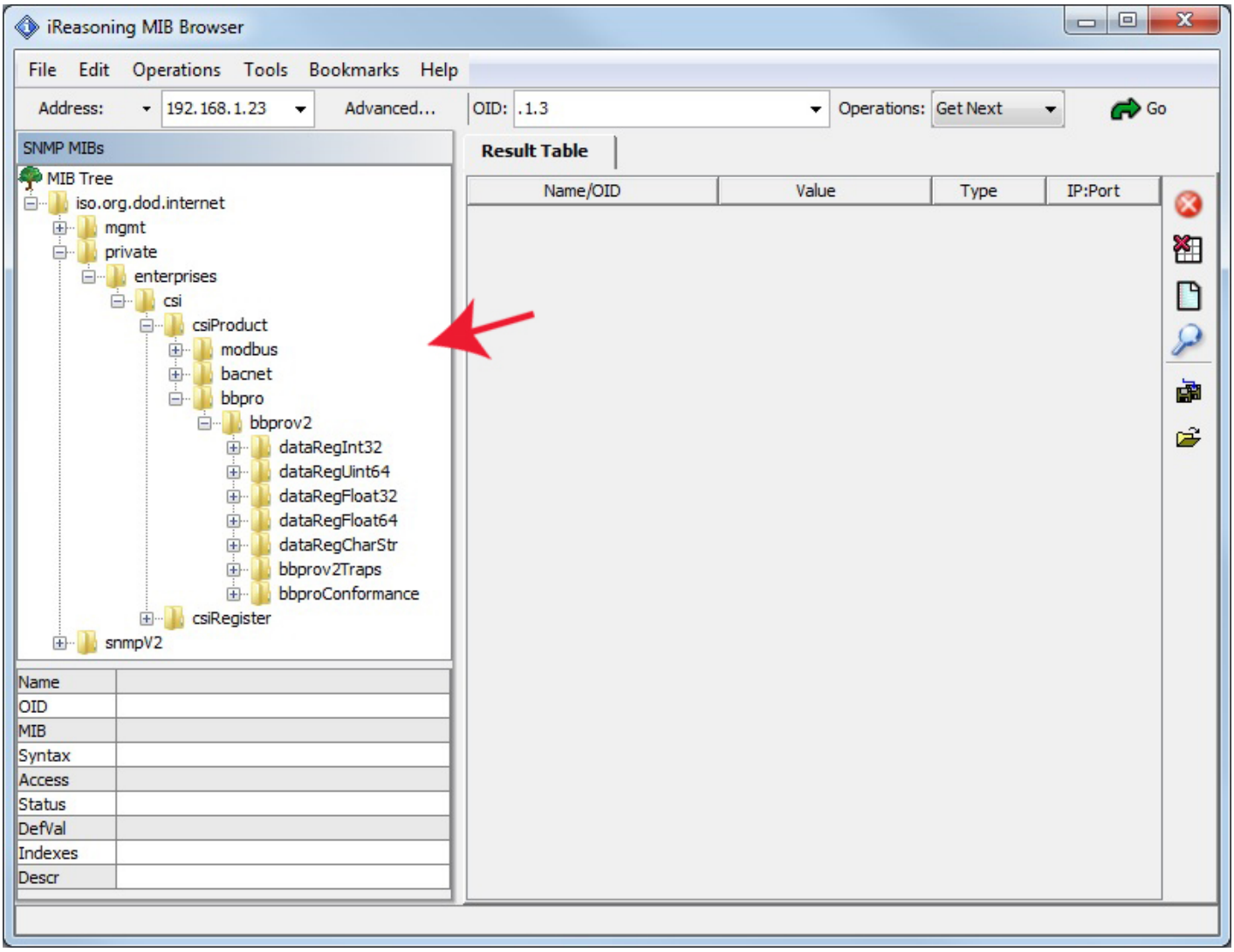
Integer 32-Bit	INTEGER	ASN_INTEGER
Float 32-bit	OCTET STRING	ASN_OCTET_STR (length 4)

8.3 Testing the SNMP Agent

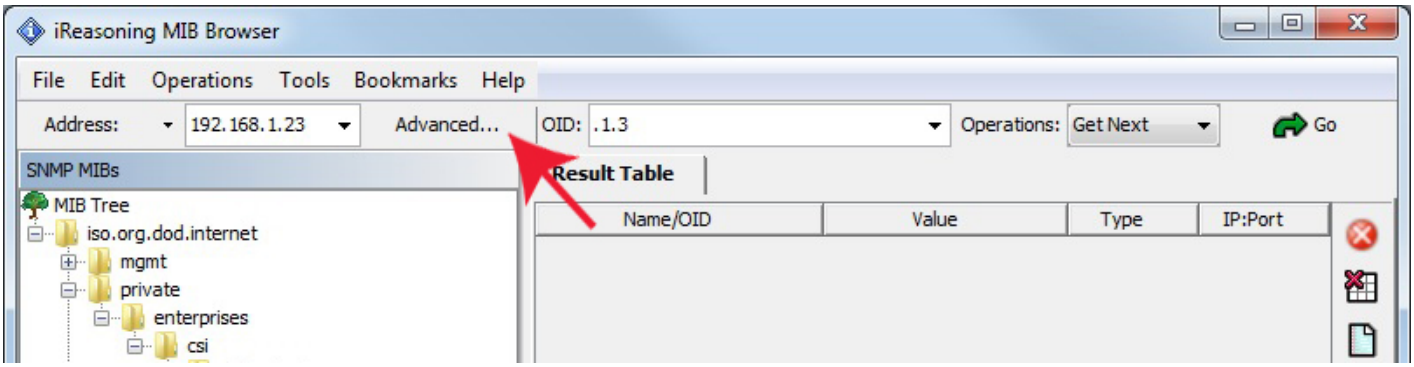
A variety of tools are available for browsing an SNMP MIB and receiving SNMP Traps.

The tool used in the following examples is the iReasoning MIB Browser. Refer to the Tools section under Support at csimn.com for more information about SNMP tools.

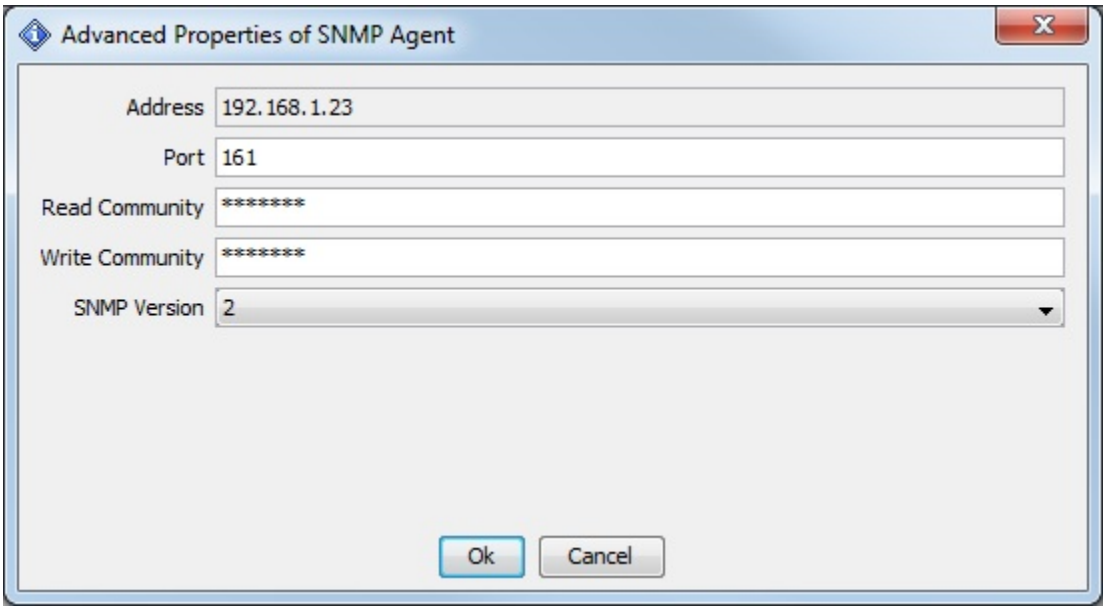
The MIB browser allows you to view the MIB variables in the Babel Buster Pro. Before you can browse the MIB in a meaningful way, you need to load the MIB files that tell the browser what it needs to know about the Pro's MIB. There are three files that need to be loaded, in order: CSIreg.mib, FLOAT-TC-MIB.mib, and CSIbbprov2.mib. Once you have loaded these files as illustrated below, you can view the tree structure of the MIB in the browser.



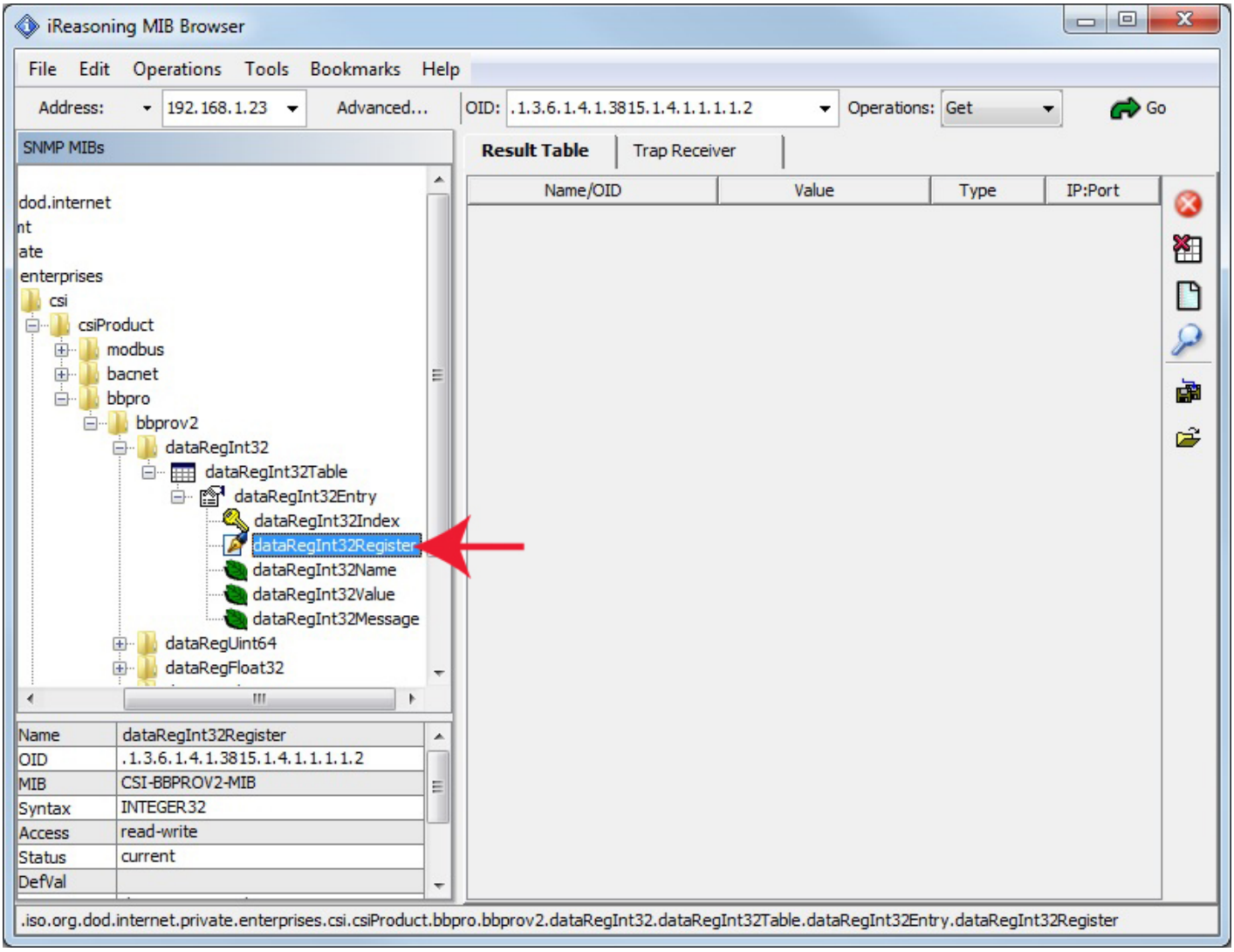
Enter the IP address of the Babel Buster Pro in the Address window. In addition. click on Advanced... to set access parameters.



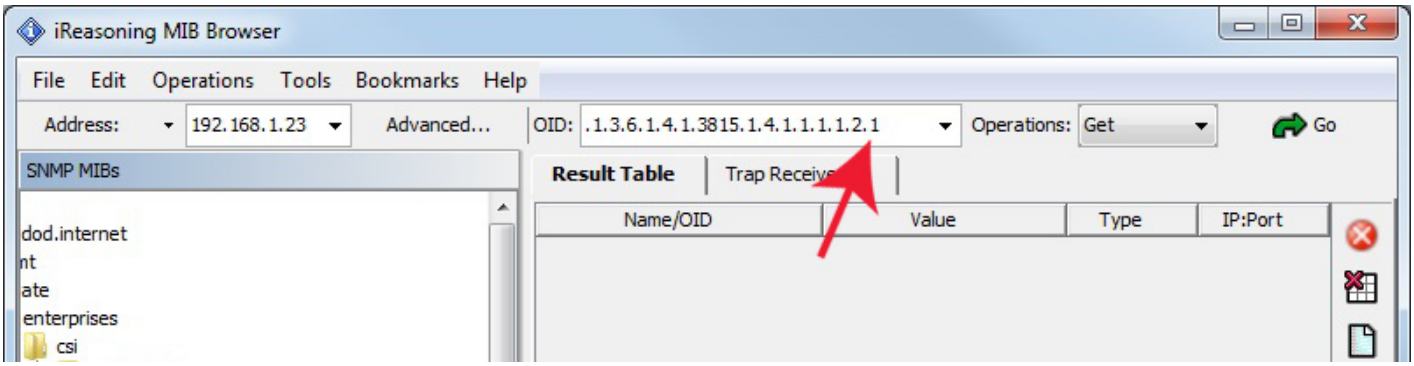
Click Advanced... to open the dialog that lets you enter the community string for the Babel Buster Pro. This work ssomewhat like a password. If the community string you provide here does not match the community string set in the Babel Buster Pro, you will be unable to access MIB variables in the Babel Buster. You also need to select the correct SNMP version number in order to get the correct result. Babel Buster Pro MIB is version 2. (Babel Buster Pro can send traps as either v1 or v2c, but Get/Set must be v2.)



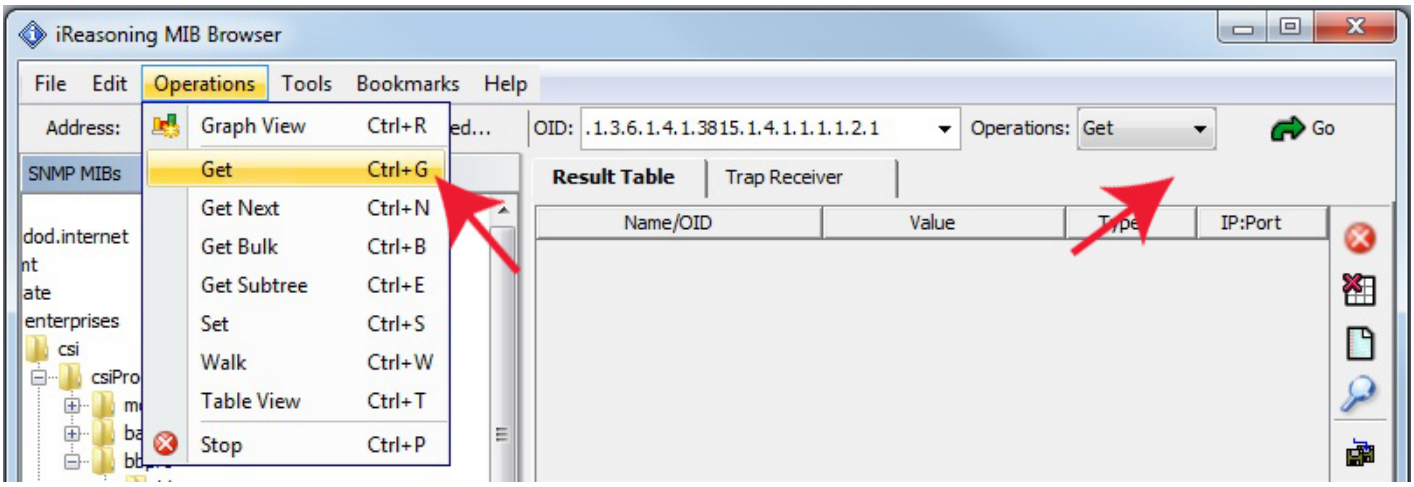
To read a MIB variable from the Babel Buster Pro, start by selecting the register member of the data table. In this case, we are selecting the 32-Bit Integer branch of the MIB.



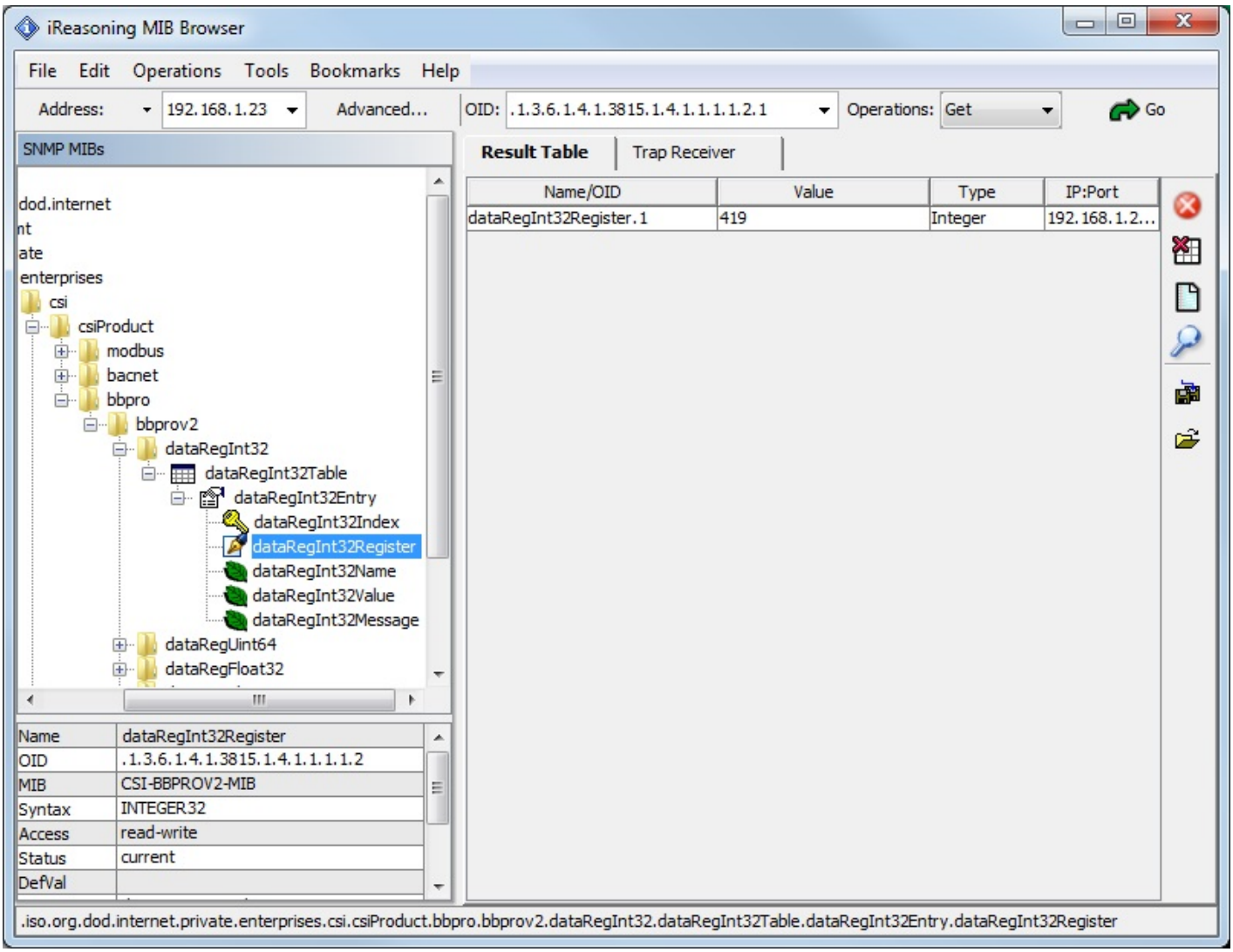
You will need to specify which row in the table you want to read. Do this by appending (by typing) a number to the OID that appeared in the OID window when you clicked on the table entry in the MIB tree view.



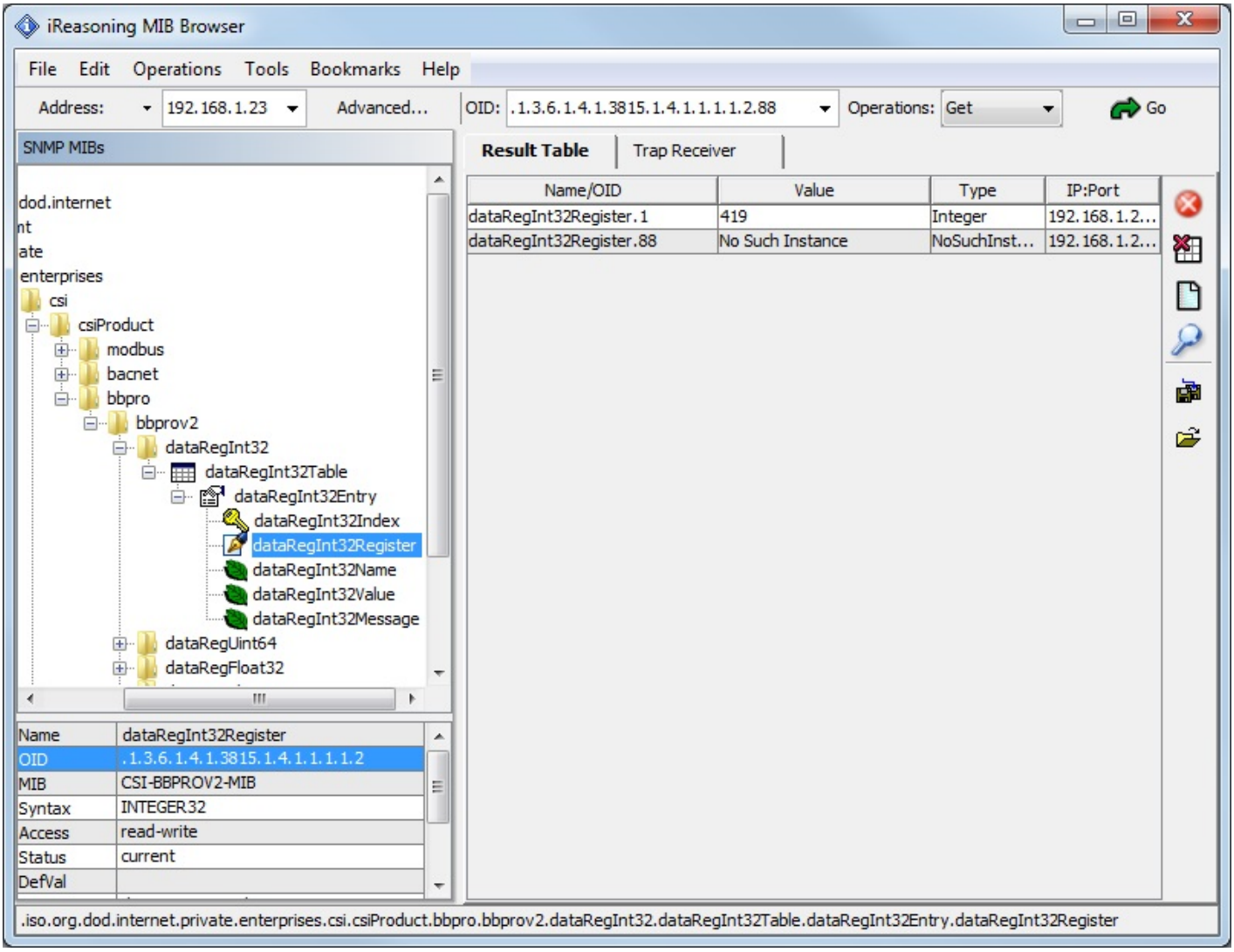
To cause the MIB browser to Get a value from the Babel Buster Pro, select Get from the Operations menu. You can select Get from either menu, left or right, in the iReasoning browser. Once Get is selected on the right, you only need to click the Go button to repeat the Get.



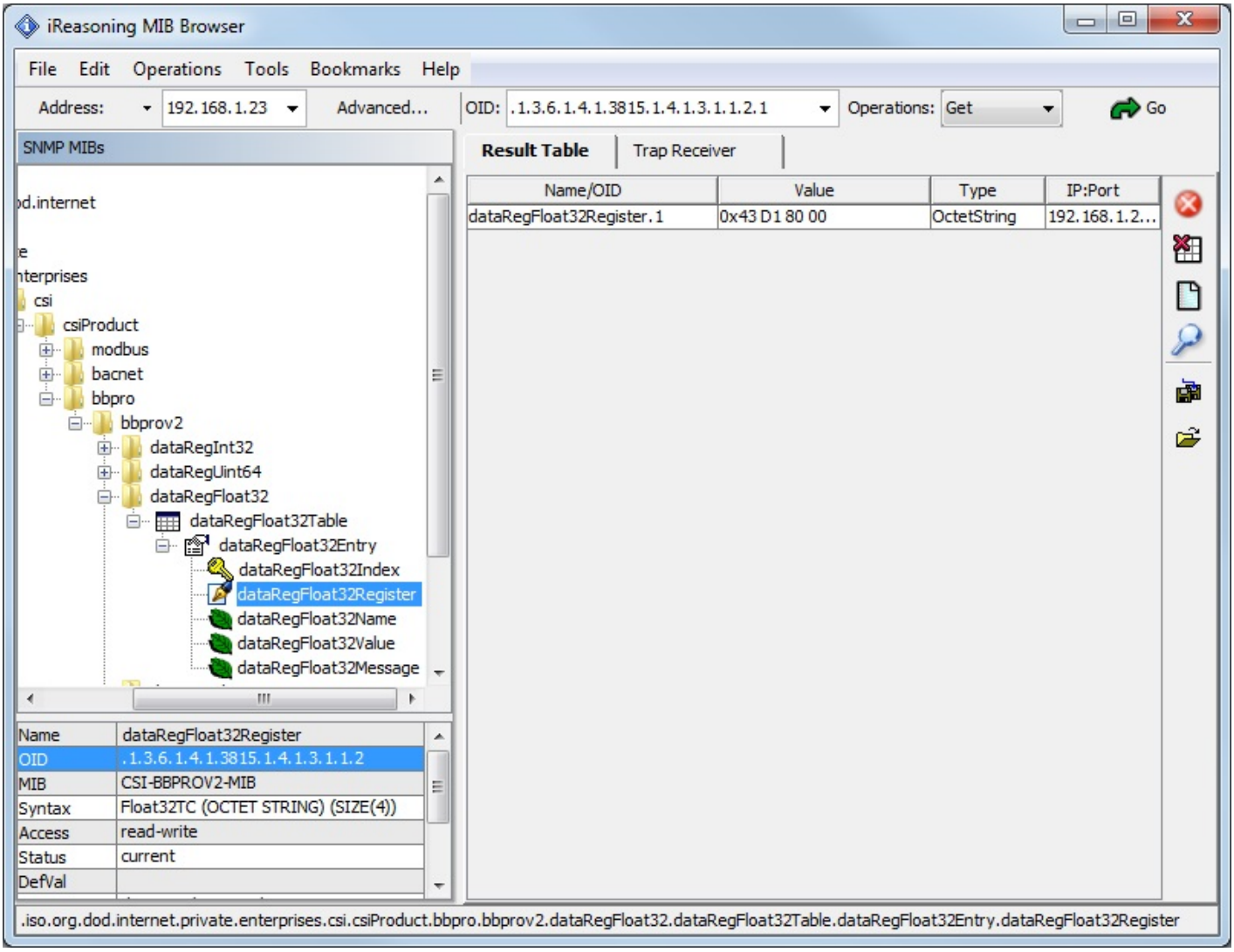
Upon successfully Getting a value, the display will appear as illustrated below.



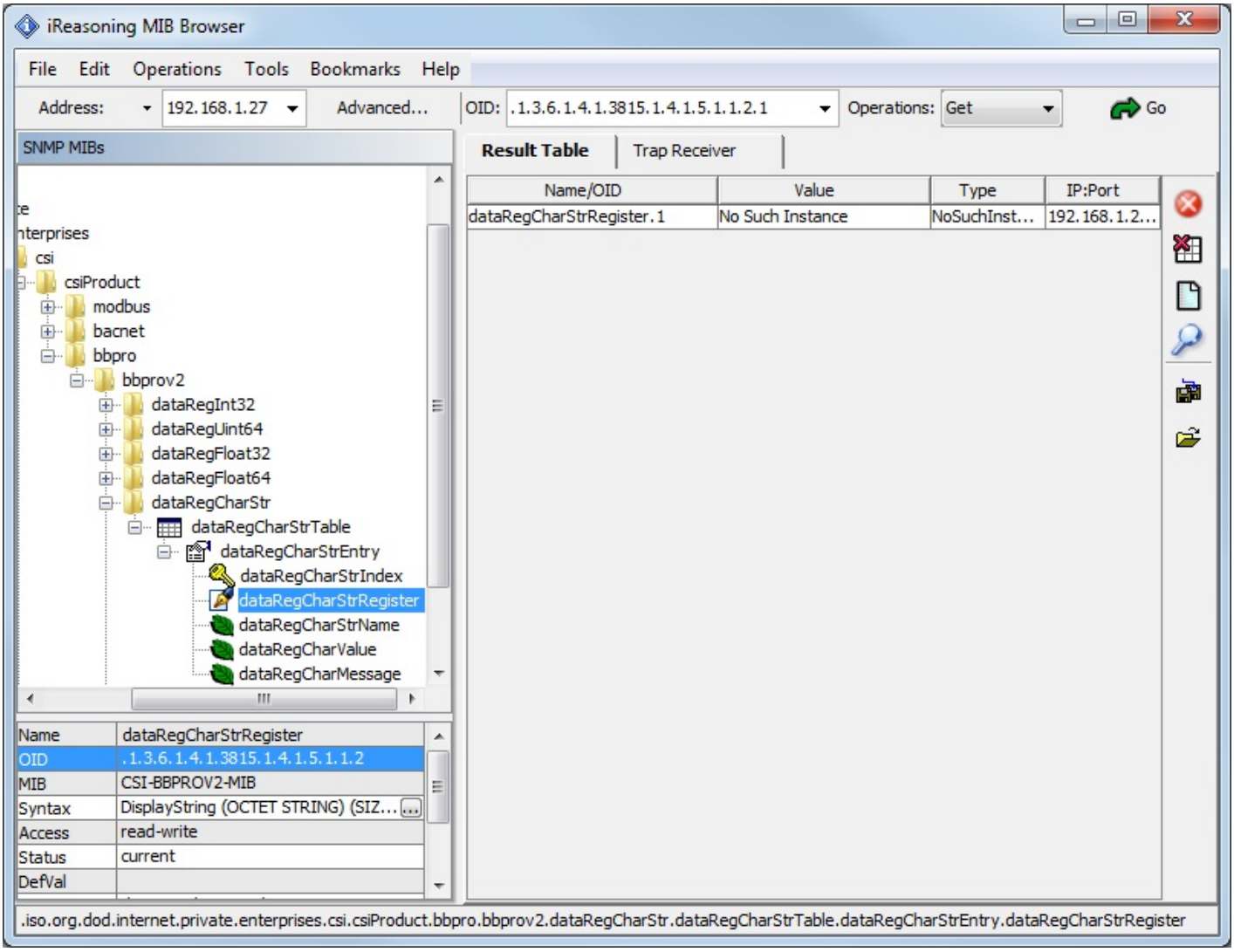
If you attempt to Get a variable that does not exist, you will get the error message illustrated below. This will also happen if the variable has been added to the MIB but you forgot to click Reload SNMP after adding the local register to the MIB. This will also happen if you did not select the right variable from the MIB tree shown, or forgot to add the index to the end of the OID in the Object ID window.



Getting our floating point value will appear as follows. The bytes "43 D1 80 00" translate to the floating point version of 419.00. You can use Google to locate conversion tools to go from IEEE 754 hexadecimal representation (as shown in the browser) to decimal.



The MIB definition for Babel Buster Pro covers both Modbus and BACnet versions of the gateway. The Modbus version has MIB branches that do not exist in the BACnet version. If you attemt to Get one of these (e.g. Character String), you will see the "No Such Instance" message illustrated below.

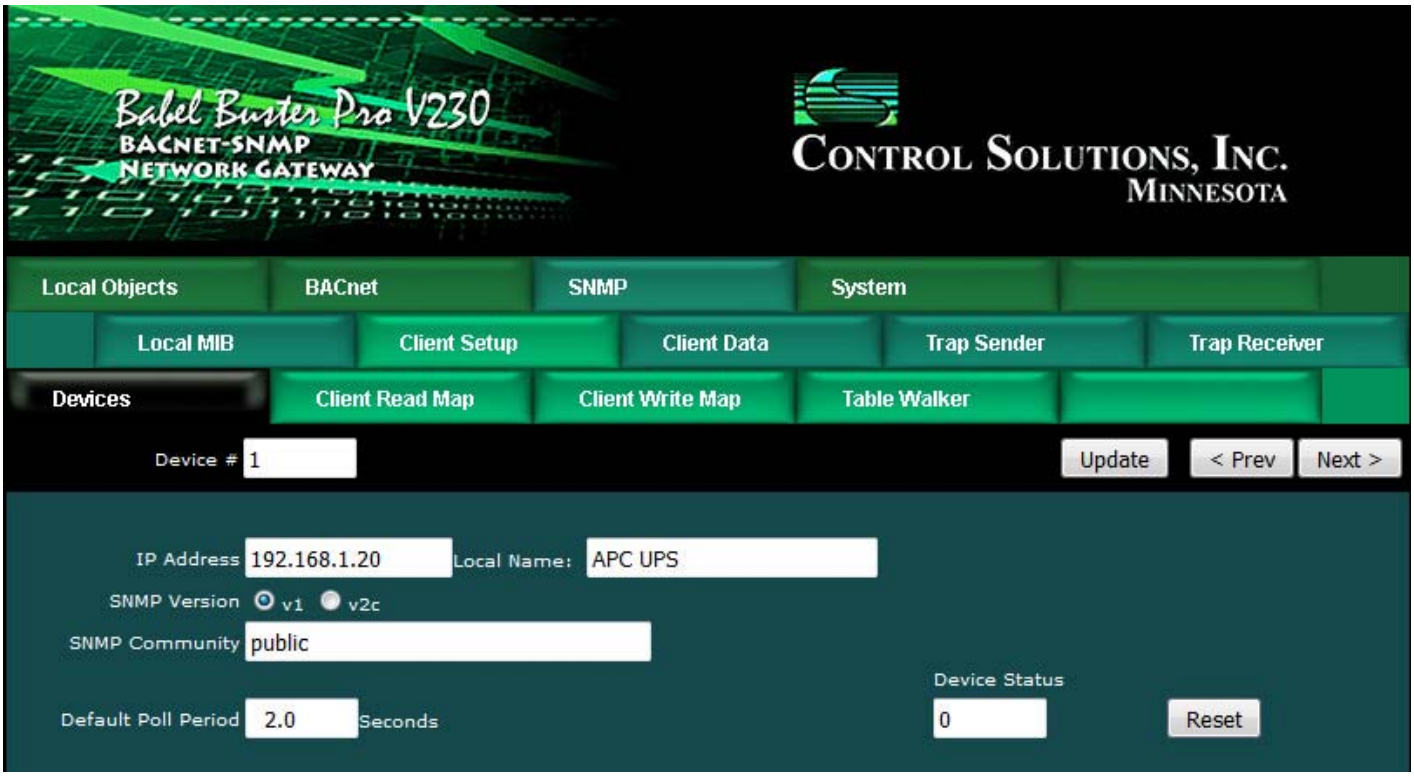

iReasoning MIB Browser
File Edit Operations Tools Bookmarks Help
Address: 192.168.1.27 Advanced... OID: .1.3.6.1.4.1.3815.1.4.1.5.1.1.2.1 Operations: Get Go
SNMP MIBs
nterprises
csi
csiProduct
modbus
bacnet
bbpro
bbprov2
dataRegInt32
dataRegUint64
dataRegFloat32
dataRegFloat64
dataRegCharStr
dataRegCharStrTable
dataRegCharStrEntry
dataRegCharStrIndex
dataRegCharStrRegister
dataRegCharStrName
dataRegCharValue
dataRegCharMessage
Result Table
Trap Receiver
Name/OID
Value
Type
IP:Port
dataRegCharStrRegister.1
No Such Instance
NoSuchInst...
192.168.1.2...
Name dataRegCharStrRegister
OID .1.3.6.1.4.1.3815.1.4.1.5.1.1.2
MIB CSI-BBPROV2-MIB
Syntax DisplayString (OCTET STRING) (SIZ...
Access read-write
Status current
DefVal
.iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegCharStr.dataRegCharStrTable.dataRegCharStrEntry.dataRegCharStrRegister

9. Configuring Gateway as an SNMP Client

The SNMP Client is used to read and write (Get and Set) data in other SNMP devices. Data read from a remote SNMP device is stored in a local object when received. Data written to a remote SNMP device is taken from a local object when sent. The local objects are the same collection of objects that are accessible to BACnet or to other SNMP devices via the Babel Buster Pro's own MIB.

9.1 SNMP Device Configuration

The list of other SNMP devices that will be accessed by the Babel Buster Pro are entered on the Devices page under SNMP Client Setup.



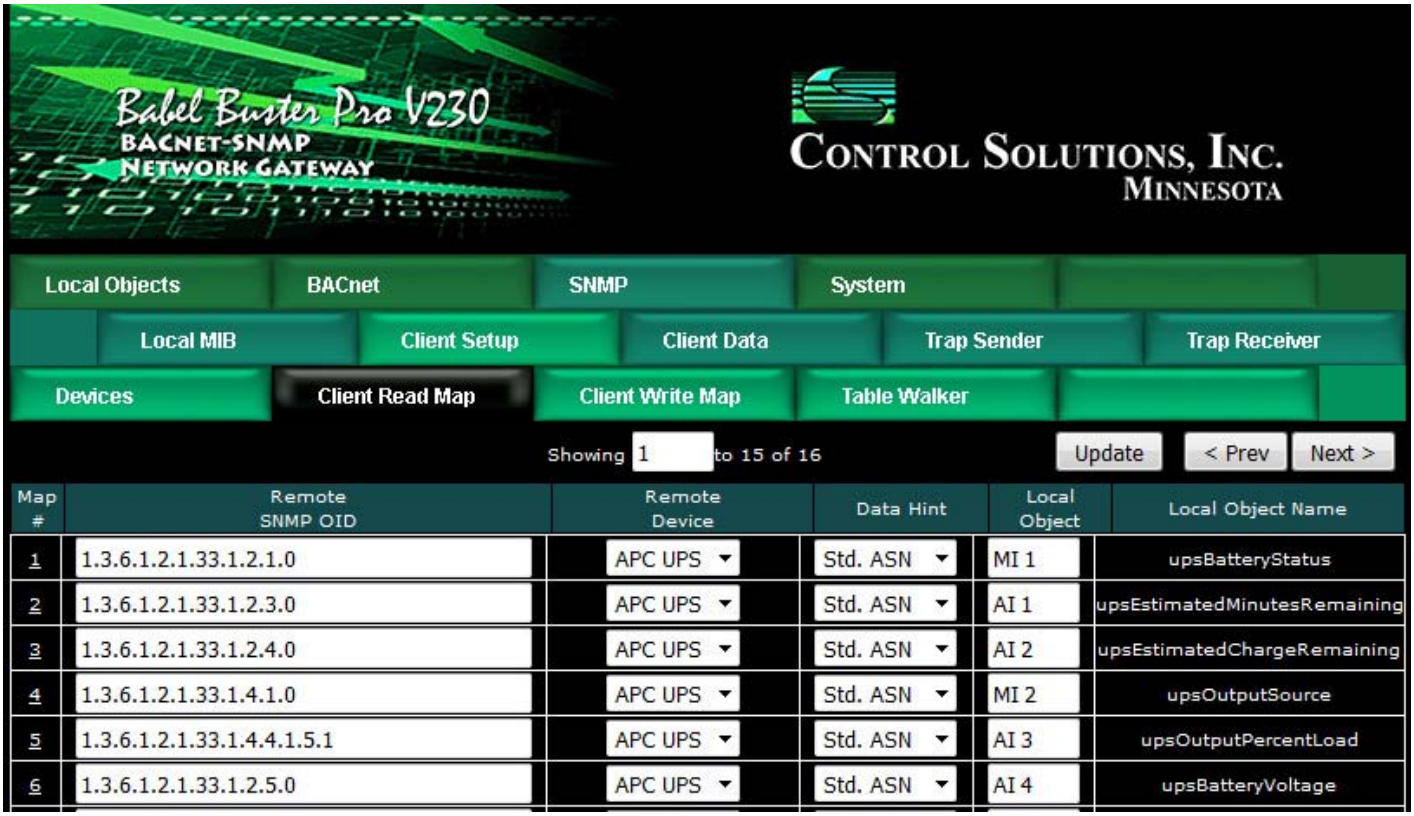
Enter the IP address of the SNMP agent that the Babel Buster Pro will send Get or Set requests to. Provide a local name for this device. This local name will appear in the device list when setting up Client Read Maps and Client Write Maps. Select whether v1 or v2 requests should be sent to this device (v3 is not supported by this model gateway). Your Get or Set request will not be honored by the remote SNMP device if you do not provide the correct community string. Enter the community here. You will

need to obtain that from the device you will be Getting or Setting.

Enter a default poll period. This sets the rate at which the Babel Buster Pro will periodically Get or Set the variables identified in the Read and Write Maps.

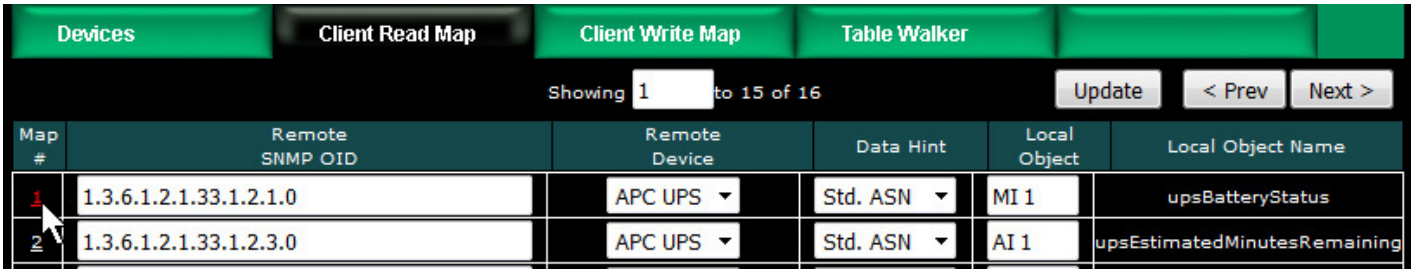
9.2 SNMP Client Read Maps (Get)

Reading or Getting data from another SNMP device requires setting up a Read Map.

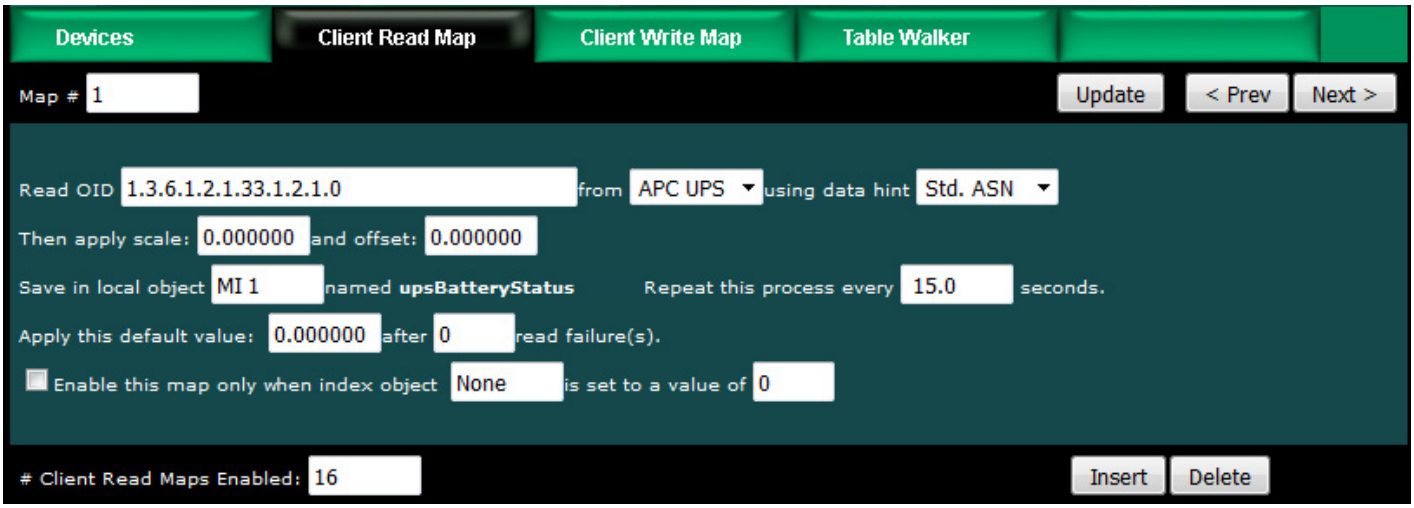


To configure a Read Map, enter the OID of the variable you wish to Get, select the SNMP device from the device list, and provide a local object number where the received data should be placed. If the remote device will be providing floating point as RFC 6340, select that in the Data Hint list, otherwise leave the selection as "Std. ASN" which means stanard ASN encoding.

Data hint helps the data parser figure out what the varible is. If ASN encoding is recognized as anything other than an octet string, the hint will be disregarded. If an octet string is found, then the parser needs to know if it should be treated as RFC 6340 floating point. If no hint is given (standard ASN), then the octet string will be treated as an ASCII character string, in which case ASCII to numeric conversion will be attempted automatically if the local object is numeric (string will simply be copied if result object is a character string type object). Note that standard, well-known ASN types are recognized as well as the NetSnmp ASN type for opaque float.



Click on the Map number in the first column to access the expanded view of the Read Map where additional optional parameters may be entered.



The optional scale and offset may be used if the value being read is scaled, or you may simply use scale and offset to perform units conversions. If non-zero, the received data will be multiplied by scale, then added to offset, before being placed in the local object.

If the remote SNMP device should be read at some rate other than the default poll time given on the Devices page, enter that here.

If you wish to have a specific default value set in the local object in the event the Get fails, enter that default value and some non-zero number of read failures. The default value will be set after this number of failed attempts.

You have the option of making this Read Map conditional. If an index object number is provided and the Enable box is checked, then this read map will only be executed when the index object (local object) contains the value given. This allows multiple read maps to supply data to the same local object based on the value of the index object. It also allows reading to simply be suspended if a single read map supplies data to the local object. In a more sophisticated scenario, you could potentially suspend reading of the remote SNMP device if you know the device is powered down.

Map number simply tells you where you're at on the list of Read 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.

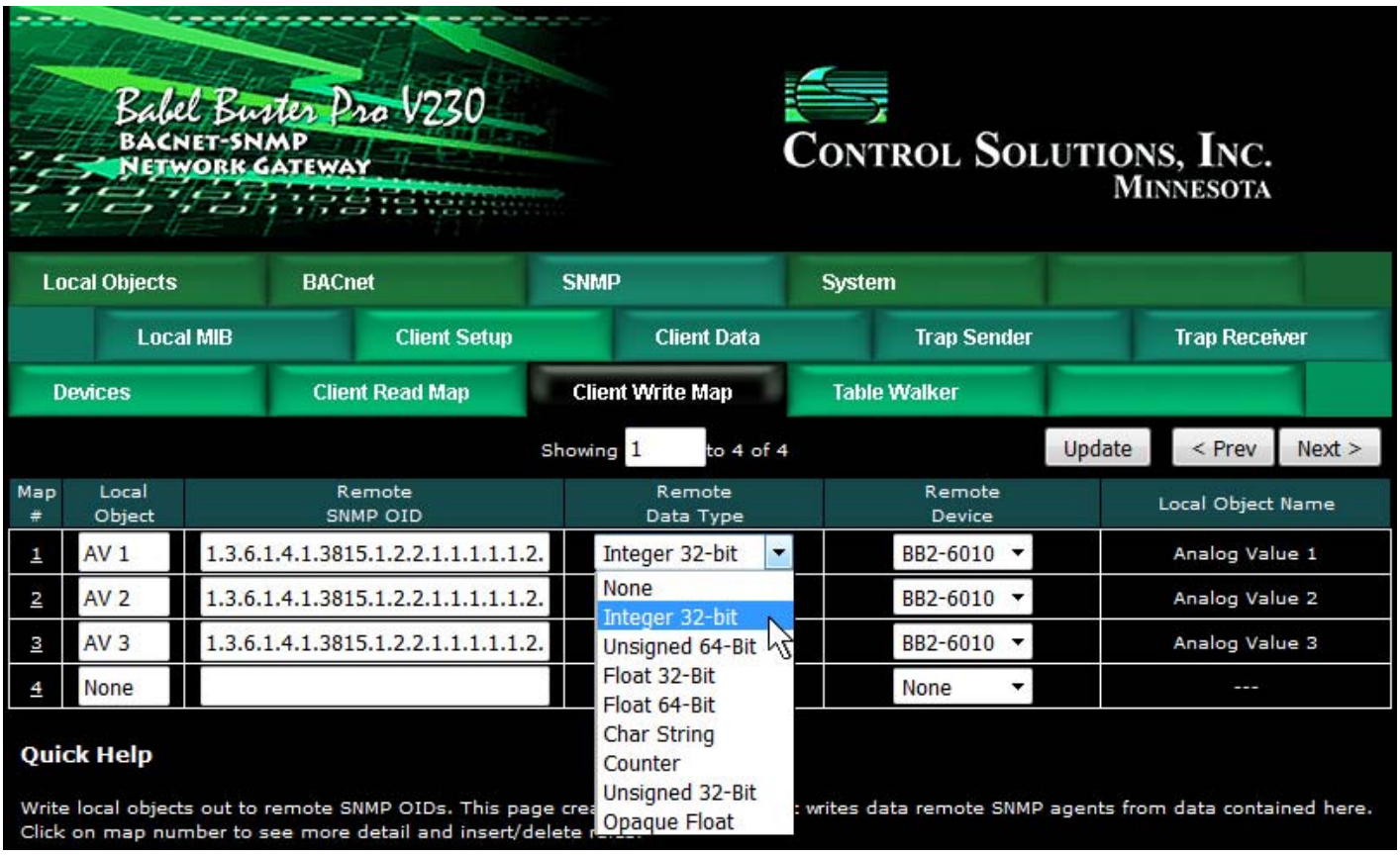
Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object 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.

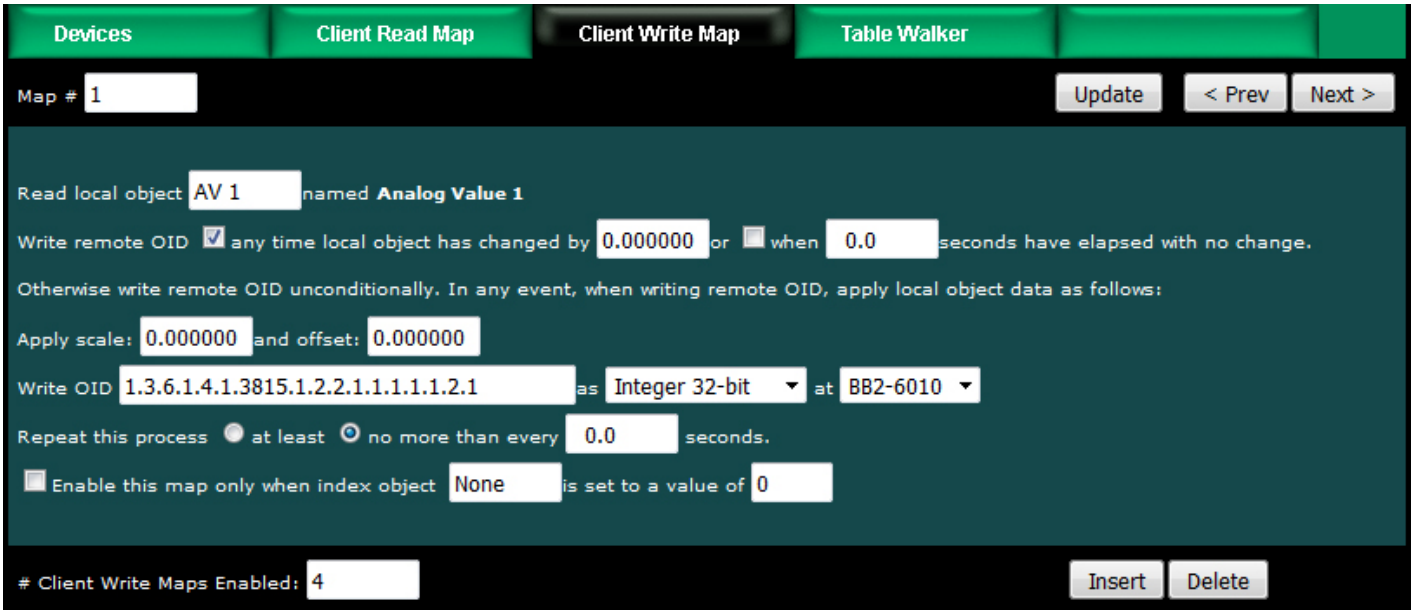
9.3 SNMP Client Write Maps (Set)

Writing or Setting data to another SNMP device requires setting up a Write Map.



Start by selecting the local object that will be the source of data to Set in the remote device. Enter the OID to write in that device. Select the data format that will be expected by that device at the OID given. Select a device from the device list. Only devices entered on the Devices tab will appear in this list.

Click on the Map number in the first column to access the expanded view of the Write Map where additional optional parameters may be entered.



The local object data may be written to the remote SNMP device 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 OID 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 OID. The local data is first multiplied by the scale factor. The offset is then added to it.

Enter the OID to write in that device. Select the data format that will be expected by the selected device at the OID given. Select a device from the device list. Only devices entered on the Devices tab will appear in this list.

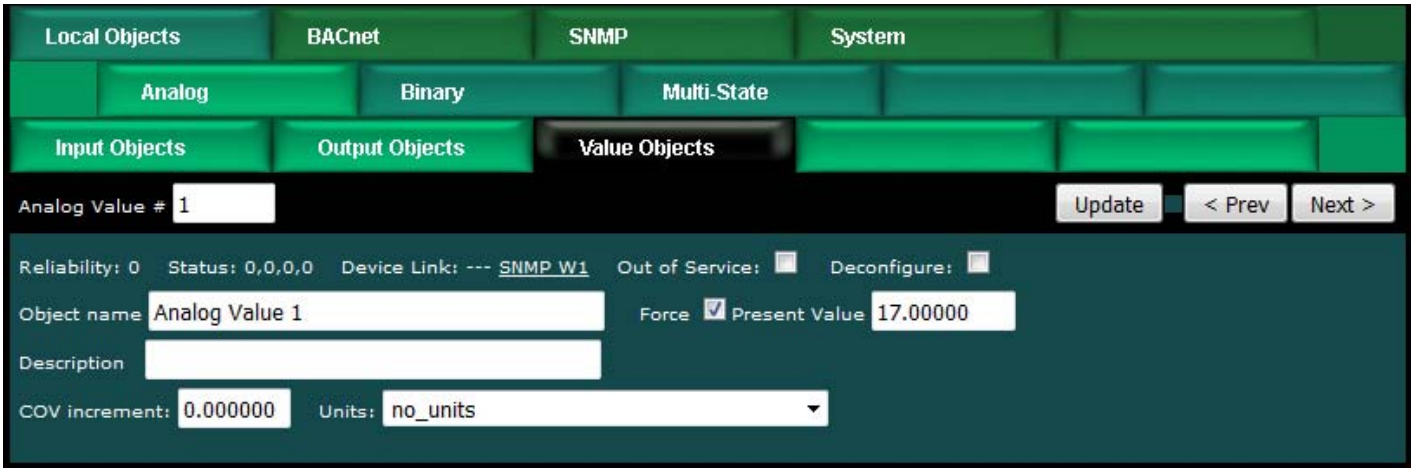
The repeat time may determine how often the remote OID will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic in the event changes are frequent, click the "no more than" button and enter the minumum time that should elapse before another write to the remote device. It is valid to select "no more than every 0.0 seconds" if you want all changes to be sent, but no periodic writes.

You have the option of making this Write Map conditional. If an index object number is provided and the Enable box is checked, then this write map will only be executed when the index object (local object) contains the value given. This allows multiple write maps to supply data to the same remote OID based on the value of the local index object. It also allows writing to simply be suspended if a single write map supplies data to the remote OID. In a more sophisticated scenario, you could potentially suspend writing of the remote device if you know the device is powered down.

Delete will remove the map 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.

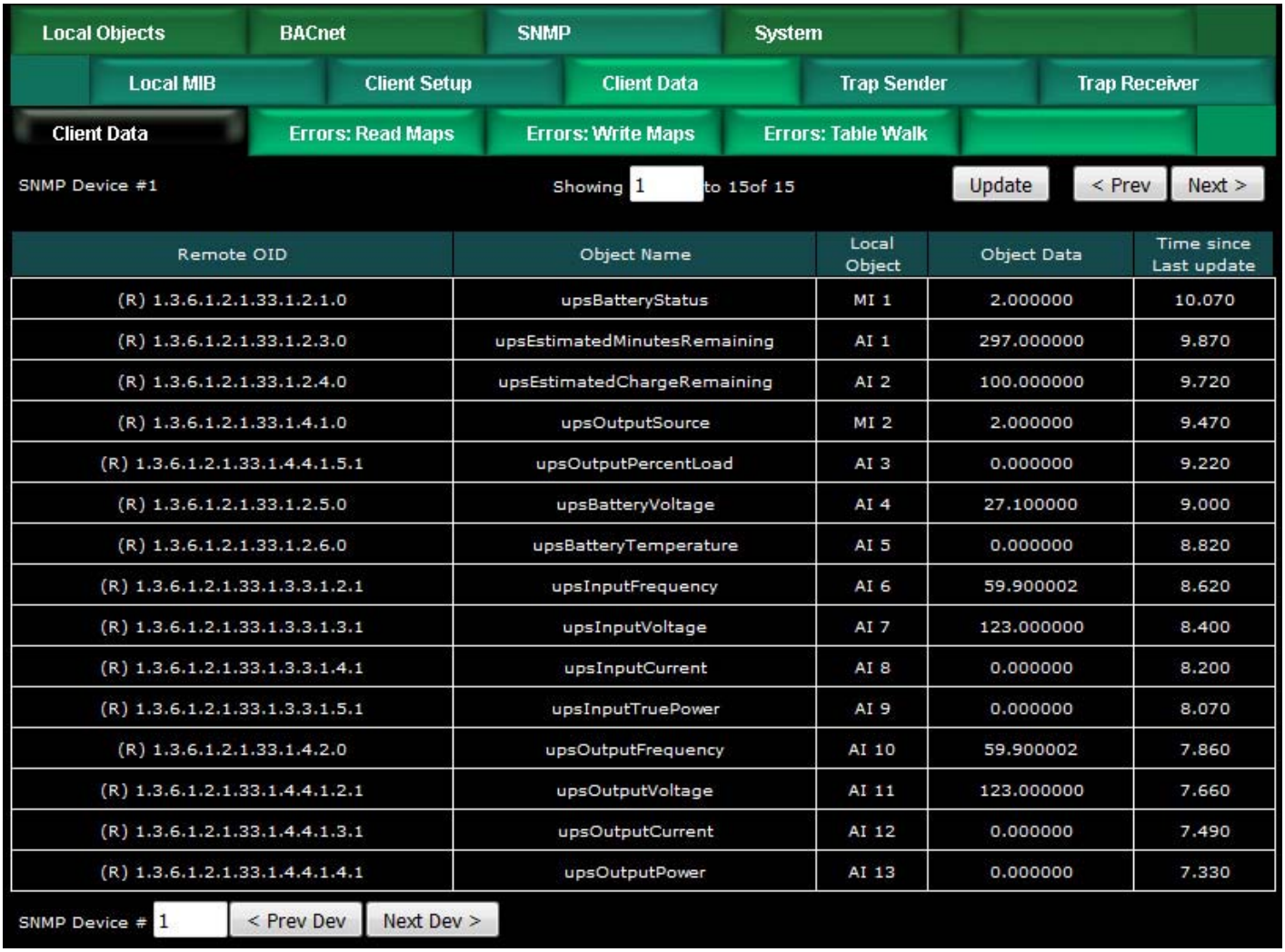
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.



To test Setting the mapped OID, you can go to the Local Objects page, enter a new value for the local object, check Set, and then click Update. If monitoring traffic in Wireshark, you should see the request go out.

9.4 SNMP Client Data Displayed by Agent

The Client Data page shows the list of local objects mapped to a remote SNMP device. The SNMP Device # indicated at the bottom of the page refers to devices in the Device list under SNMP Client Setup.



Click the Next Dev or Prev Dev buttons to go to the next or previous SNMP device, or simply enter a number in the SNMP Device # window and click Update. Maps for that device will now be displayed. In addition to a summary of the map (both read and write maps are shown), the time since last update is displayed. This time should generally be less than the poll time. If the last update time is large, it may mean there is an error preventing the update.

9.5 Supported Data Formats

SNMP does not have a universally accepted representation for floating point. The one universally known data type is INTEGER. A commonly recomended means of transmitting floating point data is either as a scaled integer or as an ASCII character string. A well known but application specific implementation (NetSNMP) uses ASN OPAQUE FLOAT. There does exist an RFC 6340 for representation of floating point. Both the NetSNMP and RFC 6340 versions are based on IEEE 754 encoding. The "Float 32-bit" and "Float 64-bit" data types in the Babel Buster Pro's own MIB refer to RFC 6340 encoding.

Data types recognized by Babel Buster Pro when Getting or Setting SNMP data in other devices are listed in the table below. Specifically, the data types are encoded with

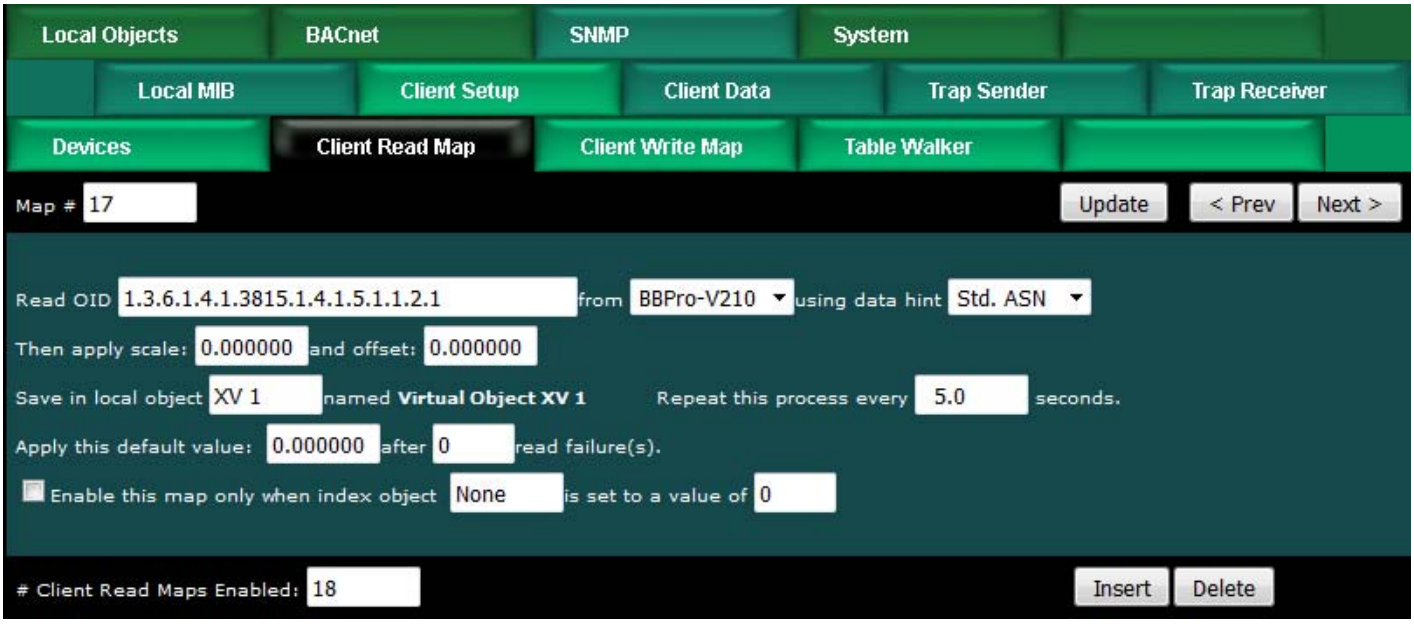
ASN types as follows:

Integer 32-Bit	INTEGER	ASN_INTEGER
Unsigned 64-Bit	COUNTER64	(ASN_APPLICATION | 6)
Float 32-bit	OCTET STRING	ASN_OCTET_STR (length 4)
Float 64-bit	OCTET STRING	ASN_OCTET_STR (length 8)
Char String	OCTET STRING	ASN_OCTET_STR (length variable)
Counter	COUNTER	(ASN_APPLICATION | 1)
Unsigned 32-bit	UNSIGNED	(ASN_APPLICATION | 2)
Float-Opaque	OPAQUE FLOAT	(ASN_APPLICATION | 8)

9.6 Virtual Objects for SNMP Access from Basic

The BBPRO-V230 contains a set of "virtual objects" that provide a data connection between SNMP and Basic without going through BACnet. This is sometimes needed because the SNMP data type known as "Octet String" is defined only as a string of bytes and those bytes can contain anything, including raw binary data. The role of the Virtual Object is to allow transferring of data directly between SNMP and Basic so that Basic can then make customized translations as necessary to derive meaningful BACnet data.

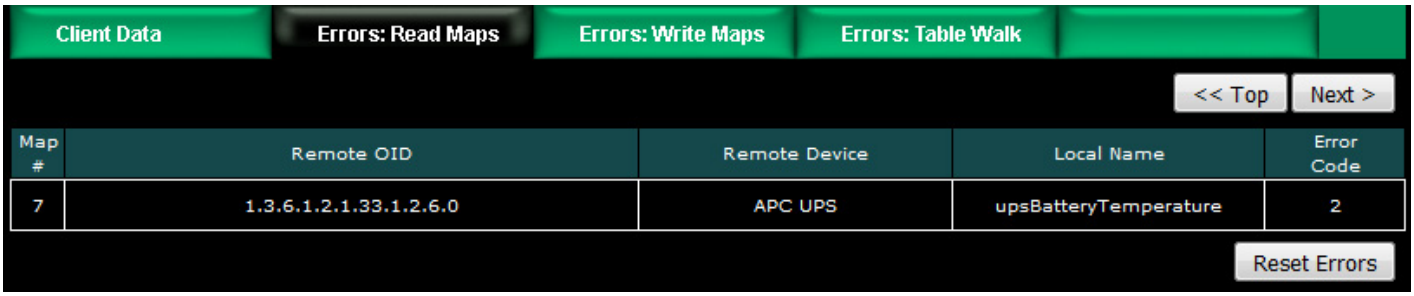
Virtual Objects are referenced as "XV n" where 'n' counts from 1 to the maximum allocated on the Resources page. In Script Basic, they are accessed using the 'getvar' and 'setvar' functions.



9.7 SNMP Errors

The errors pages will show Read Maps and Write Maps that currently have an error

status.



Error codes that may be returned by the SNMP client are as follows:

Standard SNMP error codes returned by remote Agent:
1 = SNMP_ERROR_tooBig
2 = SNMP_ERROR_noSuchName
3 = SNMP_ERROR_badValue
4 = SNMP_ERROR_readOnly
5 = SNMP_ERROR_genErr
6 = SNMP_ERROR_noAccess
7 = SNMP_ERROR_wrongType
8 = SNMP_ERROR_wrongLength
9 = SNMP_ERROR_wrongEncoding
10 = SNMP_ERROR_wrongValue
11 = SNMP_ERROR_noCreation
12 = SNMP_ERROR_inconsistentValue
13 = SNMP_ERROR_resourceUnavailable
14 = SNMP_ERROR_commitFailed
15 = SNMP_ERROR_undoFailed
16 = SNMP_ERROR_authorizationError
17 = SNMP_ERROR_notWritable
18 = SNMP_ERROR_inconsistentName

Client generated errors:
201 = No response from remote Agent (server)
202 = No such name (implicit)
203 = Unable to interpret application data
204 = Reply does not match request
205 = There is a problem with the map configuration
206 = Unable to build or parse SNMP PDU
207 = Table walk came up short of expected count

If the error code indicates no response, the device status (Client Setup Devices page) will provide an additional indication of a connection related error.

Device status may indicate the following application level error conditions:
10 = could not bind socket
11 = response timed out (may mean IP address given is not reachable)

Device status may also indicate error codes from 80 and up are IP stack errors. Since SNMP uses UDP, which is connection-less, the only likely errors are:

95 = Network is unreachable
107 = Host is unreachable
108 = Host is reachable, but expected port cannot be opened

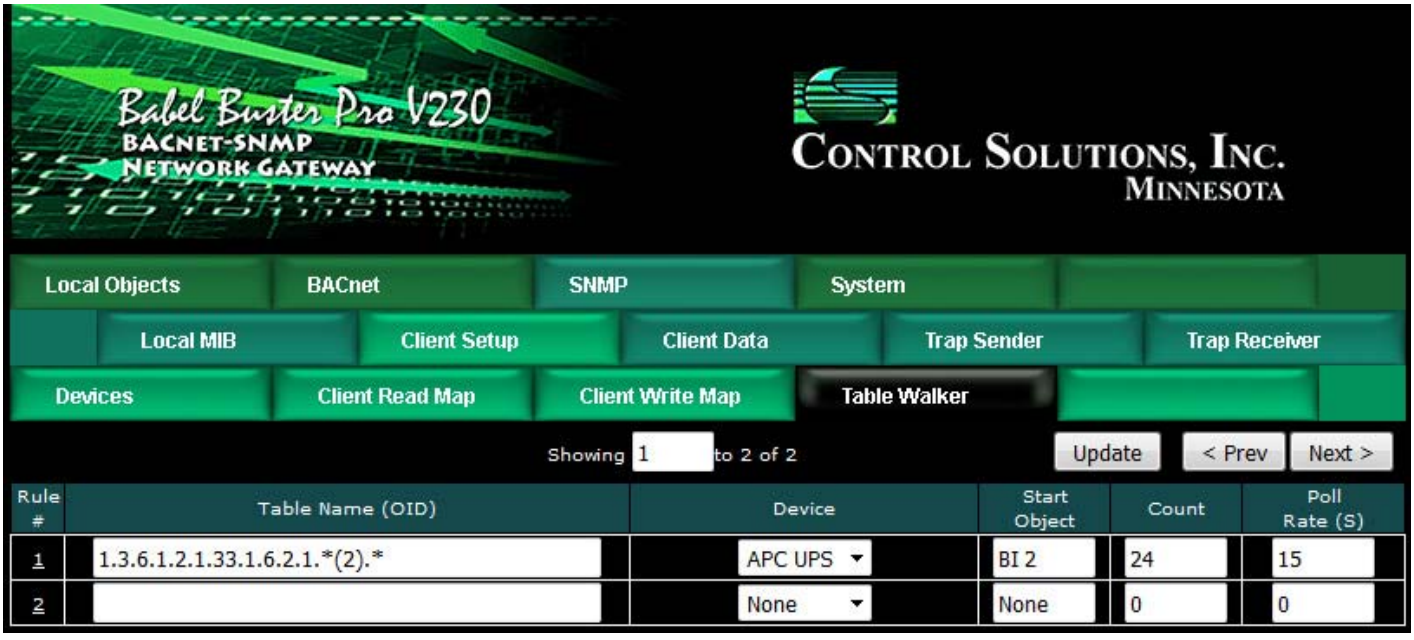
10. Configuring SNMP Table Walker

10.1 Table Walk Methods

Visualize the SNMP table as a spread sheet. Although there can technically be any number of dimensions to an SNMP table, the table walker here is limited to 2-dimensional tables, just like a spread sheet. The table consists of some number of columns each of which are identified by an OID. The table contains any number of rows, which are identified by a table index. The table index, or row number, simply appears as the last field in the OID (or in other words, append the row number to the OID that identifies the column).

The Babel Buster table walker will scan the table, picking out one column per walk rule. For each row in the table, that column will be retrieved and its data placed into the local object provided. The first data value retrieved will be placed into the starting local object, and the walk will continue until the number of objects specified by "count" are filled.

The table walk method will default to "Normal" if you use only the quick entry of a simple table walk rule. This means you enter a starting OID, select a device, enter a starting object and number of local objects to fill with table content, and set a poll rate (meaning how often to periodically walk the table).



To select any table walk method other than Normal, you need to click on the table walk rule number in the first column, and select the method on the expanded view of the walk rule.

The table walking process can become complicated by the fact that tables are allowed to have missing rows, or rows may have intermittent missing columns. In some applications, rows will be populated only temporarily and then they will disappear. The table walk method identified as "Normal" expects the given column to exist in all rows, and at all times, through the range of rows defined by the starting OID and the count. The remaining methods accommodate the other complications permitted by SNMP.

Method "**Normal**" will simply produce a 1 to 1 correlation between table entries and object numbers, placing successive values in successive objects. Data will be interpreted according to the data hint if an octet string is returned, otherwise the ASN encoding will take precedence. If the Get-Next sequence fails to return enough OIDs to fill the 'count' criteria, an error code is set for the device indicating that the table came up short on data.

Method "**Sparse**" is the same as Normal, except missing OIDs in the sequence is anticipated, and the corresponding local objects in the sequence are skipped over if the respective OID is not included in the Get-Next sequence. No error is flagged for the table being short on data.

Method "**Wildcard**" allows wildcard fields in the table OID. The walk does not care about order of OIDs returned by Get-Next as long as they match the OID given after discounting wildcard fields. No attempt is made to sequentially pair OIDs with objects. The next 'count' OIDs that successfully match the OID with wildcards will fill the next 'count' of objects beginning with the starting local object number. The OID sent out in the first Get-Next request will have zero in any wildcard fields, and each successive Get-Next will send out the OID from the response to the previous Get-Next.

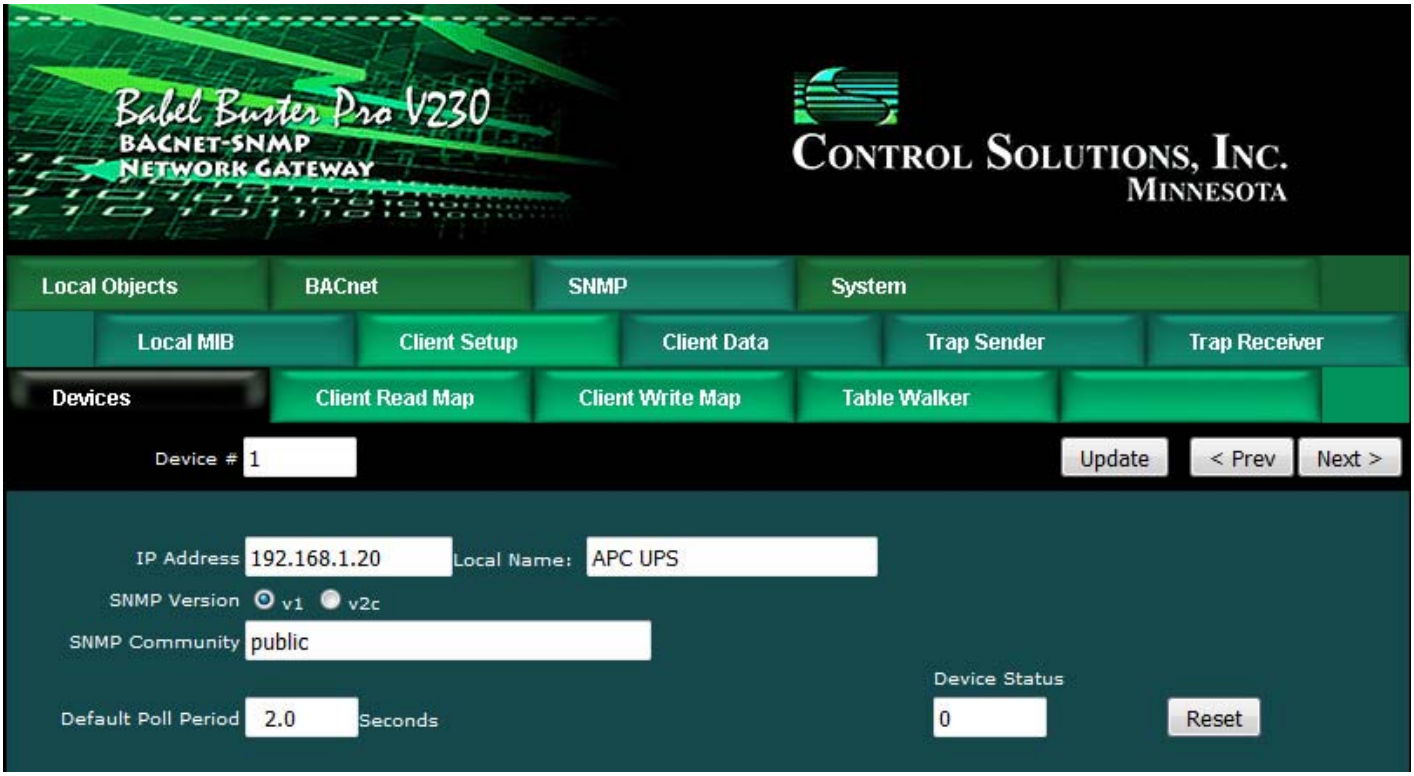
Method "**Index**" will walk the table, but expect to find that values are OIDs. In other words, the table name is an OID, but the contents of the variable at that name will be another OID. The result is that the OID index (last field of OID) from the value will be

used as the offset to calculate which object number is to be affected, and that object will be set to 1 indicating this OID is present in the table. This seemingly odd means of table walking is requiired in order to translate the alarm table from RFC 1628 for UPS systems into indexable BACnet objects that indicate the presence or absence of alarms defined in RFC 1628. (Note that the alarm table in RFC 1628 is "sparse" meaning its table entries are only present if an alarm is present, and the table is empty if there are no alarms. One cannot simply query OIDs to determine presence of an alarm. Alarms are implied by presence of a table entry that only exists while the alarm is present. Furthermore, the alarm table is simply a circular buffer of alarm entries, and the table index means nothing.)

The object set to 1 by the Index method will not be reset to zero by reading anything from the table. The object is set by finding something in the table, and not finding anything in the table will have no effect. Therefore, once set, the object will remain set indefinitely unless the table rule timeout is used. The obect will be reset to zero when the timer expires. The timer is reset every time the table walk writes to this object. But this periodic setting of the object will stop if the previously found table entry is no longer present, and then the timeout will clear or reset the object (e.g., change a Binary object from Active to Inactive).

10.2 Configuring Table Walk Rules

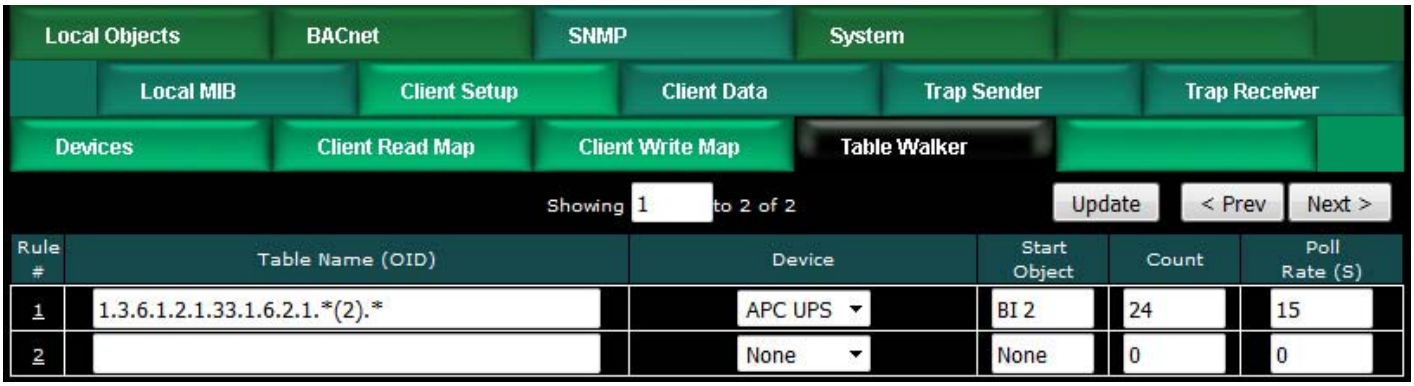
The table walk rules apply to a specific SNMP device whose table is to be walked. The first step in creating a table walk is to enter the device information on the Devices tab. The same set of devices are used for SNMP client read and write rules.



Enter the IP address of the SNMP device, and provide a local name. This local name is

only used to populate the device list on the rules pages. The SNMP version will be either v1 or v2c for this gateway (v3 is not supported here - v3 requires a different model). This determines which version is indicated in the Get-Next requests generated by the table walker. The community should match whatever the SNMP device to be queried is expecting to see. The default poll rate is used whenever the rate in the individual rule is left at zero.

Next, proceed to the Table Walker page. The page that opens initially is a tabular list of multiple rules. You can enter everything necessary for a "Normal" method table walk using just this tabular form. However, if you will be using wildcards in the OID or any method other than Normal, you must click the rule number in the first column and set up the rule using the expanded form. Upon clicking a rule number, the expanded form of the rule is displayed. To return to the tabular list, click on the Table Walker tab again.



The parameters for a table walk rule are as follows. Only the table name, device, starting object, count, and poll rate are entered on the tabular list of rules. To enter the additional parameters or use a method other than "Normal", the expanded view must be used.

Table Name or OID: The walk will begin at the variable name or OID given. It is not necessary to start at the beginning of a table. To walk a section of the table, enter an OID whose last field is the desired first index, minus one. The count will limit the scope of the walk.

The table name OID includes optional wildcard fields when the walk method is Index or Mask. For example, to walk the alarm table of a UPS using RFC 1628, the OID would be 1.3.6.1.2.1.33.1.6.2.1.*(2).*. The *(2) means allow any value in the second to last field but only act on the value when this field is 2. The last asterisk means disregard the last field entirely (which for RFC 1628 increments with each new alarm until reaching some rollover point, then starts back at 1 again).

Method: Select method as defined above in section 10.1.

Device: Select an SNMP device from the list created in the Devices page.

Data Hint: Data hint provides instruction for interpretation of data in the event the value is an Octet String. If the data is an ASN type that is clearly recognizable, the

ASN encoding will take precedence. If the Octet String should be interpreted per RFC 6340 for floating point, select RFC 6340. Otherwise, leave the "Std. ASN" selection as is. An Octet String will be treated as a character string if the destination local object is defined as a character string. Otherwise, an attempt will be made to convert the Octet String as if it were an ASCII representation of some number. If the octet string contains no numeric information, the result will be zero.

Starting Local Object and Count: The "local objects starting at #" is the first BACnet object to be filled by this table walk, and "count" is the number of objects in the sequence that are to be affected. Count is a number of object instances. The count also determines how many OIDs in the table will be read since there is a 1 to 1 correspondence between table entries and local object values saved.

Poll Rate (Repeat...): This sets the rate at which the table should be walked. The amount of time to wait between table walks is entered. Be prudent with poll rate - this will generate a lot of network traffic.

Timeout: The object will be set to zero after this amount of time has expired with no update from the table walk. This only applies when sparse tables are walked (e.g. RFC 1628 alarm table).

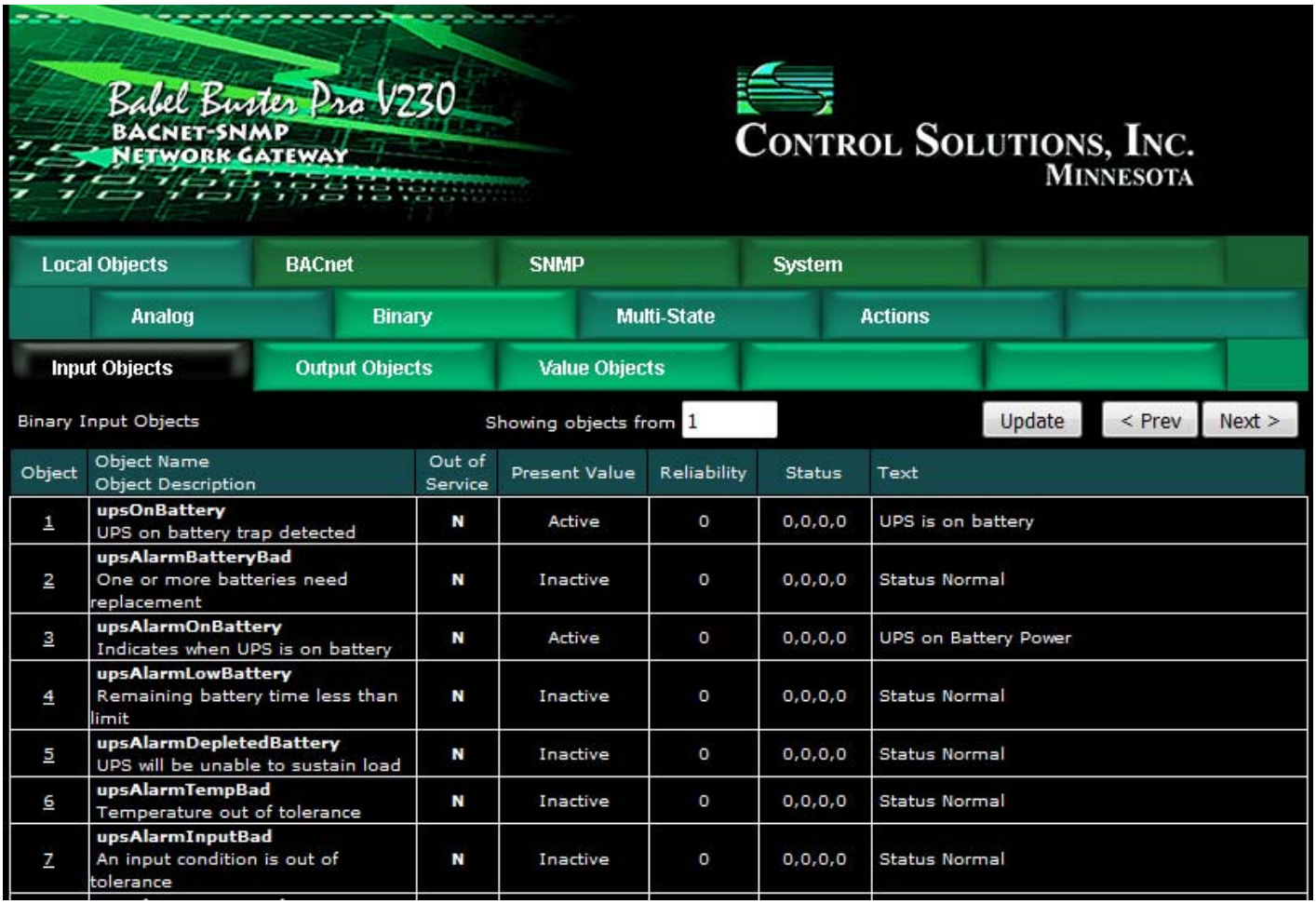
Enable Index Object: You have the option of enabling this rule only when a selected object contains a given value. Any local object may be used as the index object. As the name implies, you could have the same local objects contain different values based on different rules as indexed by the index object.

Click Update to object your changes. Don't forget to save all changes in the configuration file when done. To insert a rule before an existing rule, enter the existing rule number in the "Rule #" window at the top, and click Insert. To delete an existing rule, enter the rule number in the "Rule #" window and click Delete.

Table Walk Example 1:

The table walk rule illustrated above will walk the alarm table in a UPS that

implements RFC 1628. Using the Index method, Binary Input objects illustrated below will be set to"active" when the corresponding alarm condition is present. RFC 1628 defines 24 "well known" alarm conditions. Alarm condition 2 is defined to mean "UPS is on battery". The screen shot below illustrates the UPS being on battery, with no other alarms currently present.



Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Text
1	**upsOnBattery** UPS on battery trap detected	N	Active	0	0,0,0,0	UPS is on battery
2	**upsAlarmBatteryBad** One or more batteries need replacement	N	Inactive	0	0,0,0,0	Status Normal
3	**upsAlarmOnBattery** Indicates when UPS is on battery	N	Active	0	0,0,0,0	UPS on Battery Power
4	**upsAlarmLowBattery** Remaining battery time less than limit	N	Inactive	0	0,0,0,0	Status Normal
5	**upsAlarmDepletedBattery** UPS will be unable to sustain load	N	Inactive	0	0,0,0,0	Status Normal
6	**upsAlarmTempBad** Temperature out of tolerance	N	Inactive	0	0,0,0,0	Status Normal
7	**upsAlarmInputBad** An input condition is out of tolerance	N	Inactive	0	0,0,0,0	Status Normal

The Index method of table walk will set objects corresponding to OIDs found that reference the "well known alarms" defined in RFC 1628. Objects are skipped if no corresponding OID is found in the table. That means the Index walk will set objects but never clear them. To get the objects to reset after being set to indicate an alarm, the timeout feature of the table walk rule must be used.

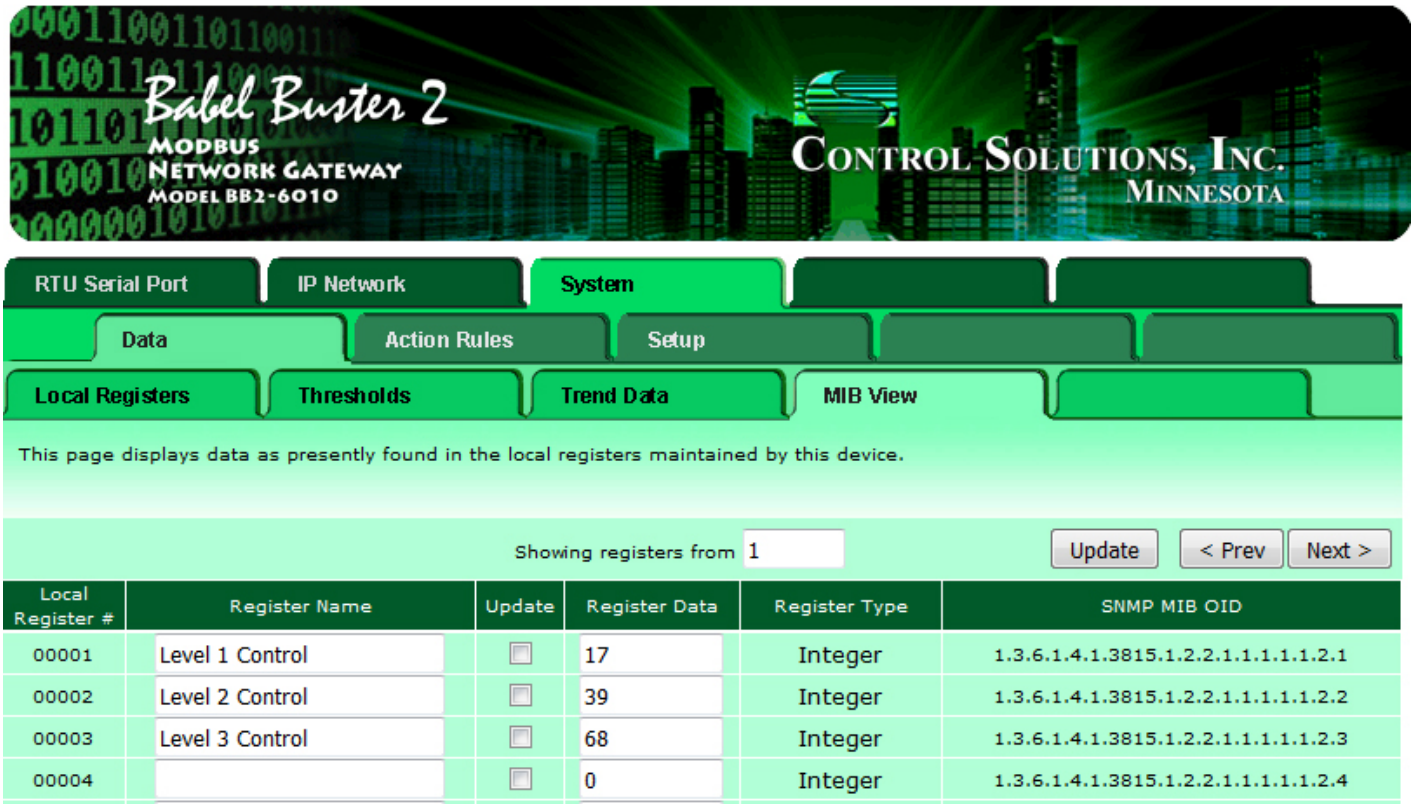
As illustrated above, the timeout is set to 20 seconds. The table walk in this case is set to repeat every 15 seconds. If the alarm is still present, it will be set once again. The timer is reset every time the table walk writes to this object. But this periodic setting of the object will stop if the alarm is no longer present, and then the 20 second timeout will clear or reset the object (change a Binary object from Active to Inactive).

Table Walk Example 2:

The next example is a simple table walk using the Normal method, included here for completeness of discussion. The real power of the Babel Buster Pro is in its ability to

walk the types of tables that require more complex methods. The Normal table walk illustrated here does not need to be a table walk at all. It could just as easily be a set of SNMP client read map rules. The client read rules would periodically Get the values named. Likewise, the Normal table walk periodically Gets the values named. The problem with relying solely upon periodic polling is that in applications like RFC 1628, alarm OIDs only exist while the alarm is active and any attempt to do a direct Get at other times results in an error. Such a problem is solved by the other methods available for table walking in the Babel Buster Pro.

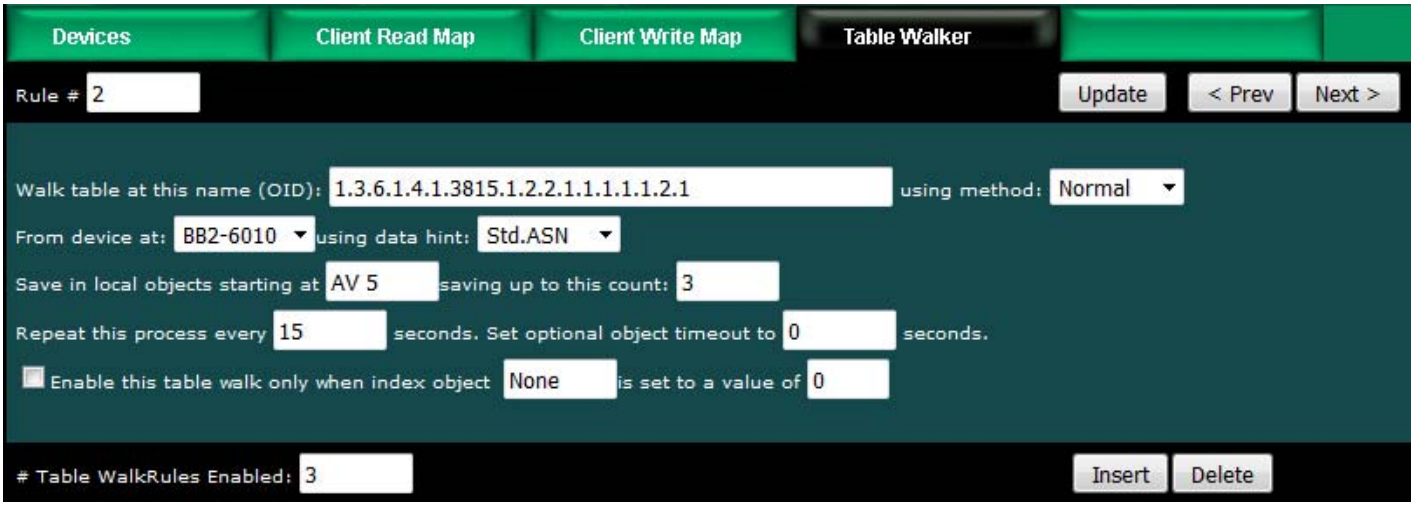
To illustrate the Normal table walk, we will get values from the contiguous full table provided by another gateway, the BB2-6010. A screen shot of its MIB View shows the OIDs we will retreive with our table walk.



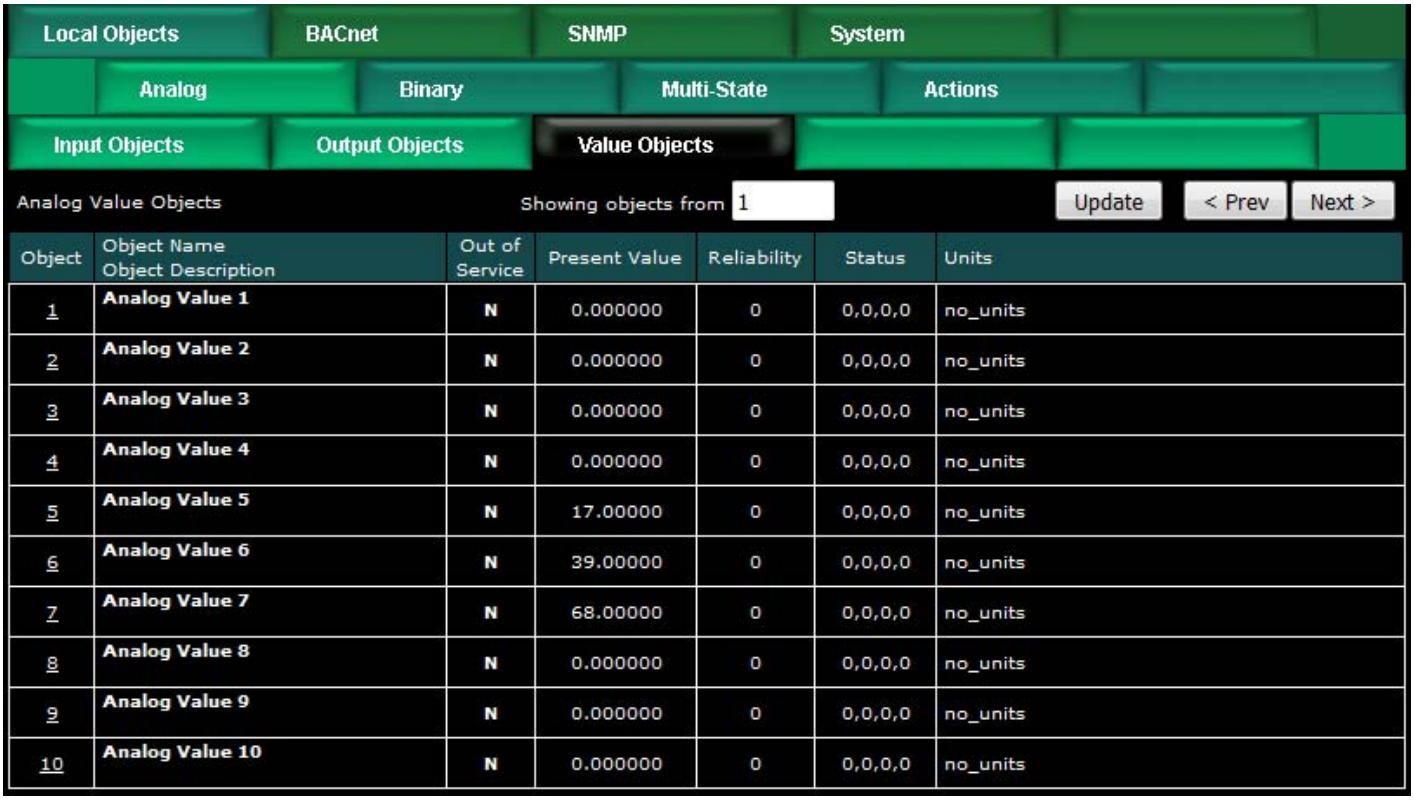
Local Register #	Register Name	Update	Register Data	Register Type	SNMP MIB OID
00001	Level 1 Control	☐	17	Integer	1.3.6.1.4.1.3815.1.2.2.1.1.1.1.1.2.1
00002	Level 2 Control	☐	39	Integer	1.3.6.1.4.1.3815.1.2.2.1.1.1.1.1.2.2
00003	Level 3 Control	☐	68	Integer	1.3.6.1.4.1.3815.1.2.2.1.1.1.1.1.2.3
00004		☐	0	Integer	1.3.6.1.4.1.3815.1.2.2.1.1.1.1.1.2.4

The screen shot of the walk rule list shows the previous example plus this example added as a second rule. Because this is a simple "Normal" table walk, everything needed can be entered on the tabular rule list. The first OID to be read is given as the table name. Select the device, starting local object and count, and poll rate. That is all that is needed. (Note: The device in this example has already been set up over on the devices page, first tab.)

If you were to click the rule number (2) to look at the expanded form of this rule, here is what you would see. No additional entries are required to make this table walk functional.



Given the screen shot of the BB2-6010 MIB View, here is what we expect to see in the Babel Buster Pro when the table walk starts running.

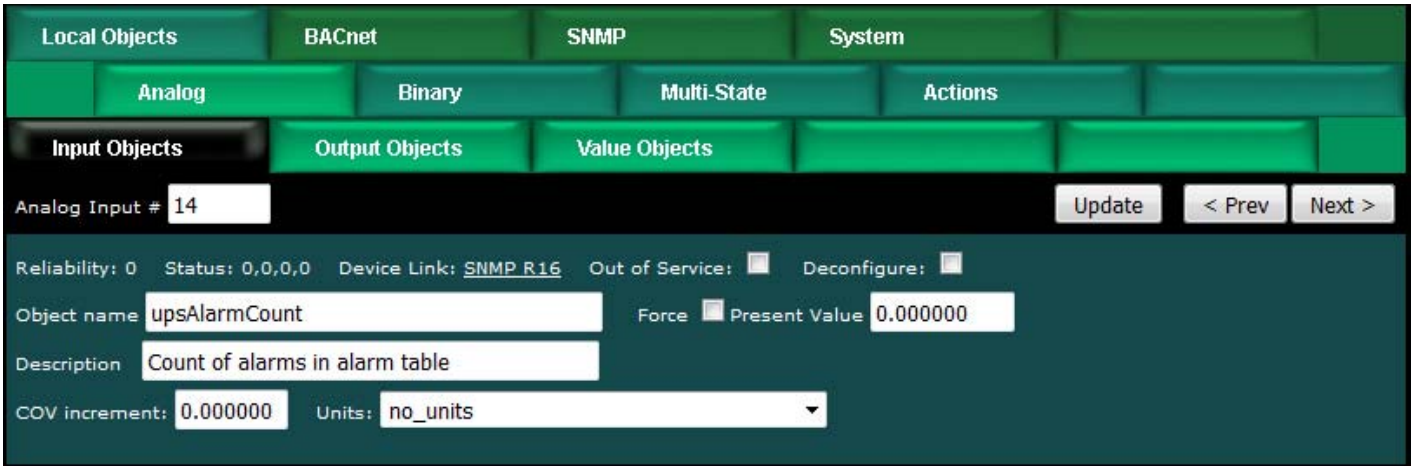


Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Value 1**	N	0.000000	0	0,0,0,0	no_units
2	**Analog Value 2**	N	0.000000	0	0,0,0,0	no_units
3	**Analog Value 3**	N	0.000000	0	0,0,0,0	no_units
4	**Analog Value 4**	N	0.000000	0	0,0,0,0	no_units
5	**Analog Value 5**	N	17.00000	0	0,0,0,0	no_units
6	**Analog Value 6**	N	39.00000	0	0,0,0,0	no_units
7	**Analog Value 7**	N	68.00000	0	0,0,0,0	no_units
8	**Analog Value 8**	N	0.000000	0	0,0,0,0	no_units
9	**Analog Value 9**	N	0.000000	0	0,0,0,0	no_units
10	**Analog Value 10**	N	0.000000	0	0,0,0,0	no_units

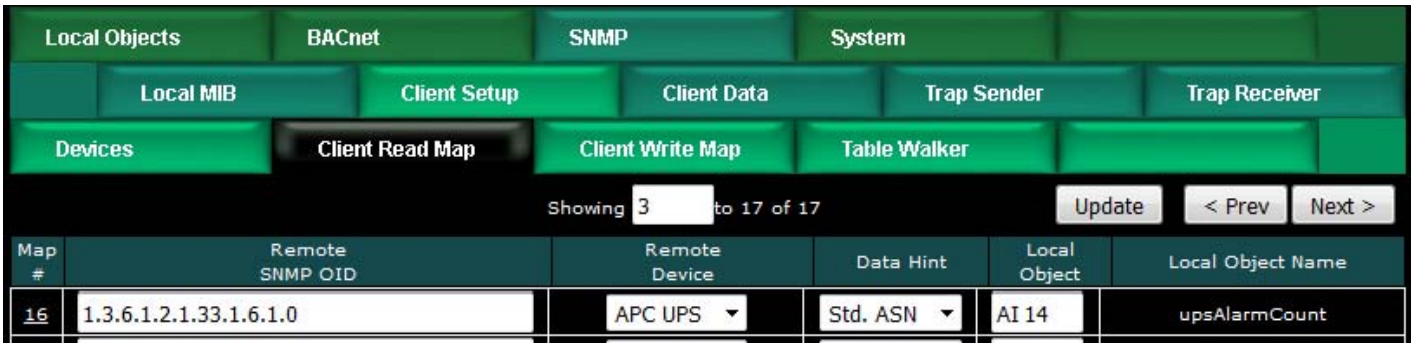
Table Walk Example 3:

The following example elaborates on the RFC 1628 alarm table walk originally illustrated above. We are now going to walk the alarm table conditionally. Namely, we will only walk the table when we know there are alarms in the table. There is a variable in the RFC 1628 MIB that tells us how many alarm entries are currently in the table. This is found at the OID illustrated below, and we have set up an SNMP client read map to retrieve that count, and place it in local object AI 14.

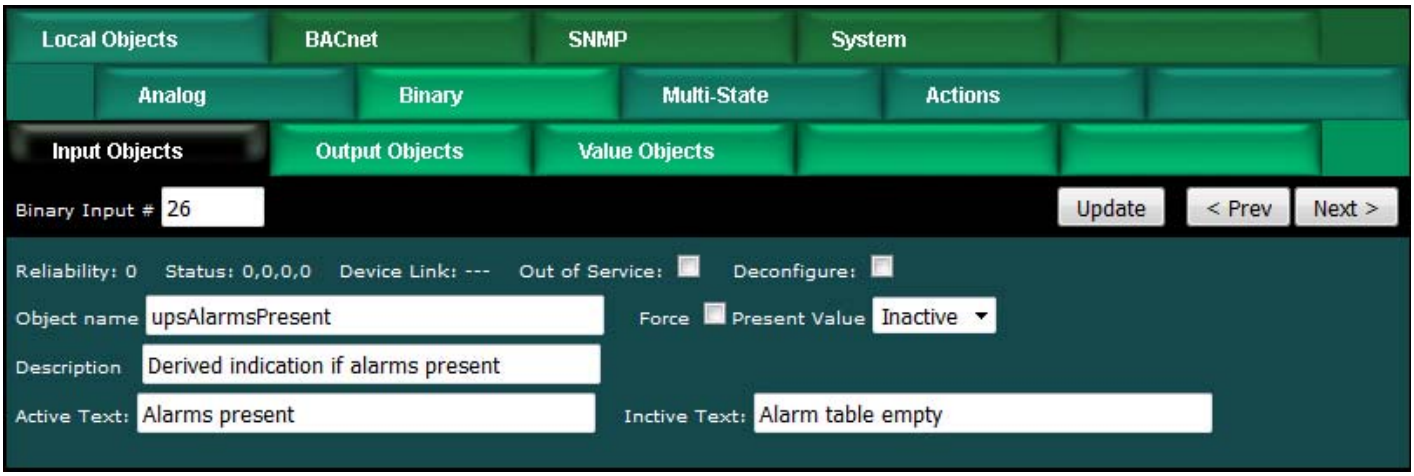
Our definition of AI 14 looks like this:



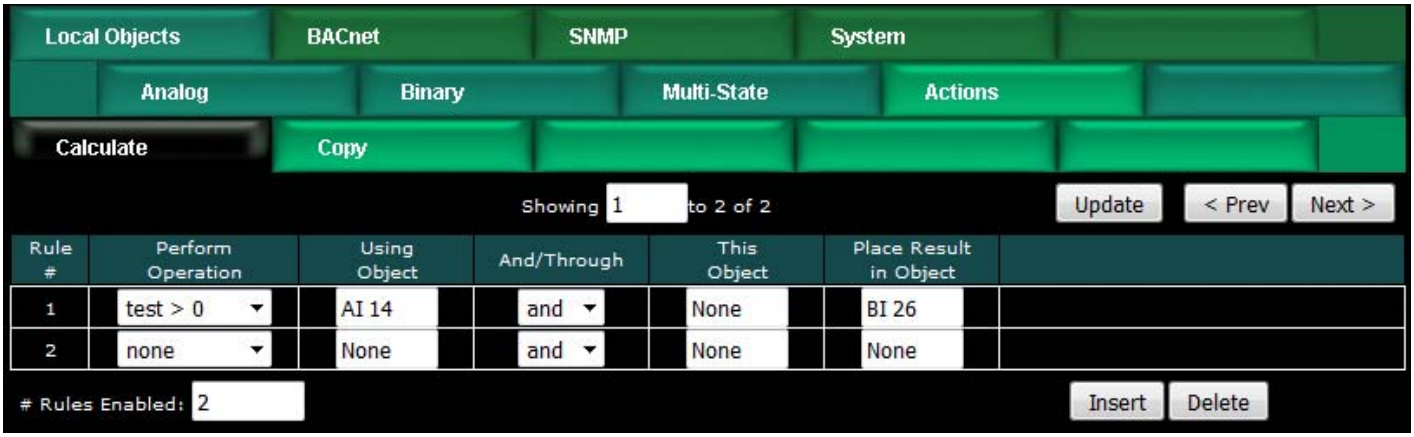
The SNMP Client Read Map looks like this:



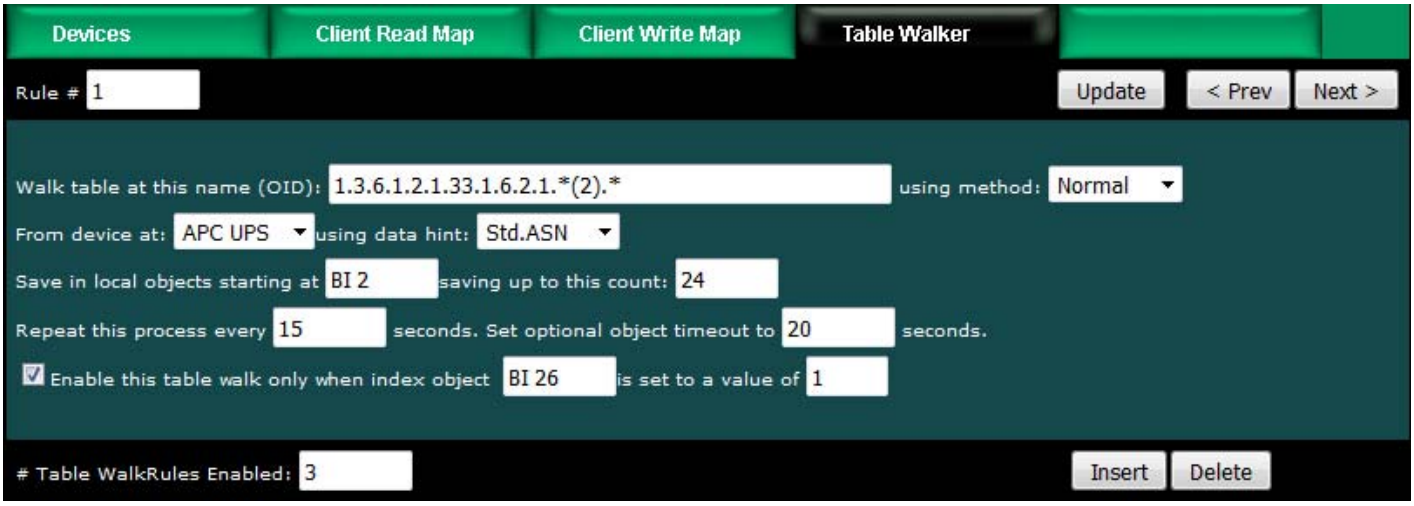
We will create a Binary Input object to hold the result of the rule count test, as follows:



The index object in the table walk rule wants to see a specific number. Since there may be zero to many alarms, we need a way of simply reducing that to an effective "yes or no" to the walk rule. We accomplish this by simply testing to see if the alarm count is greater than zero by using a Calculate rule. The result of the test will be 0 (false) or 1 (true) with the result placed in Binary Input 26.

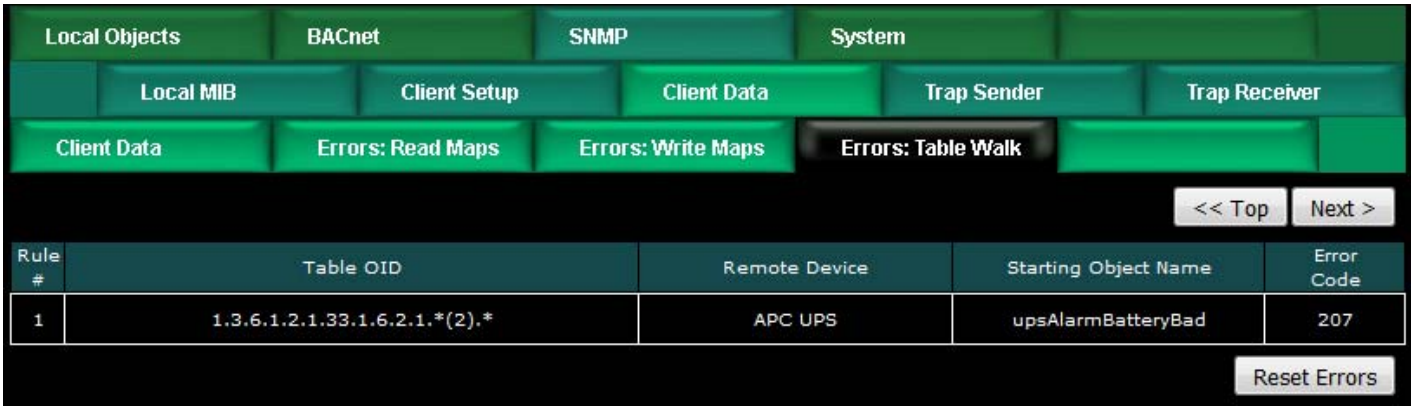


Now that we have a object whose value will be 1 (active) only when one or more alarms are in the RFC 1628 alarm table, we can enable the index object in the table walk rule as illustrated below. We have now reduced network traffic by only walking the alarm table when we know alarms are present.



10.3 Table Walk Errors

If the table walk is unsuccessful, the table OID, device, and an error code are displayed here for the walk rule that ran into trouble.



The following errors are possible. Except for error code 207, all of these error codes apply to any SNMP activity including client Get or Set activity defined by the SNMP client read and write rules. Error 207 applies only to table walking, and means that the Normal method of table walking did not find enough OIDs in the table to satisfy the requested count of local objects to fill.

Standard SNMP error codes returned by remote Agent:
1 = SNMP_ERROR_tooBig
2 = SNMP_ERROR_noSuchName
3 = SNMP_ERROR_badValue
4 = SNMP_ERROR_readOnly
5 = SNMP_ERROR_genErr
6 = SNMP_ERROR_noAccess
7 = SNMP_ERROR_wrongType
8 = SNMP_ERROR_wrongLength
9 = SNMP_ERROR_wrongEncoding
10 = SNMP_ERROR_wrongValue
11 = SNMP_ERROR_noCreation

12 = SNMP_ERROR_inconsistentValue
13 = SNMP_ERROR_resourceUnavailable
14 = SNMP_ERROR_commitFailed
15 = SNMP_ERROR_undoFailed
16 = SNMP_ERROR_authorizationError
17 = SNMP_ERROR_notWritable
18 = SNMP_ERROR_inconsistentName

Client generated errors:
201 = No response from remote Agent (server)
202 = No such name (implicit)
203 = Unable to interpret application data
204 = Reply does not match request
205 = There is a problem with the rule configuration
206 = Unable to build or parse SNMP PDU
207 = Table walk came up short of expected count

If the error code indicates no response, the device status (SNMP Client Setup Devices page) will provide an additional indication of a connection related error.

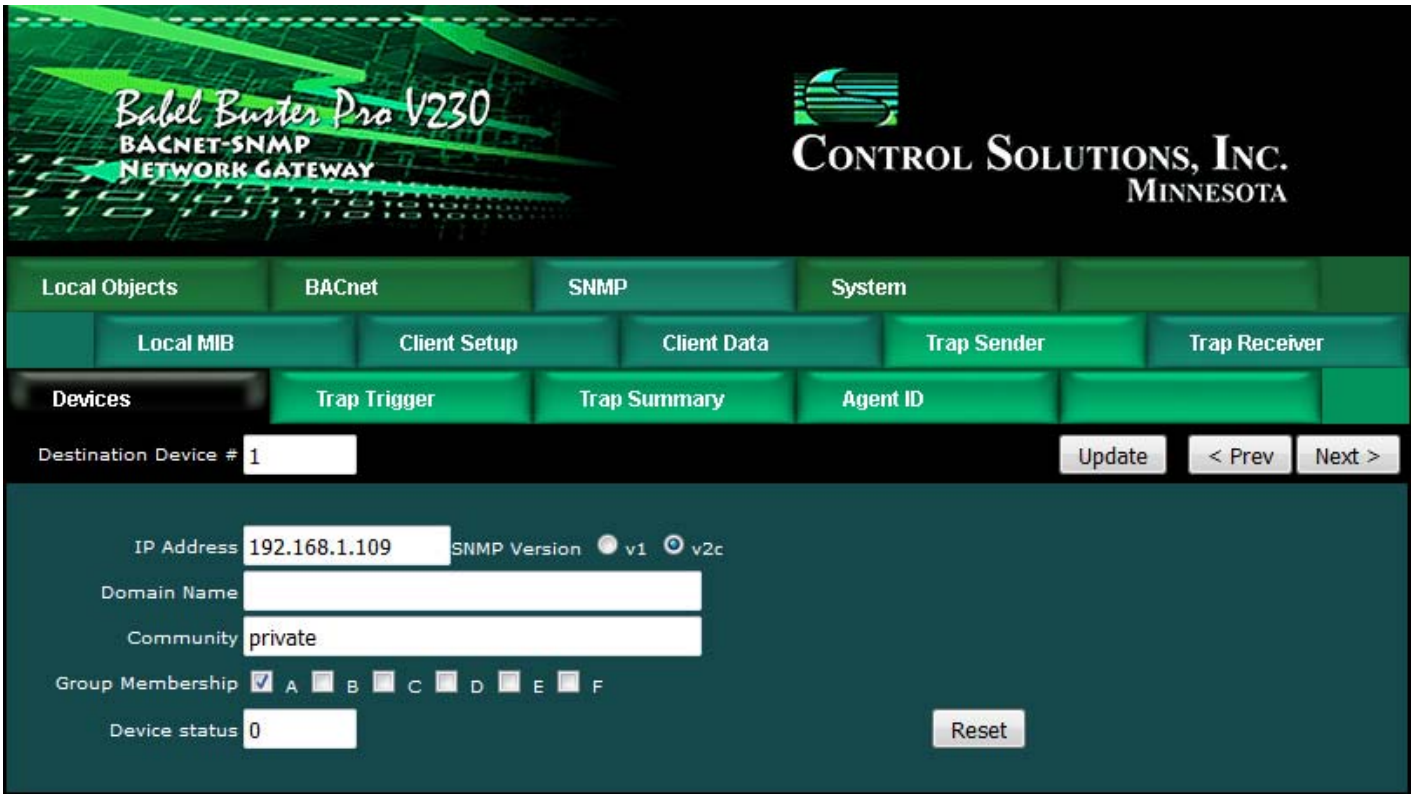
11. Configuring SNMP Trap Sender

11.1 SNMP Trap Destinations

The first step in sending traps is to tell Babel Buster Pro where to send them. This is done on the Devices page of the Trap Sender (which is not the same list of devices used by the Trap Receiver or SNMP Client).

For each destination that traps should be sent to, enter the IP address, select v1 or v2, and provide the community string that should be used when sending traps to that device.

Be sure to select one or more groups for membership for this device. The device must be a member of the group before traps assigned to that group will be sent to this device. This membership method allows different traps to be sent to different devices if desired.



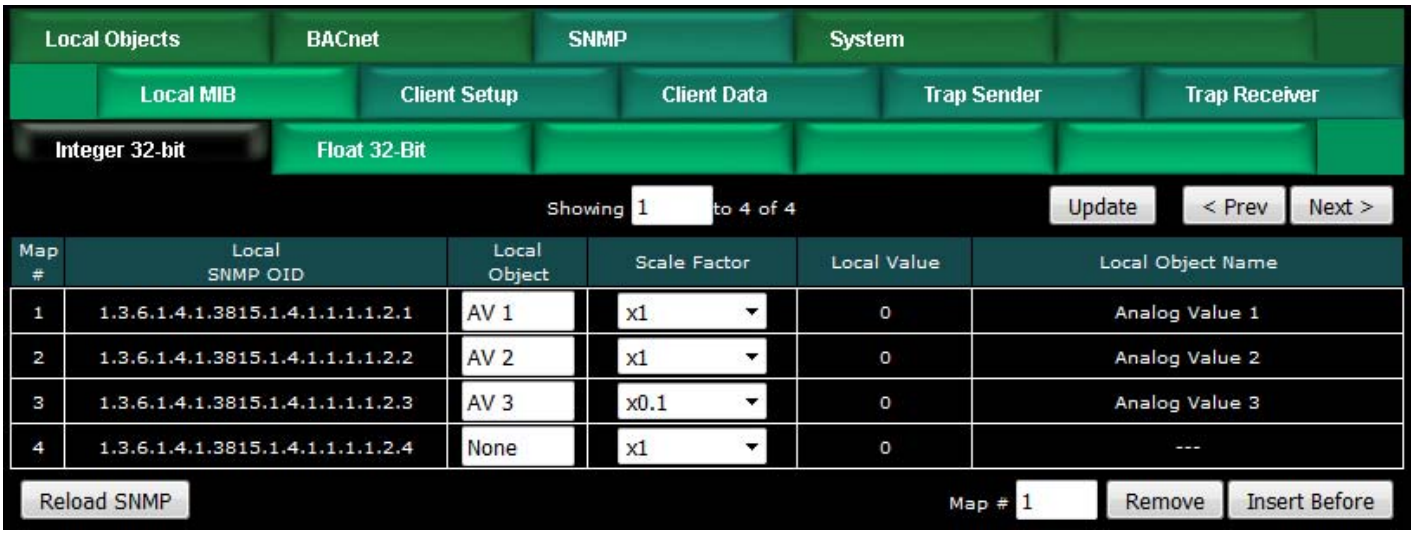
The structure of a v2 trap is noticeably different than a v1 trap, but the only difference you need to take note of here is simply selecting one or the other for the given IP

address.

11.2 SNMP Trap Triggers

Create a Trap Trigger for each trap that should be sent.

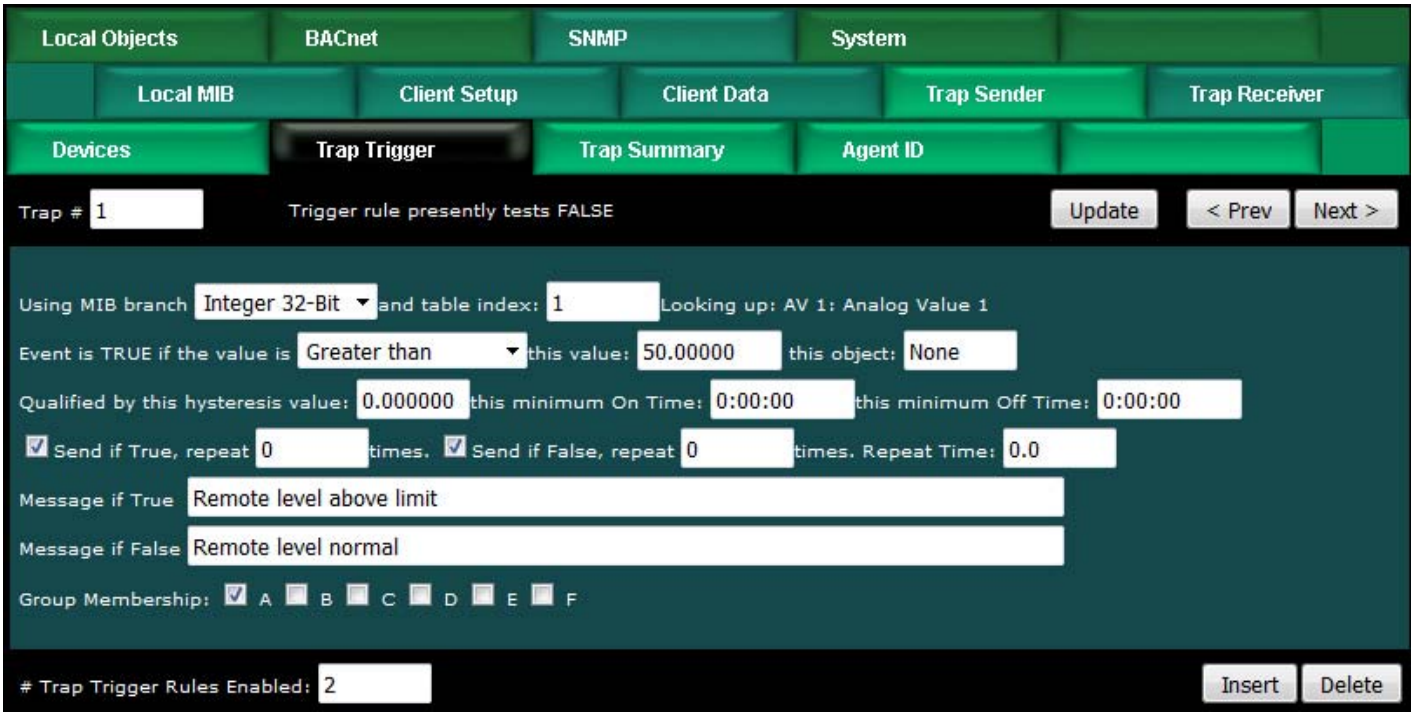
A trap must reference a variable in the local MIB. Therefore, before you can create a trap trigger, you must assign the local object of interest to a spot in the local MIB. (See "Configuring Gateway as an SNMP Server".) Once the local object has a home in the MIB, select the branch and MIB index in the trap trigger rule. Do not enter local object number here. Enter the MIB table index from your local MIB configuration pages.



We are now going to create a trap that will be triggered by Analog Value 1.

Once the variable of interest is selected, define the threshold. This will be a test such as "greater than" or "less than", etc. You can provide a fixed value for the threshold, or you may reference a local object that will hold a threshold that may change from time to time. The trap trigger will be "true" when the MIB variable meets the criteria given.

You have the option of specifying a hysteresis value. If non-zero, this means that the referenced MIB variable must fall away from the threshold by this amount before a "true" result will return to "false". You also have the option of specifying a minimum On and Off time. The "On" time means the rule must test true for this amount of time before the status will actually be set true and the trap will actually be sent. The "Off" time means the rule must test false for this amount of time before its status will actually be returned to false.



Trap number simply tells you where you're at on the list of the local SNMP Agent's traps. Click "Next" and "Prev" to scroll through the list. To advance directly to a specific trap trigger, enter the desired number in the "Showing" box, then click Update.

Traps can only be generated for variables that are delcared in the Local MIB. Select the MIB branch and table index to be tested. The local object referenced in the MIB will be tested to see whether this trap is true or false. A variable not found in the Local MIB cannot generate traps.

The object found in the MIB branch, at the table index given, is displayed for reference as "Looking up...". To change this object, go to that MIB branch and table index, and change the MIB definition, or select a different branch and/or table index.

Select a comparison or test. Enter the value against which the MIB variable should be tested, or alternately, enter a local object number that will contain the value against which the MIB variable should be tested. If object is zero, then the fixed value will be used. If a non-zero object number is given, then the fixed value is disregarded.

Quaifications are optional, and enabled only when values are nonzero. How hysteresis is applied depends on the comparison. For a test that becomes true if greater than, the test will not return to false until the local object is less than the test value by a margin of at least this hysteresis value. If a test becomes true if less than, it will not return to false until the local object is greater than the test value by a margin of at least this hysteresis value.

Special test types: "Deviates from" will test against the value given, and use Hysteresis as the margin of deviation. This is effectively a "greater than or less than" test for deviation from a setpoint. "Changes by" will become true each time the given

variable changes by the value given, and Hysteresis has no effect on this test. If "Changes by" references a value of zero, then this becomes a special test whereby the event is true any time something in the system updates the variable (e.g. Script Basic writes a new value to this variable). A "changes by zero" should not be used when the variable is continuously read from a slave device since this will result in continuous traps. If "changes by zero" is applied to a variable that is written by Script Basic only upon certain conditions within the program, then this gives the Basic program a means of generating traps on demand.

On time and off time, if specified, determine how long the condition must be true (on time) or false (off time) before the true or false response is actually taken. Times are given in HH:MM:SS format (hours, minutes, seconds).

Check "Send if True" and/or "Send if False" to indicate when traps should be sent. Enter a repeat count if the trap should be repeated. If repeat count is zero, the trap will be sent one time. If repeat count is 1, the trap will be sent 2 times, and so on. The interval between traps will be the Repeat Time in seconds. Enter -1 for trap repeat count if the trap should simply repeat indefinitely at the Repeat Time interval. A repeat count of -1 for "Send if True" is acceptable. A repeat count of -1 for "Send if False" will be treated as no repeat since indefinite repeating of non-true events is ill-advised.

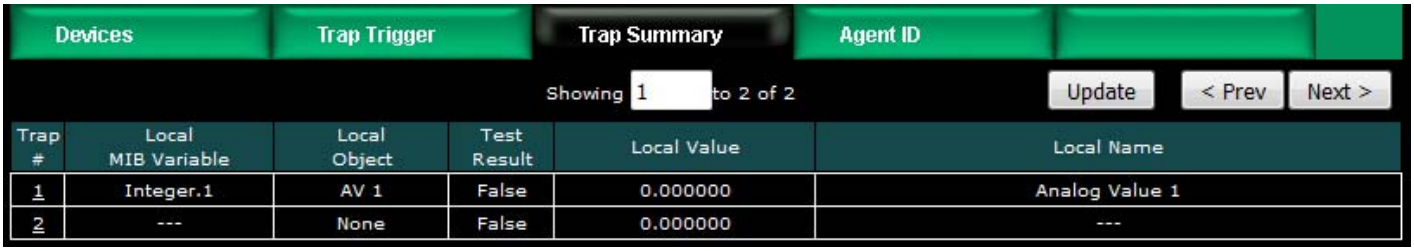
One of the varbinds in the trap message is an arbitrary ASCII character string, sent as an ASN Octet string. The "True" message will be sent when the trap event is true, and the "False" message will be sent when the trap event is false.

To select which destinations from the Trap Sender Devices page that this trap will be sent to, mark the group memberships that will match at least one of the group memberships for that device.

To delete the rule shown, click Delete. To insert a new rule before the rule displayed, click Insert. To add a rule to the end of the list, click Next when at the end of the list, enter new rule, and click Update. The number of rules enabled simply limits the scope of rule review so that you do not have to review a lot of unused rules. If the displayed rules are used up and you need more, increase the enabled number.

11.3 SNMP Trap Summary

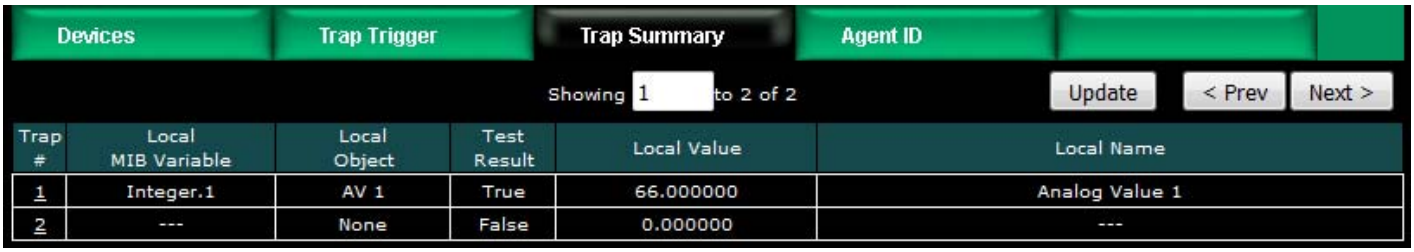
The trap summary provides a page where you may see the present status of all of your trap trigger rules. Here we see the normal "false" status of the above trigger.



Trap #	Local MIB Variable	Local Object	Test Result	Local Value	Local Name
1	Integer.1	AV 1	False	0.000000	Analog Value 1
2	---	None	False	0.000000	---

The thresohld has been exceeded in the above rule, and therefore a trap has just been

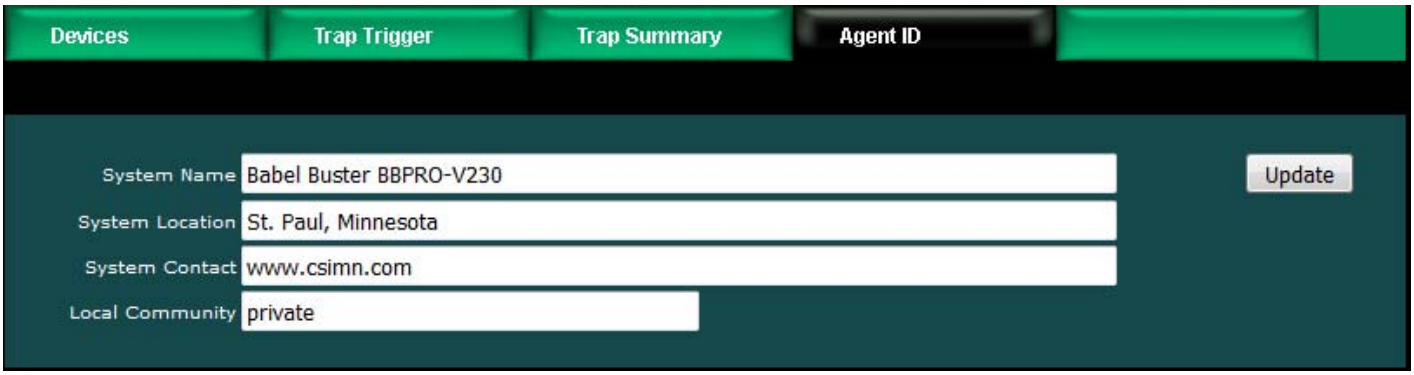
sent in the illustration below.



Trap #	Local MIB Variable	Local Object	Test Result	Local Value	Local Name
1	Integer.1	AV 1	True	66.000000	Analog Value 1
2	---	None	False	0.000000	---

11.4 SNMP Identity

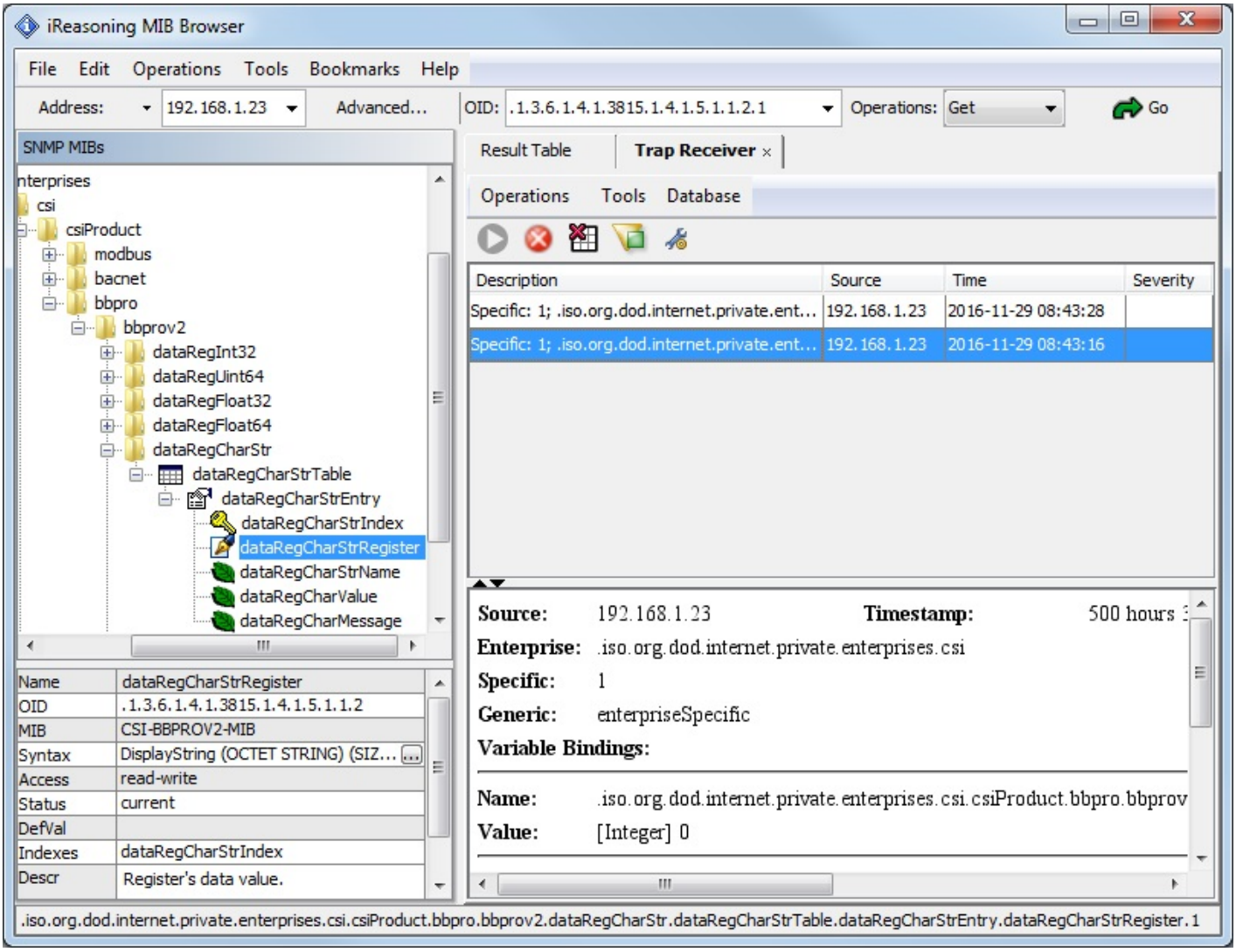
Sending traps requires setting up the local MIB. Part of setting up a local MIB includes setting the identity of the local device. This is done on the Agent ID page. Local community is also accessible on the Network setup page, but the remaining parameters for local identity are only accessible here.



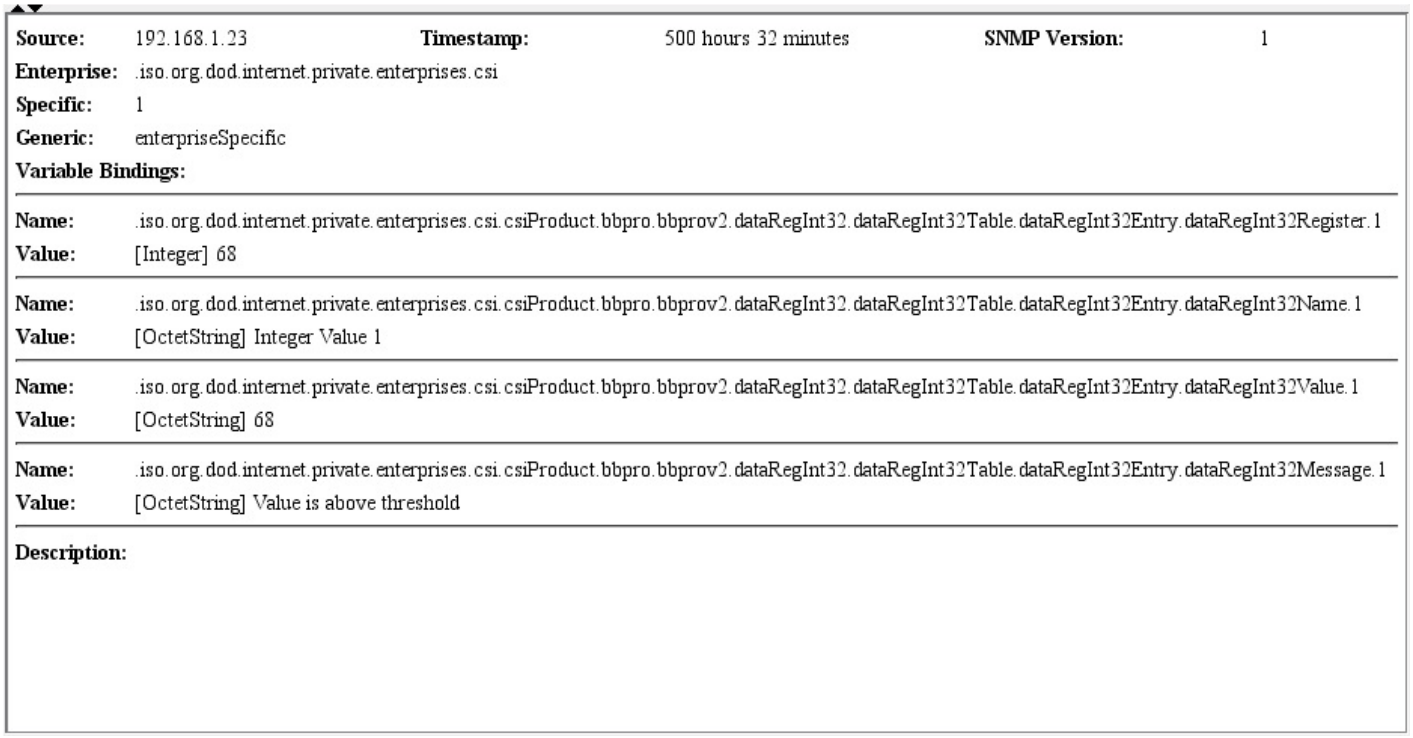
11.5 Testing the SNMP Trap Sender

A variety of tools are available for browsing an SNMP MIB and receiving SNMP Traps. The tool used in the following examples is the iReasoning MIB Browser. Refer to the Tools section under Support at csimn.com for more information about SNMP tools.

The iReasoning MIB Browser allows you to browse the MIB, do table walks, etc., but of primary interest here, also allows you to test traps. When you open the iReasoning browser, under Tools, select Trap Receiver. The screen illustrated below shows the trap viewer after receiving a couple of v1 traps from our Babel Buster Pro.



If you expand or scroll the window in the lower right, you will see the full view of the trap as illustrated below.



Source: 192.168.1.23 **Timestamp:** 500 hours 32 minutes **SNMP Version:** 1
Enterprise: .iso.org.dod.internet.private.enterprises.csi
Specific: 1
Generic: enterpriseSpecific
Variable Bindings:

Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Register.1
Value: [Integer] 68

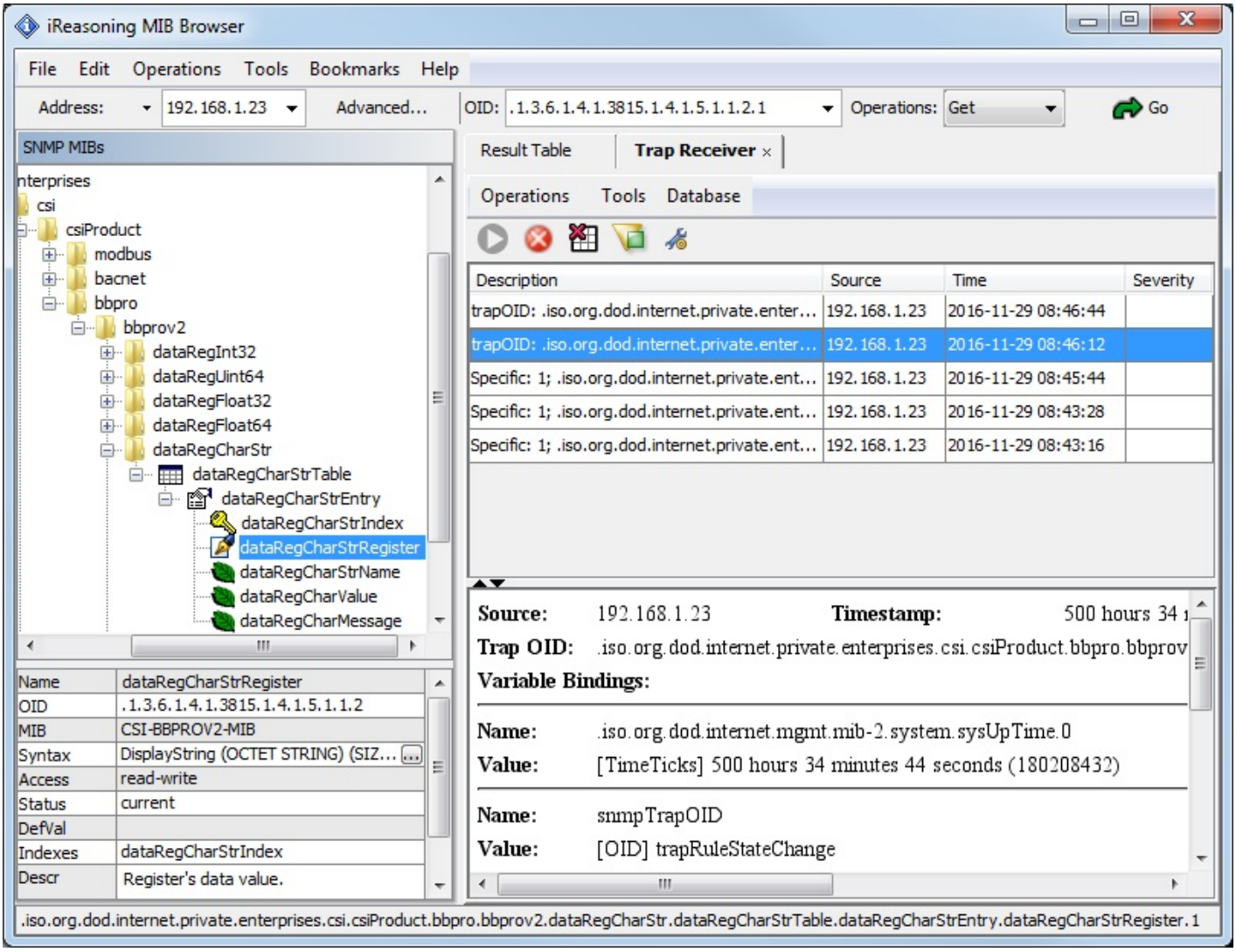
Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Name.1
Value: [OctetString] Integer Value 1

Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Value.1
Value: [OctetString] 68

Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Message.1
Value: [OctetString] Value is above threshold

Description:

The trap viewer, after reconfiguring the Babel Buster Pro to send v2c traps instead, is illustrated below.



The similar but different content of the v2c version of the same trap (with slightly different value) is illustrated below.

Source: 192.168.1.23 **Timestamp:** 500 hours 34 minutes 44 seconds **SNMP Version:** 2

Trap OID: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.bbprov2Traps.trapRuleStateChange

Variable Bindings:

Name: .iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0
Value: [TimeTicks] 500 hours 34 minutes 44 seconds (180208432)

Name: snmpTrapOID
Value: [OID] trapRuleStateChange

Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Register.1
Value: [Integer] 81

Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Name.1
Value: [OctetString] Integer Value 1

Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Value.1
Value: [OctetString] 81

Name: .iso.org.dod.internet.private.enterprises.csi.csiProduct.bbpro.bbprov2.dataRegInt32.dataRegInt32Table.dataRegInt32Entry.dataRegInt32Message.1
Value: [OctetString] Value is above threshold

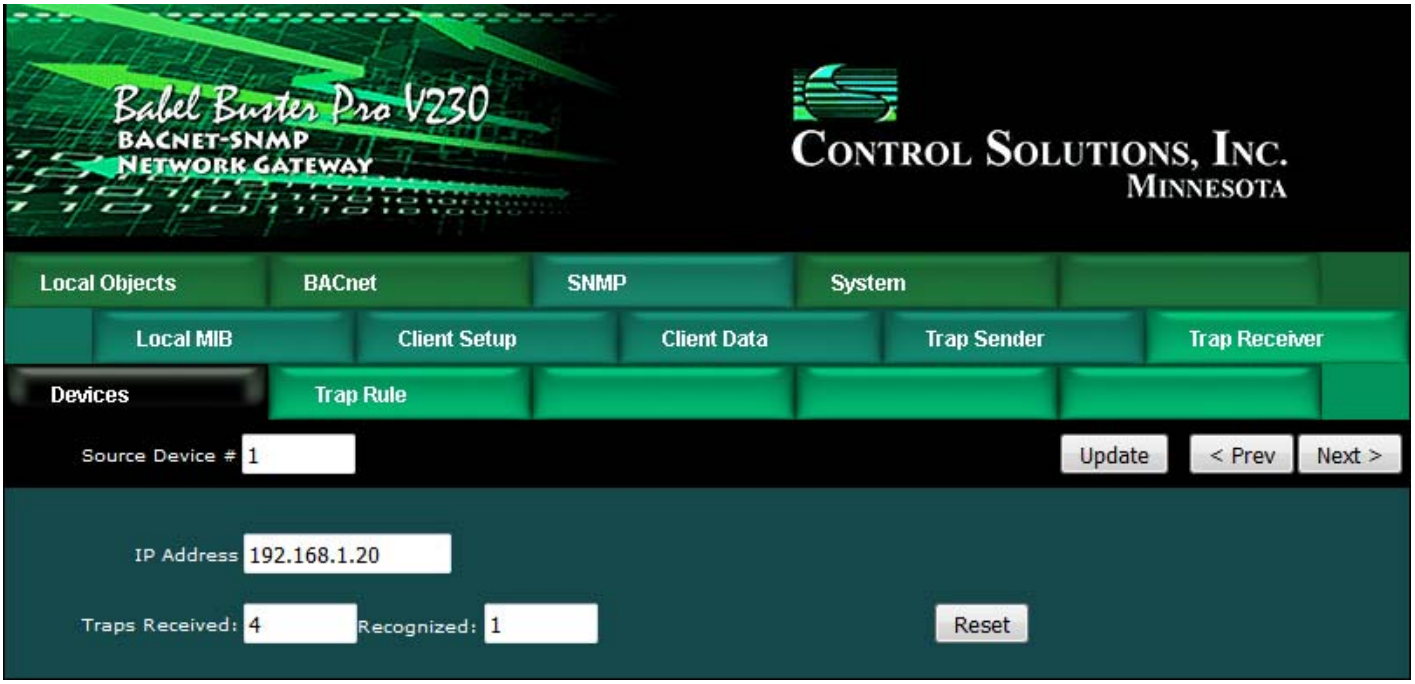
Description:

12. Configuring SNMP Trap Receiver

12.1 Trap Receive Devices

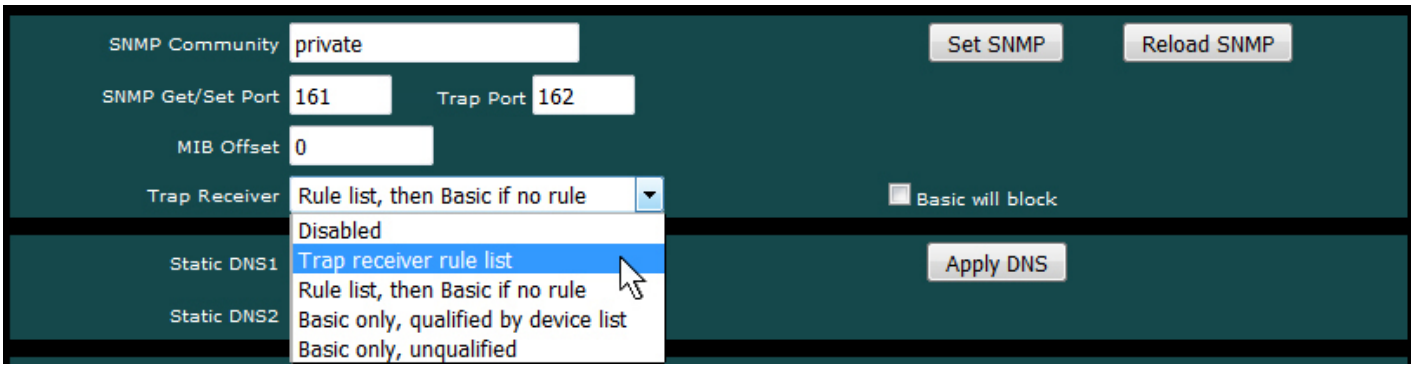
The Devices page under Trap Receiver provides IP addresses of devices that will be sending traps to this Babel Buster. Traps will not be recognized by the trap rules if the IP address is not recognized in the list provided here.

The number of traps received from this device is counted. The counts reflect the number of traps received from this IP address since the last reset (or power up). Click Reset to reset the count. Both total traps and traps recognized by rules in the table are tabulated. Traps processed by Script Basic are not counted in the "recognized" tabulation since this counter has no way of knowing what Script Basic might have done with the trap message.

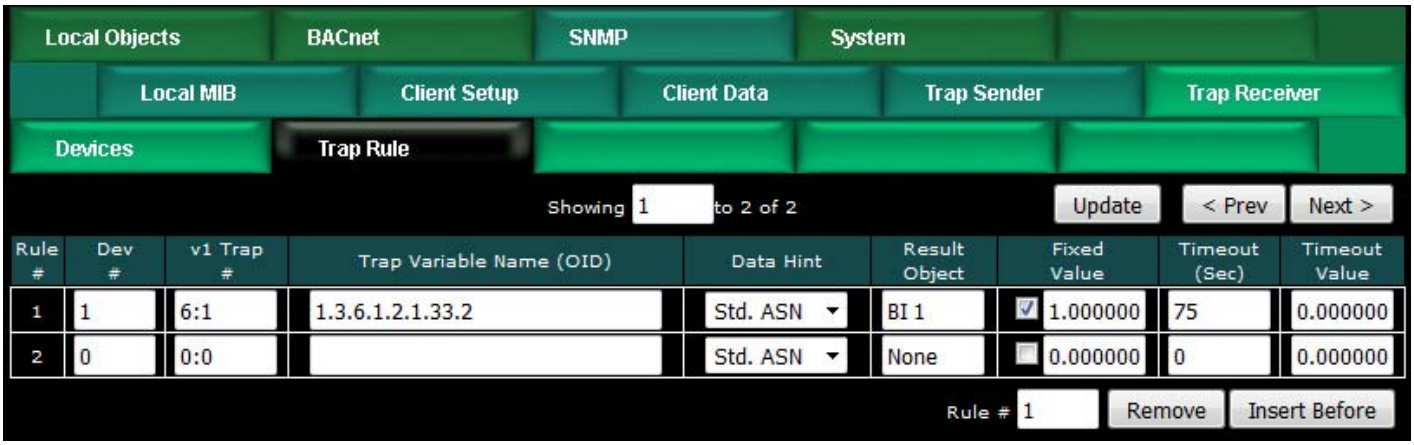


In addition to entering the IP addresses of devices that will be sending traps to this Babel Buster, you need to enable trap receiving on the Network setup page. If disabled, nothing entered on the Trap Receiver pages will be utilized. If only trap rules will be used, select "Trap receiver rule list". If Script Basic will be used, select the applicable option. Rules entered on the Trap Rule page will be ignored if either of the "Basic only" options are selected.

The "Basic only, unqualified" option is the one instance where you do not need to enter any IP addresses on the Devices page. In this case, there is no filtering by IP address and all traps regardless of source are then sent to Script Basic. It then becomes your program's job to figure out where they came from and what if anything to do about the trap.



12.2 Trap Receive Rules



Rules for recognizing received traps are defined on this page. You have the option, on the System Setup Network page, to declare that trap receiving is disabled, or if enabled, processed by these rules and/or Script Basic programming.

Rule number simply tells you where you're at on the list of Trap receiver rules. Click "next" and "prev" to scroll through the list. To advance directly to a specific rule, enter the desired number in the "Showing" box, then click Update.

Device number specifies which device on the SNMP Client Devices list this trap will be received from. If traps are received from devices not listed on the Devices page, they will be discarded unless Script Basic has been selected to receive unqualified traps (see Systen Setup Network page). Setting device number to zero will result in the rule being removed from the list the next time the configuration file is saved.

If the trap received is a v1 (version 1) trap, you can specify the trap number expected as "a:b" where "a" is the generic trap number (for example, 6 for private enterprise) and "b" is the specific trap number as found in the device's MIB file. Leave this set to 0:0 if no v1 trap number is expected or should be disregarded. (Note: SNMP v2c traps

will not have this number, and if trap type is detected as v2c, then this entry will be disregarded.)

The variable name that is expected to be found in the trap should be given as an OID for v2 traps, and optionally for v1 traps. If this OID is not found in the trap message, no further action is taken for this trap rule. If no OID is given, especially for v1 traps, then the trap number alone is enough to qualify this trap, and you should use the Fixed Value as the value to be placed into the result object.

Special case use of Trap Name OID: If the trap is v1 and trap numbers are given, then this OID, if provided, will be used to check the enterprise OID of the trap message received. If the OID is provided in the rule, then it must match the v1 trap enterprise OID before the rule will be considered successful.

Data hint helps the data parser figure out what the varible is. If ASN encoding is recognized as other than an octet string, the hint will be disregarded. If an octet string is found, then the parser needs to know if it should be treated as RFC 6340 floating point. If no hint is given (standard ASN), then the octet string will be treated as an ASCII character string, in which case ASCII to numeric conversion will be attempted automatically if the result object is numeric (string will simply be copied if result object is a character string type object). Note that standard, well-known ASN types are recognized as well as the NetSnmp ASN type for opaque float.

If a matching variable name is found, and the data parser succeeds in parsing the data value, then it will be placed into the object identified as "Result Object". However, if Fixed Value is checked, then whatever value is entered next to the check box will be placed into the result object just because the variable name was found. In the case of v1 traps, if no OID is given, then the trap number alone is enough to cause the fixed value to be placed into the result object.

Timeout is optional, and will be in effect if a non-zero time in seconds is provided. If given, then the timeout value will be placed into the result object after this amount of time has elapsed since receiving the trap. If another of the same trap is received, the timer will be reset. Only after this time has expired without receiving the trap, but after receiving it at least once, the timeout value is placed into the result object.

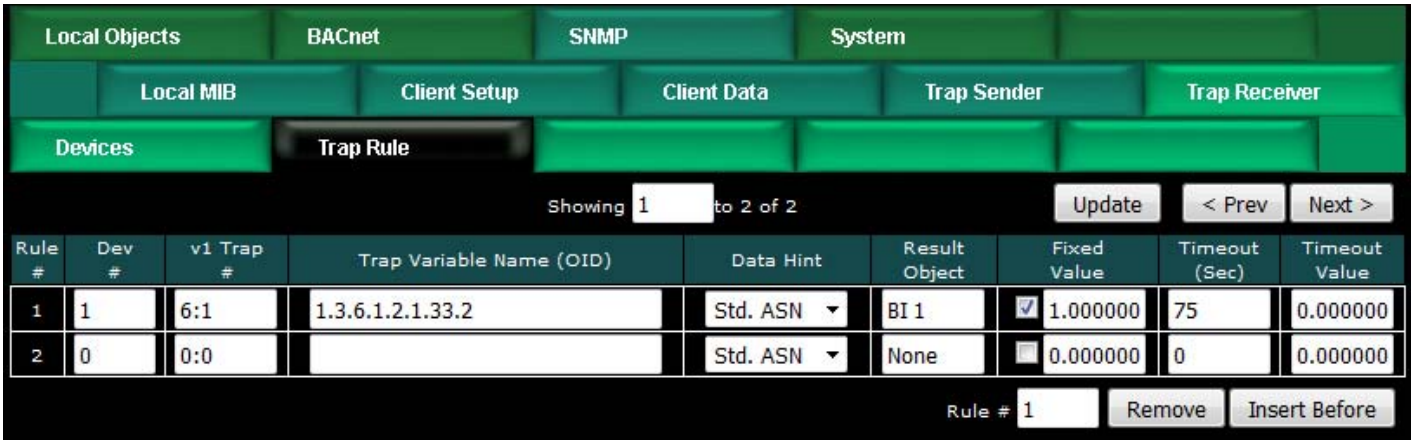
Click Update when all entries have been made. After all configuration has been entered, be sure to go to the Config File page and click Save so that changes are retained through power cycles.

To insert or delete trap rules in the table, enter the trap number and click the appropriate button. To add a new rule to the end of the list when already at the end, simply click the Next button at the top.

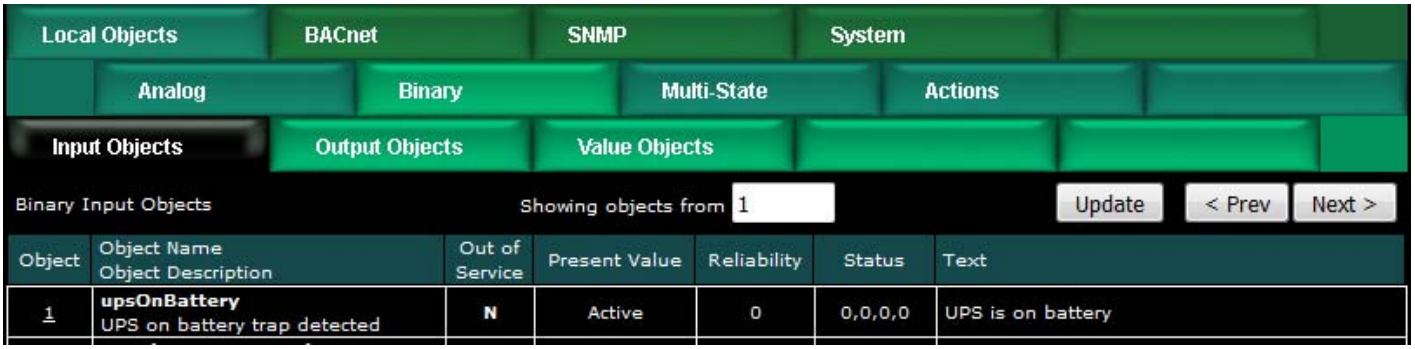
12.3 Trap Receive Examples for SNMPv1

The trap receive rules illustrated in the following example will set Binary Input 1 to

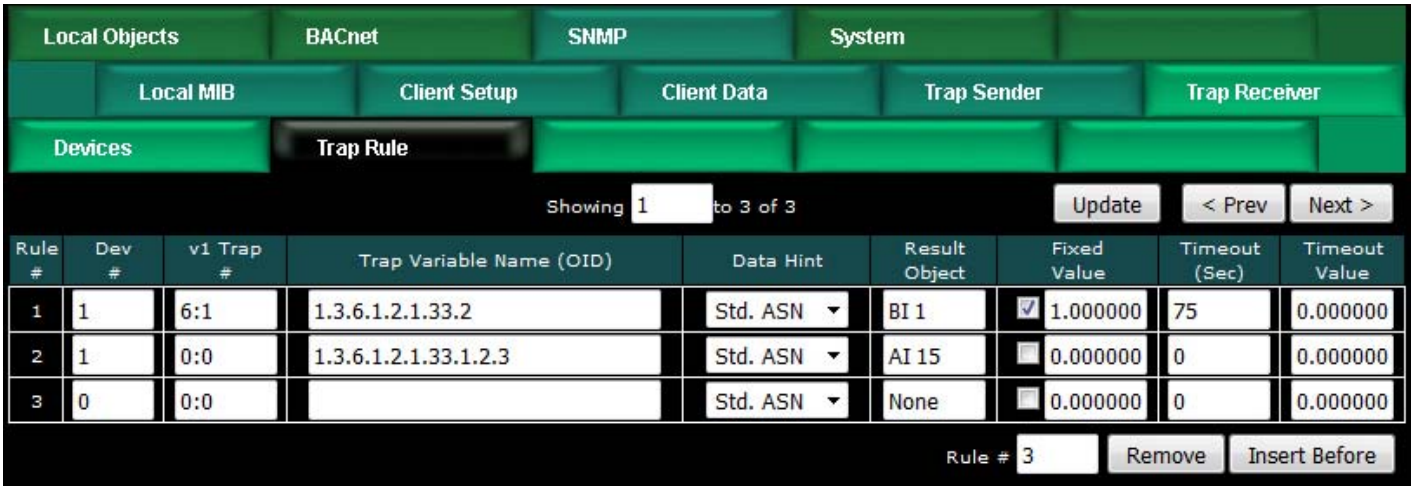
Active when the "on battery" trap is received from RFC 1628. This trap (per RFC 1628) will repeat every 60 seconds until power is restored. Therefore, we have set a timeout for just longer than 60 seconds so that our Binary Input object returns to Inactive after the alarm traps stop ocurring.



Upon receiving the trap, our Binary Input will become Active as illustrated here.

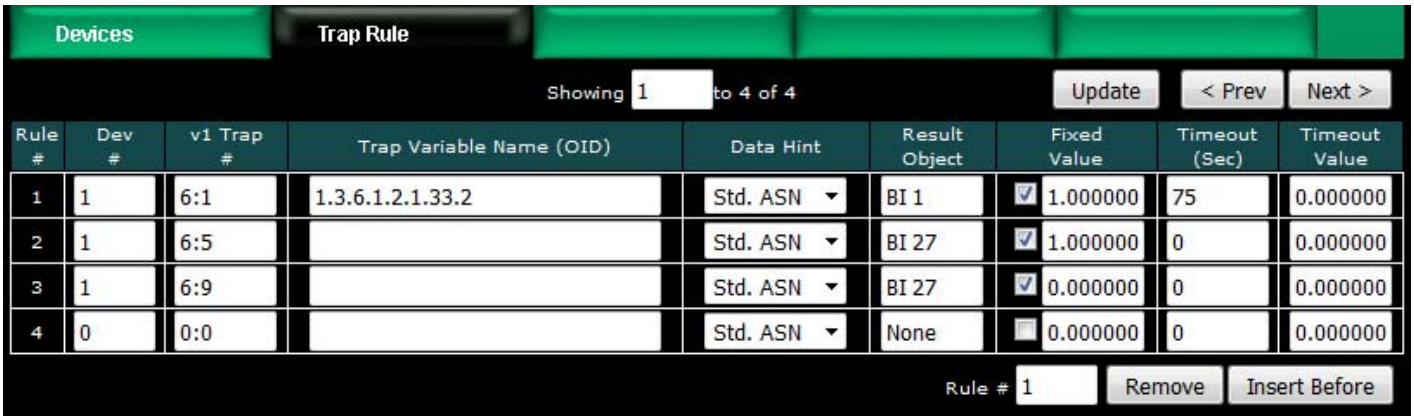


In addition to the trap number itself, the trap includes some varbinds (data pairs consisting of an OID and the data value identified by that OID). Rule #2 in the next example will capture the remaining minutes of battery power sent by the UPS with each of those trap messages sent every 60 seconds. Although this is still a v1 trap, it is important to leave the trap number set to 0:0 so that the varbind is processed by the rule rather than the trap number.

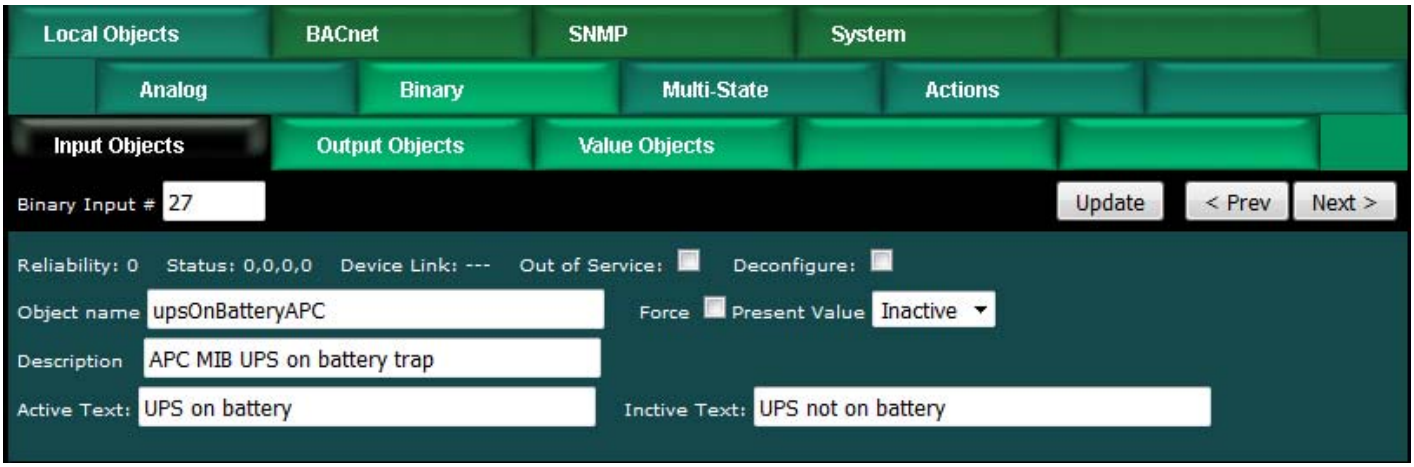


Now each trap will set Binary Input 1 to Active, and also place the minutes remaining in Analog Input 15.

An APC UPS includes both RFC 1628 and their own private enterprise MIB which starts at 1.3.6.1.4.1.318. The APC MIB sends enterprise specific trap number 5 when the UPS switches to battery power, and trap number 9 when the UPS switches off battery power (utility power restored). In the example below, rules 2 and 3 will set Binary Input 27 to Active when the "on battery" trap is received, and clear Binary Input 27 to Inactive when the "not on battery" trap is received.

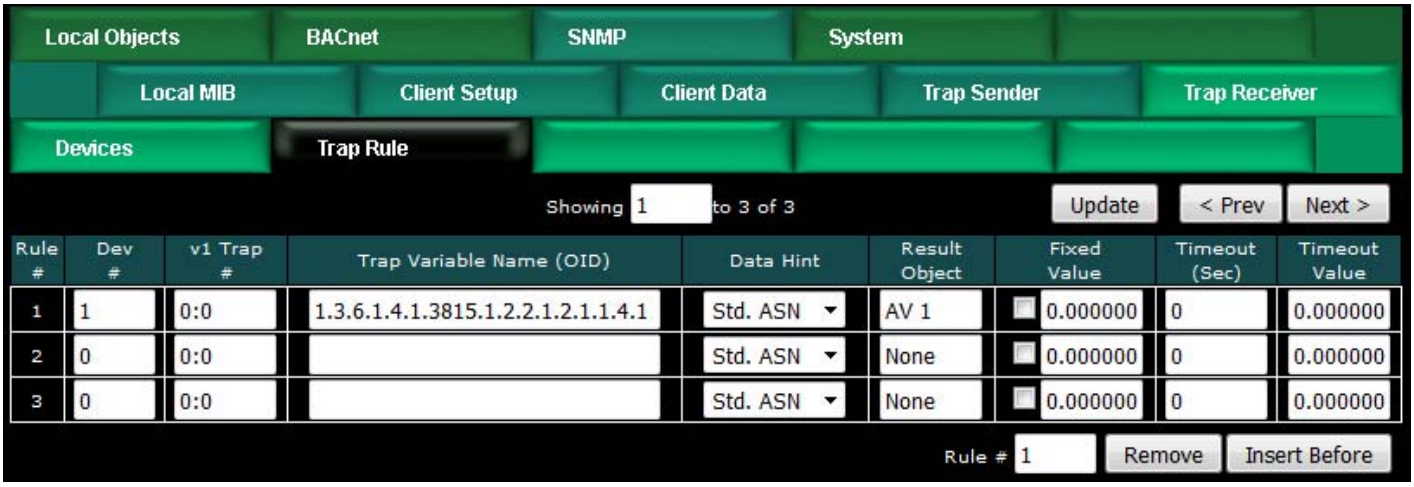


Binary Input 27 has been defined as follows.

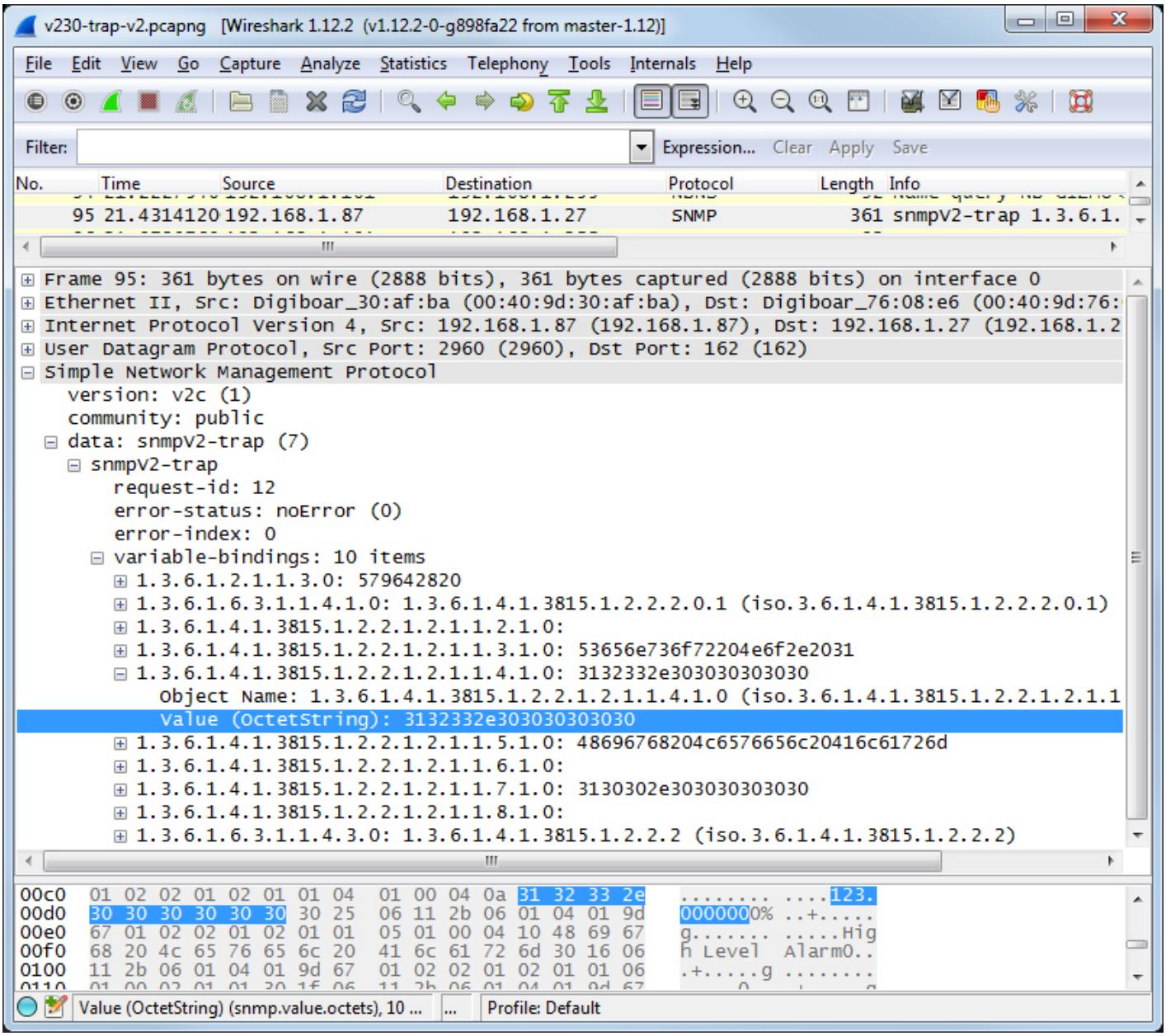


12.4 Trap Receive Example for SNMPv2

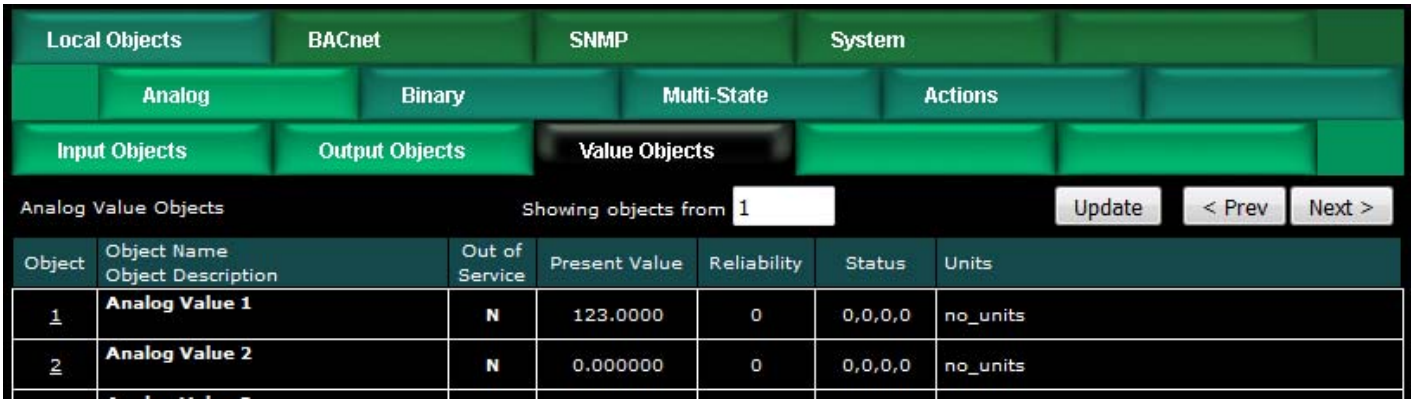
Most of what is discussed for SNMPv1 trap receive rules also applies to SNMPv2. The only real difference is that SNMPv2 does not send trap numbers, therefore you cannot use the v1 trap "a:b" number for v2 traps. Anything that will be recognized as a v2 trap will require a Trap Variable Name (OID). Other than trap number, everything else about configuring the v2 trap receive rule works the same as for v1.



The Wireshark capture of a trap from this v2 device is illustrated below. In this case, the OID in the trap rule identifies a numeric (floating piont) value that is sent as an ASCII string.



Numeric converstion of the Octet String is attempted automatically since the data hint is "Std. ASN" (standard ASN). The result is the correct value placed in a floating point Analog Value object.



13. Programming with Script Basic

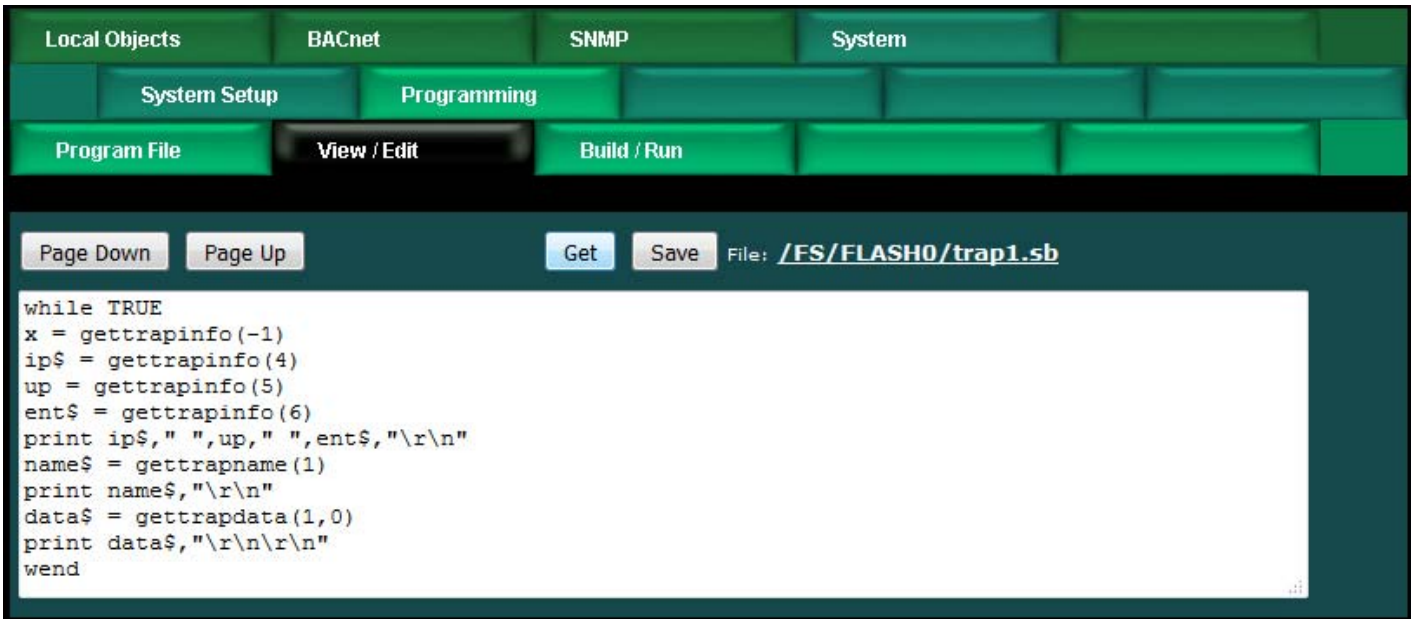
13.1 Creating a Program

Start by selecting a Directory. For most applications, you would select the Flash directory since anything stored in the Ram directory will be lost when power is lost.

Enter a new file name ending in ".sb" for your program. The program should use only alphanumeric characters and be limited to 20 characters. As soon as you click the New button, the file will be created and automatically selected. If you are returning to edit a previously created program. select that file from the File list, and click Select.

There is a local, very simple text editor available via the web View/Edit page. To edit an existing file, start by clicking Get. After entering a new program, or editing an existing program, click Save.

It is recommended that you use an external text editor for large programs, and simply upload it on the Program File page. The HTTP Post will only accept a maximum of 4K characters. Therefore, if you type more than will fit in a Post, you will be unable to save what you typed. For this reason, it is a good idea to type only a few lines per edit, and use an external editor for creating a longer program.



13.2 Testing the Program

The virtual terminal can be used while testing programs. Any "print" statement will send its output to the Output window, and any "line input" statement will take input from the Input window. The print and line input statements will have no effect when running in the background (i.e. running as Auto run from startup).

The Virtual Terminal page does not auto refresh, therefore any output produced by a print statement will not appear until you click the Refresh button. Some initial output may appear immediately since the program began running faster than the initial page refresh that occurs after clicking Start, but you will need to click Refresh to see additional output.

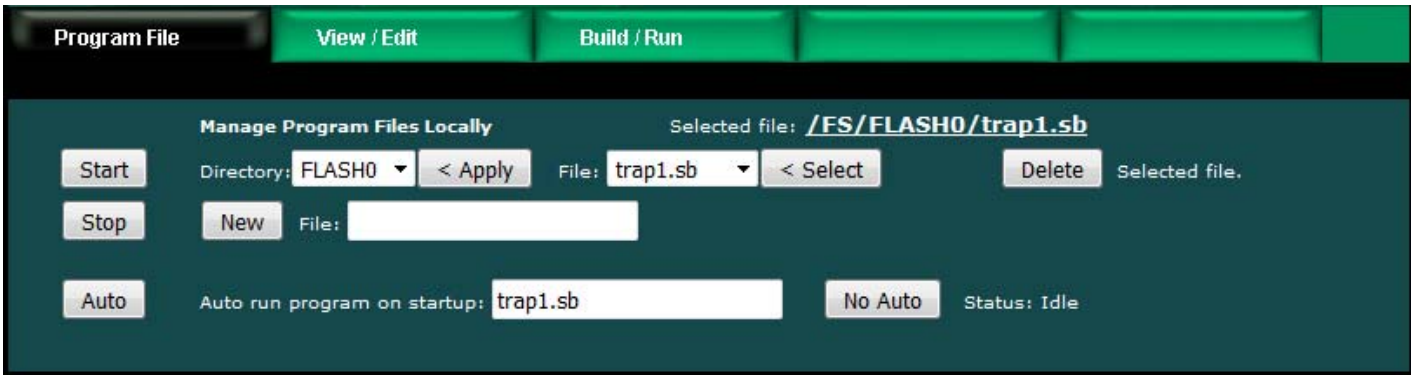
WARNING: If you are testing a program, you should select No Auto (see below), then restart the Babel Buster Pro. You will have two programs running at the same time if the auto run program is running and you are also running a program here. The results can be unpredictable if you are running two copies of the same or similar program at the same time.

An abbreviated screen shot of the Virtual Terminal is illustrated below. In this example, output from the above trap test program is illustrated.

13.3 Setting the Program to Auto-Run on Startup

Your program will not be very useful if it does not start up when the Babel Buster Pro boots up. You cause that to happen by selecting your file from the list, and then clicking the Auto button. Your program selection is saved in the configuration file, so to complete this process, you should next go to the Config File page and click Save (with your configuration file selected from the list).

From this point on, your program named in the "Auto run program on startup" window will be automatically started upon bootup. Of course, if the program has errors, it might not keep on running. Be sure to test your program ahead of time, and also consider use of the "On Error" feature of Basic. What exactly you might do upon error is entirely up to you, but one potentially useful option is to set an error number of your own making in a specific data object that will trigger attention in some way.



To eliminate the auto run program, simply click the No Auto button (and then go to the Config File page and click Save to save your updated configuration). Clicking the No Auto button will only make certain no program is started upon bootup from this

point forward. You will need to restart the Babel Buster Pro to halt any program that was already running from the last bootup.

13.4 Special Functions for SNMP Access and Diagnostics

The Babel Buster Pro includes several functions that are unique to its implementation of Script Basic. The language in general is summarized in Appendix C. A much more detailed collection of help may be found in the external compiled help file that may be downloaded at csimn.com. The compiled help includes reference guides for all of the string manipulation functions.

Appendix D outlines the special features that have been added to Basic to support BACnet and SNMP in the BBPRO-V230. The SNMP functions are also summarized here.

Trap Information Retrieval:

```
n = gettrapinfo (x)
```

This function is used first of all to see if any trap has been received. Then, if a trap has been received, it is used to retrieve several pieces of information about that trap. Each time gettrapinfo(0) or gettrapinfo(-1) returns a non-zero value indicating a trap was received, the previously retained trap is discarded. Until gettrapinfo(0) or gettrapinfo(-1) is called again, the same received trap will be retained for all subsequent trap related function calls.

For gettrapinfo(x), the various pieces of information are requested using selector 'x' using one of:
-1 = same as 0, but suspend until there is a trap to return
0 = return trap version if new trap (0=no new trap, 1=v1, 2=v2)
- each gettrapinfo(0) releases previous trap, queues up new trap if any for subsequent calls using x=1..6
1 = return trap generic number if v1, or 0
2 = return trap specific number if v1, or 0
3 = return number of varbinds in trap
4 = return IP address of sender as character string
5 = return uptime from trap as integer
6 = return enterprise OID from trap as character string

Trap Name Retrieval:

```
n = gettrapname (x)
```

Returns var name (OID) as string for varbind 'x', where x is 1..max as returned by gettrapinfo(3).

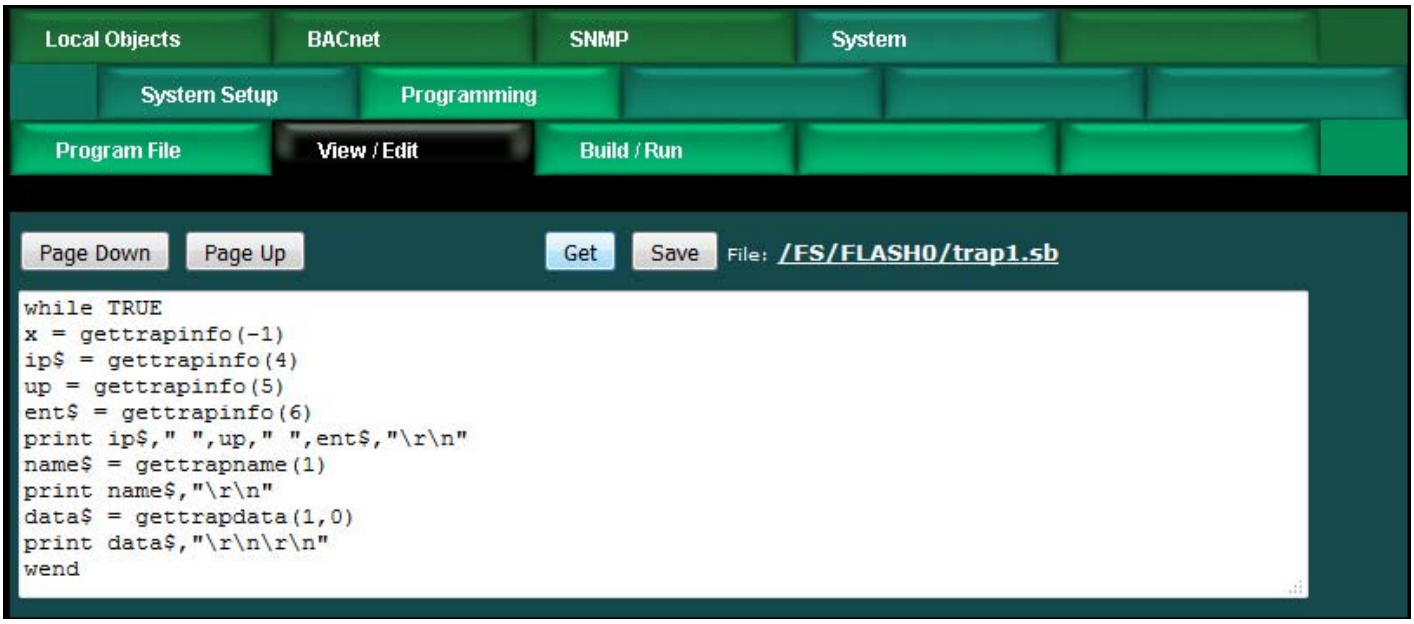
Trap Data Retrieval:

```
n = gettrapdata (x,y)
```

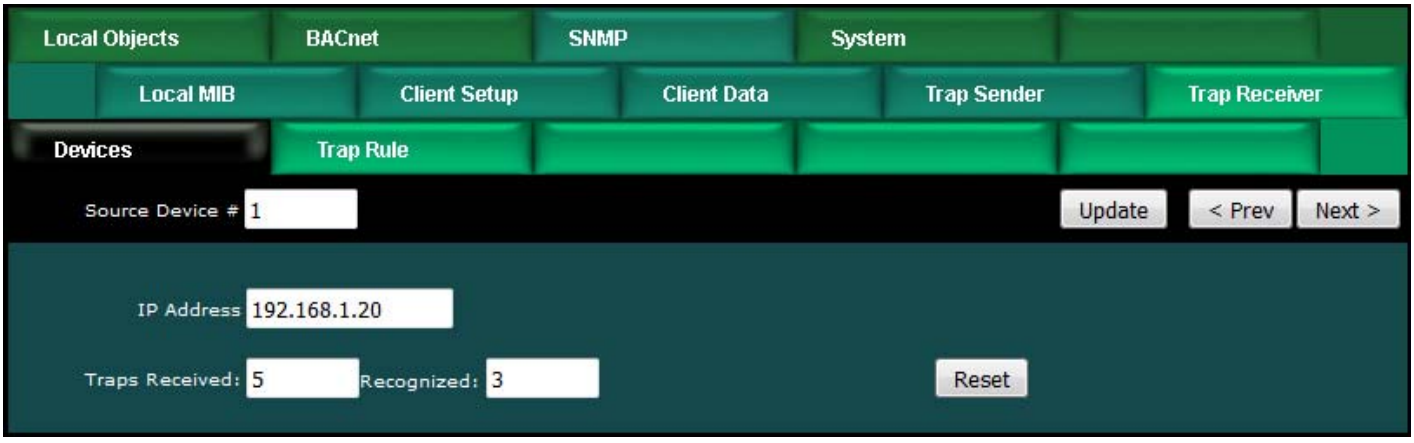
Returns var value for varbind 'x', where x is 1..max as returned by gettrapinfo(3). Format of data is specified by format 'y' where format is:
0 = want character string
1 = want integer value (convert as necessary)
2 = want floating point (convert as necessary) (also indicates RFC 6340 expected if octet string)

13.5 Example: Trap Receiver Test, SNMP v1

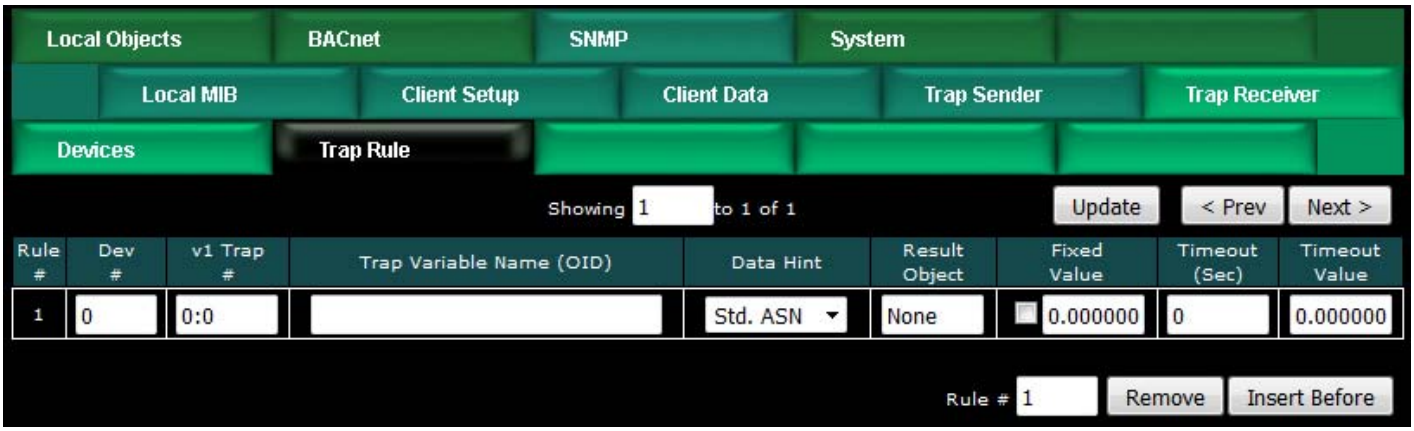
The following example program utilizes some of the special SNMP functions found in this version of Script Basic. Refer to the special functions descriptions above. This test program will receive a trap, and display results in the virtual terminal screen of the Babel Buster Pro. This is not useful for a final in-service program since output from the print functions are only viewable via the web pages. But this is useful to see if you are receiving traps in Basic.



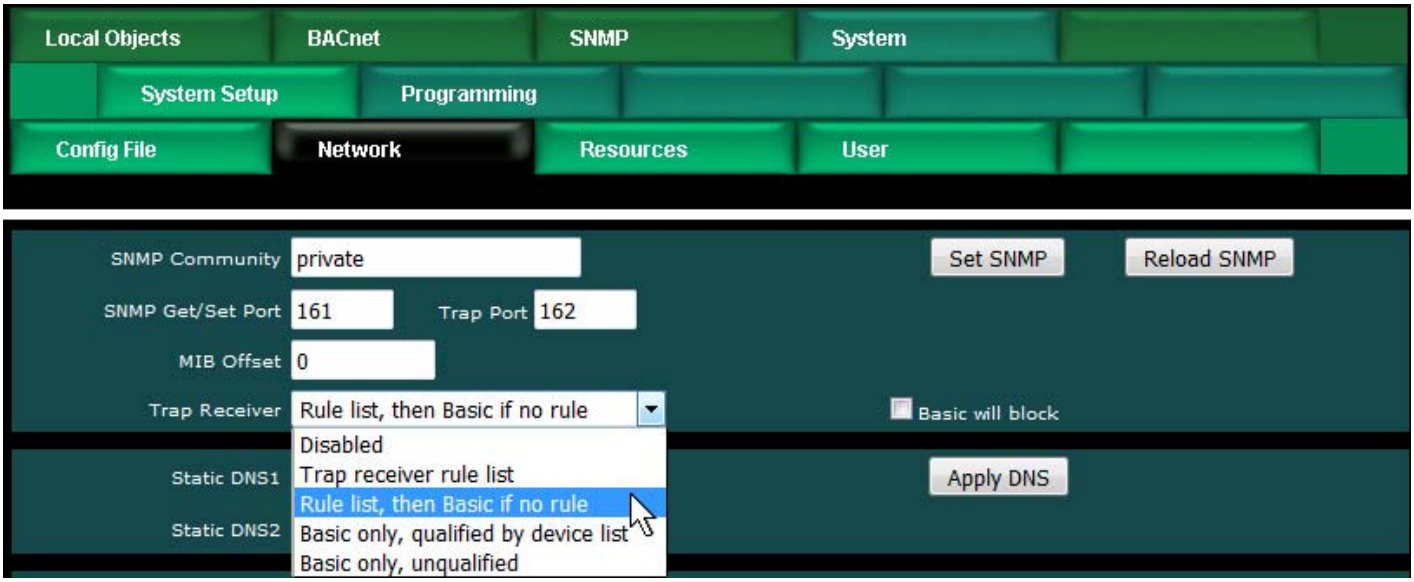
To set up for receiving traps, you need to enter the IP address of the device that will be sending traps on the Devices page of the Trap Receiver, assuming the trap receiver is enabled as illustrated shortly.



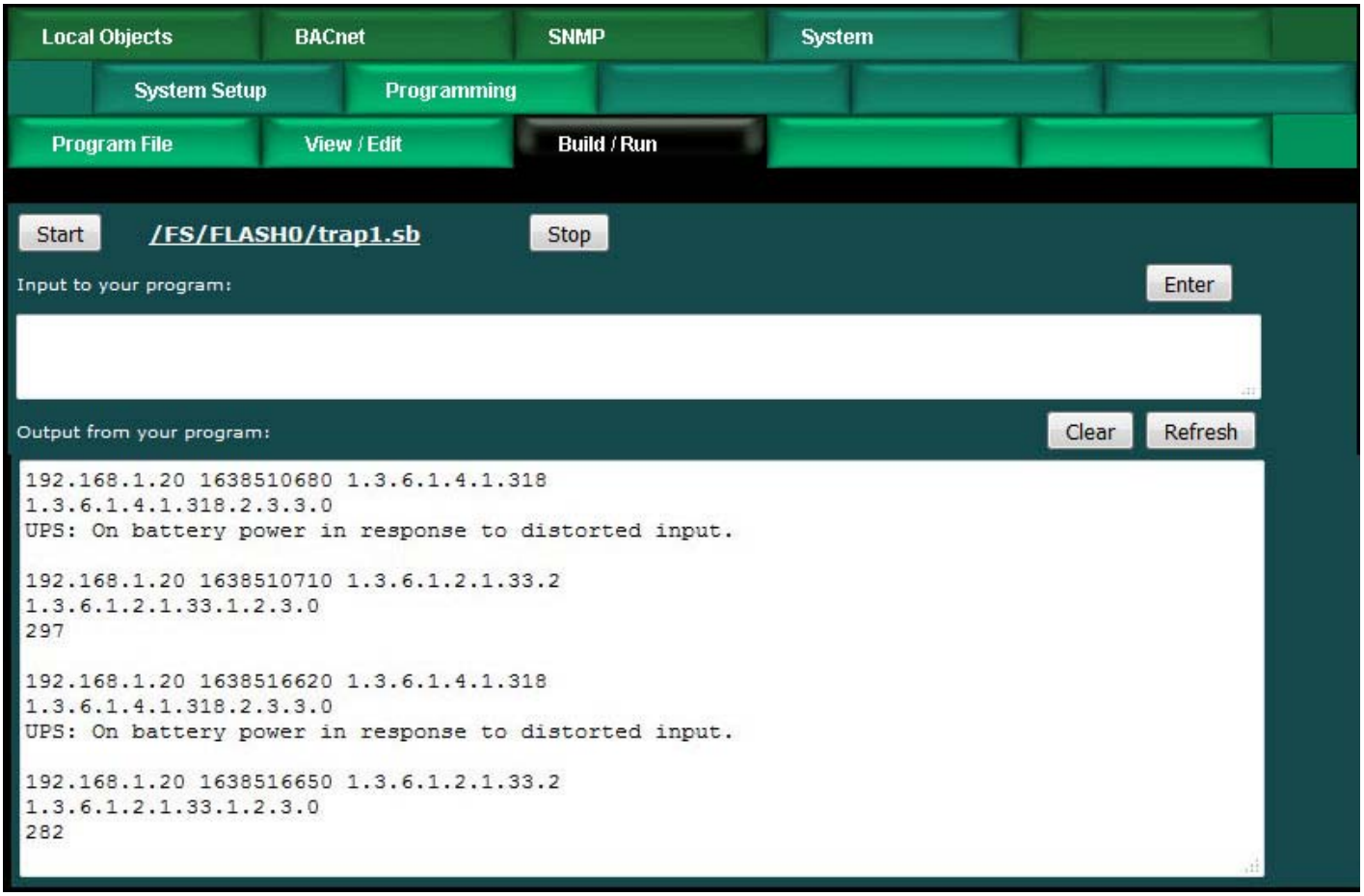
We are going to allow ourselves the option of adding trap rules later. For now, we want Basic to receive all traps from the above IP address, and therefore our Trap Rule list should be empty as illustrated here.



Enable trap receiving by selecting any of the Basic options on the Network setup page. If you select either of the "Basic only" options, then any rules entered later in the Trap Rule list will be ignored. Selecting "Rule list, then Basic if no rule" means that Basic will receive the trap if there is no rule matching the received trap.



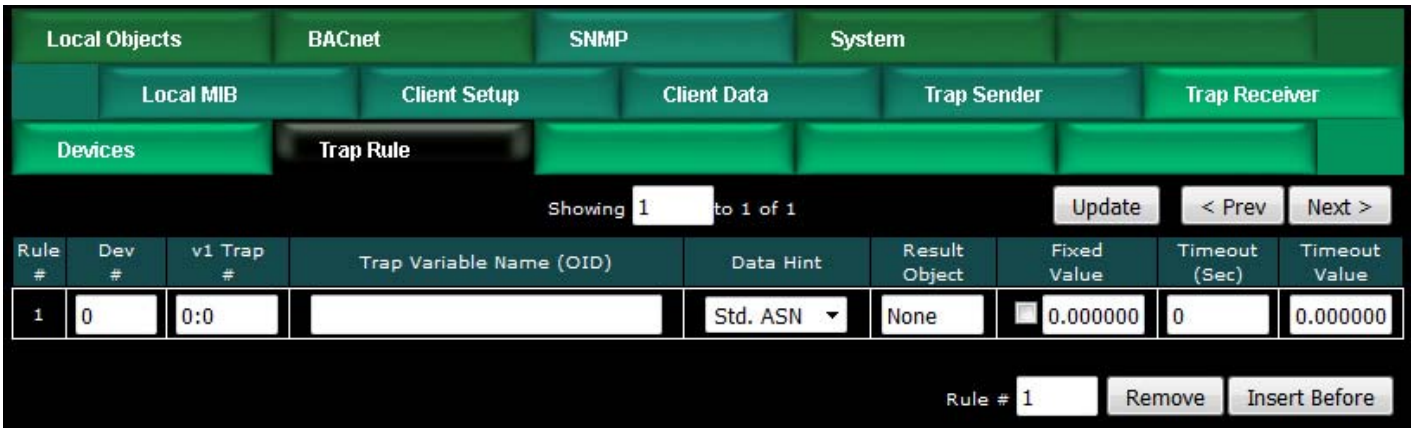
Click the Start button on the Build/Run page. Now cause your trap sending device to send some traps. In our example, we are receiving traps from an APC UPS. This UPS has both RFC 1628 enabled, and APC's own MIB. The traps from RFC 1628 are those with OIDs that start with 1.3.6.1.2.1.33 and the APC MIB OIDs start with 1.3.6.1.4.1.318.



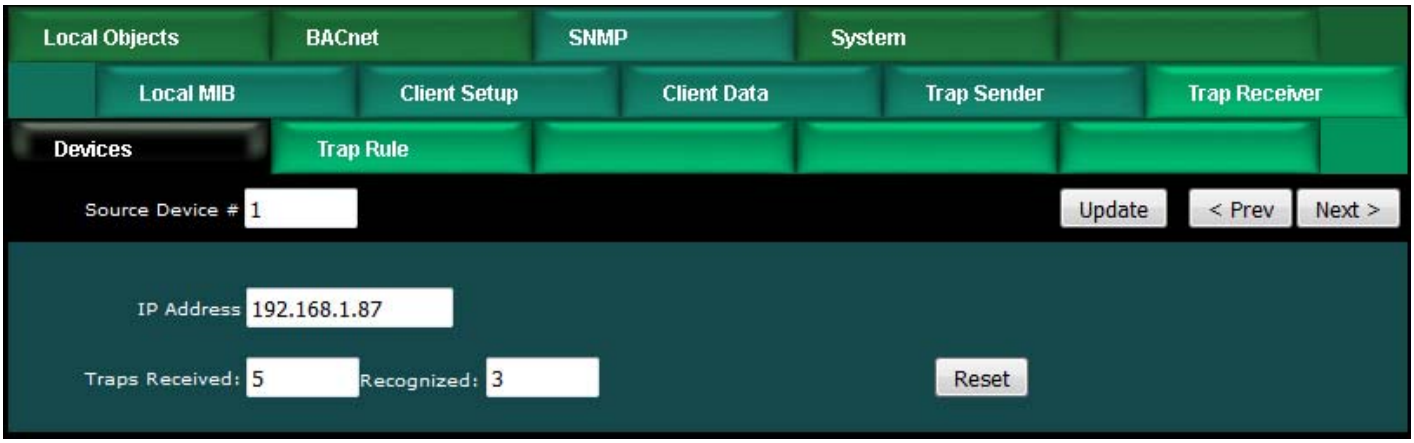
The traps received above happened when power to the UPS was removed. The traps below were received when power was restored.

13.6 Example: Trap Receiver Test, SNMP v2

The following example program utilizes some of the special SNMP functions found in this version of Script Basic. Refer to the special functions descriptions above. This test program will receive a SNMP v2c trap, and place results in BACnet objects. We sill start by making sure there are no trap receiver rules to get in the way of our Basic program.

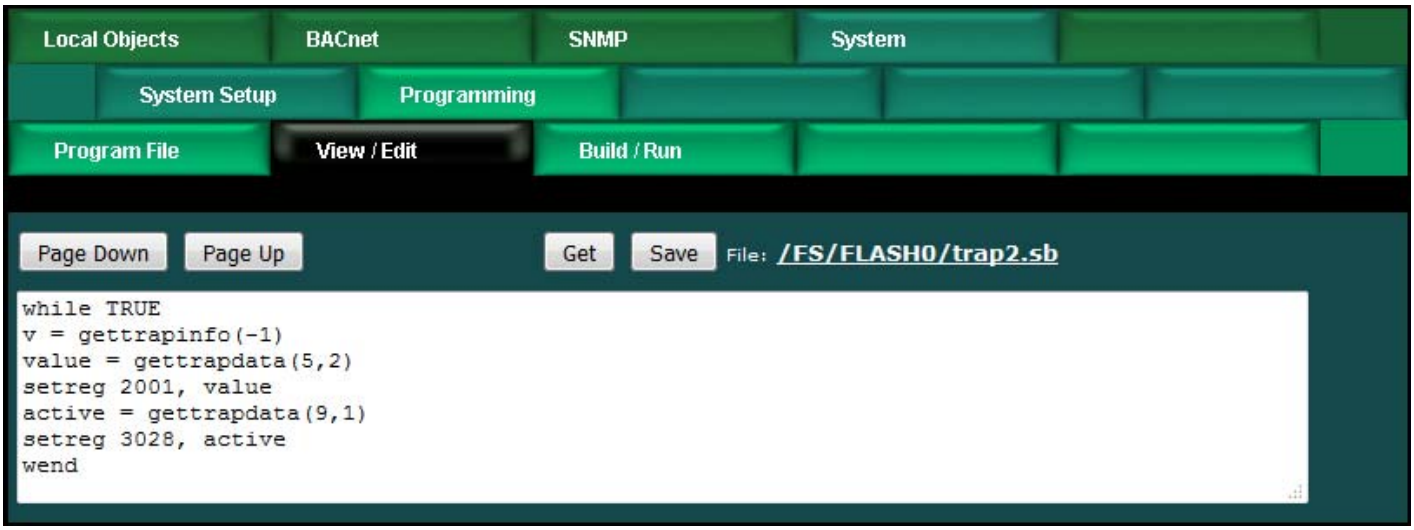


To set up for receiving traps, you need to enter the IP address of the device that will be sending traps on the Devices page of the Trap Receiver, assuming the trap receiver is enabled as illustrated shortly.



Enable trap receiving by selecting any of the Basic options on the Network setup page. If you select either of the "Basic only" options, then any rules entered later in the Trap Rule list will be ignored. Selecting "Rule list, then Basic if no rule" means that Basic will receive the trap if there is no rule matching the received trap.

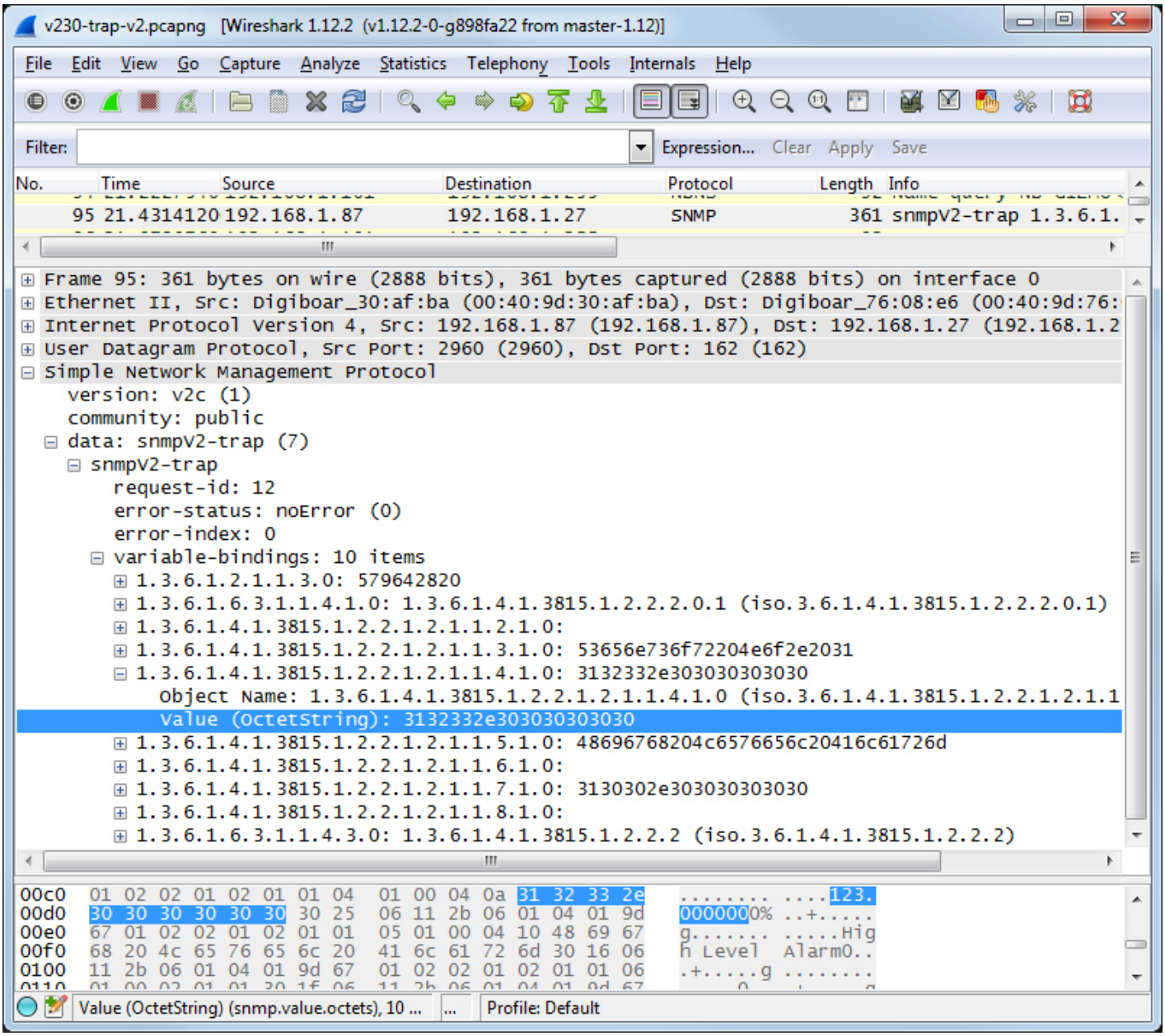
Our program, illustrated below, will wait for a trap, and then put two selected pieces of information in BACnet objects.



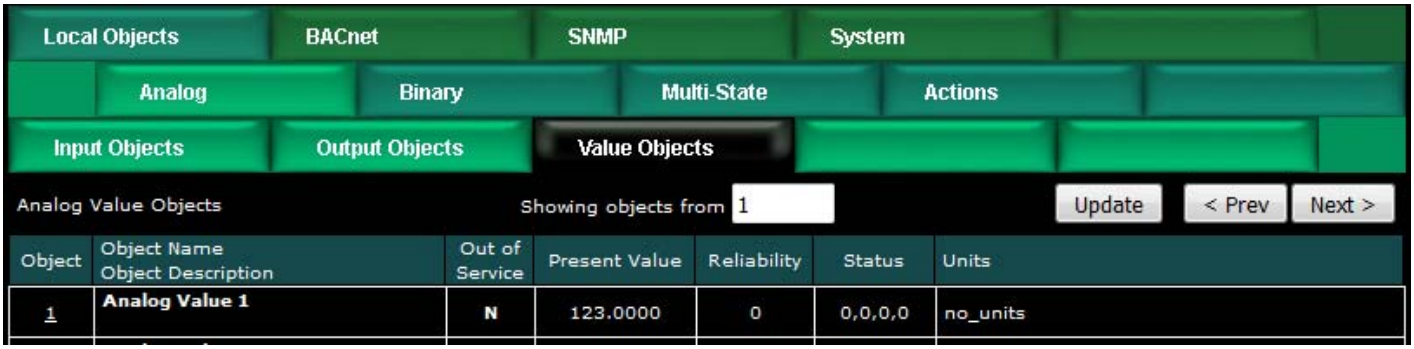
The MIB for the device sending traps in this case defines the content of traps as follows.

```
eventStateChange      NOTIFICATION-TYPE
      OBJECTS {
            eventRegNum,
            eventRegName,
            eventRegData,
            eventName,
            eventTestType,
            eventTestVal,
            eventState
      }
      STATUS            current
      DESCRIPTION
            "This trap is sent when an event changes state
            if notifications are enabled for the object that changed."
      ::= { modbusTraps 1 }
```

The trap as captured by Wireshark looks like the following. The string highlighted is the present value of the object that generated the trap.

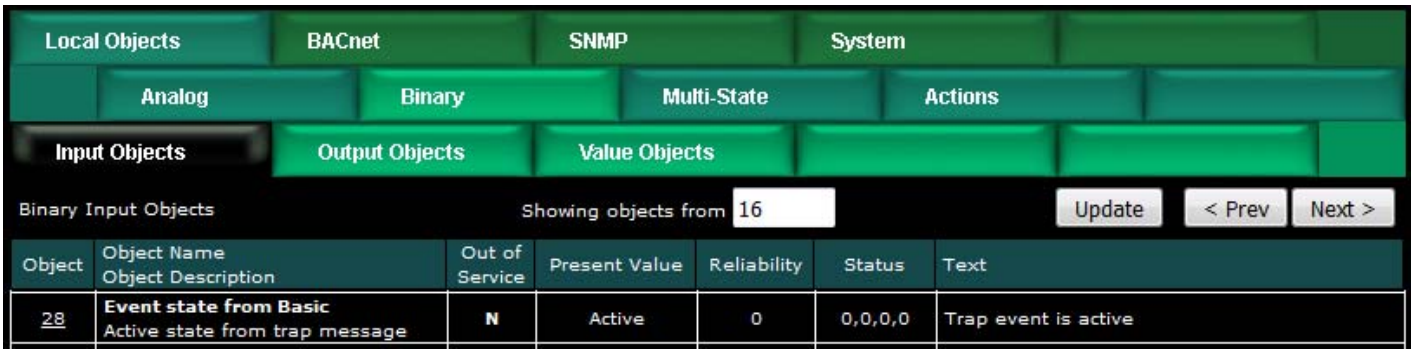


The first 'gettrapdata' in our program will get the value received in the trap, and the subsequent 'setreg' places that value into Analog Value 1, with the result illustrated below.



The second 'gettrapdata' is getting the active/inactive state of the event sent in the trap, and the subsequent 'setreg' places that state in Binary Input 28, with the result

illustrated below.



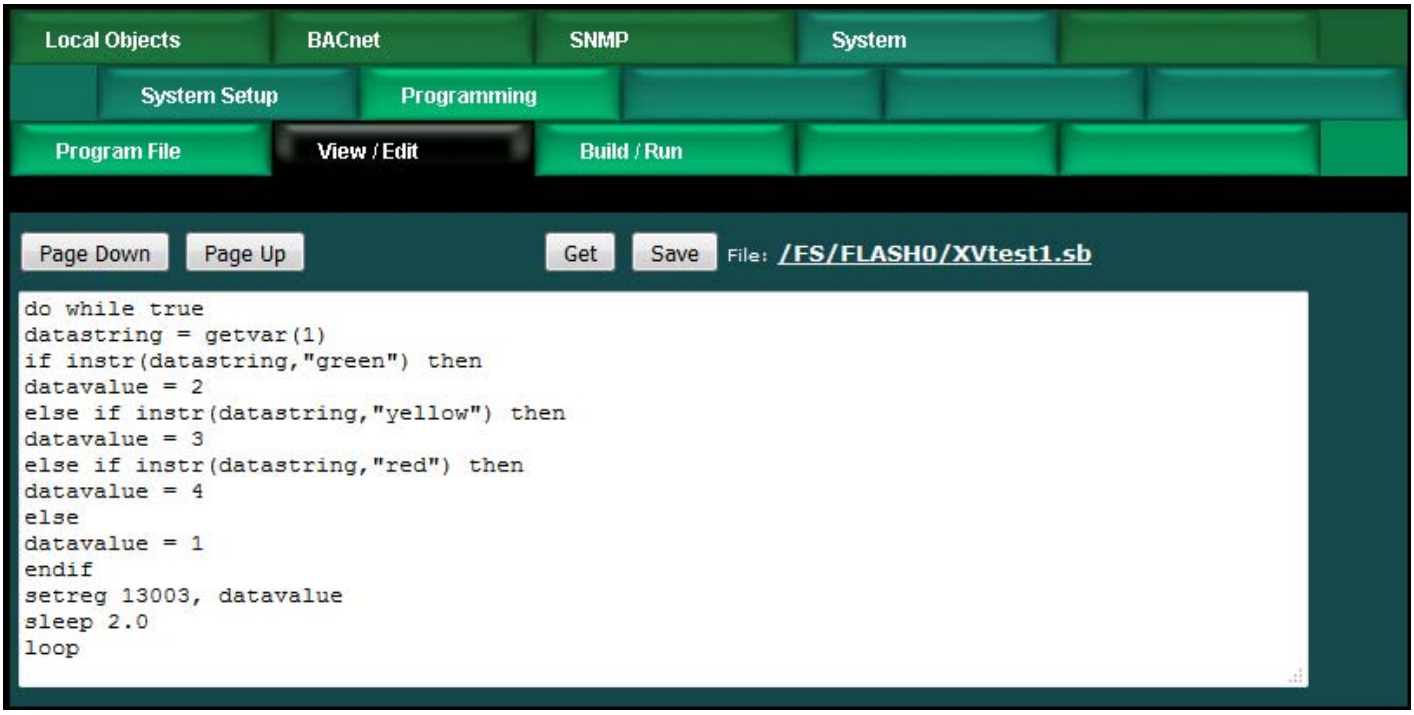
13.7 Example: Converting Octet Strings to BACnet Objects

The SNMP Client Read Map in this example is going to be reading a character string from another device. We cannot directly convert this to something meaningful to BACnet. Therefore, the read map stores the string in a virtual object XV 1, and Basic will perform the conversion.

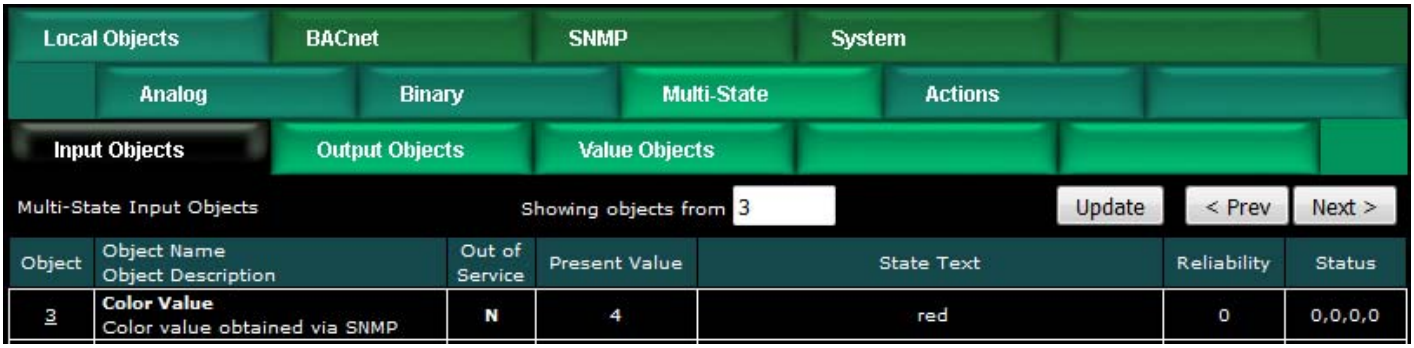
Virtual Objects are internal variables used to transfer non-standard data between SNMP and Basic. They have no other purpose and are not directly accessible via BACnet.

The Basic program that is looking at the character string and deciding how to translate it is illustrated here. The 'getvar' function retrieves XV 1 into a string variable. Then

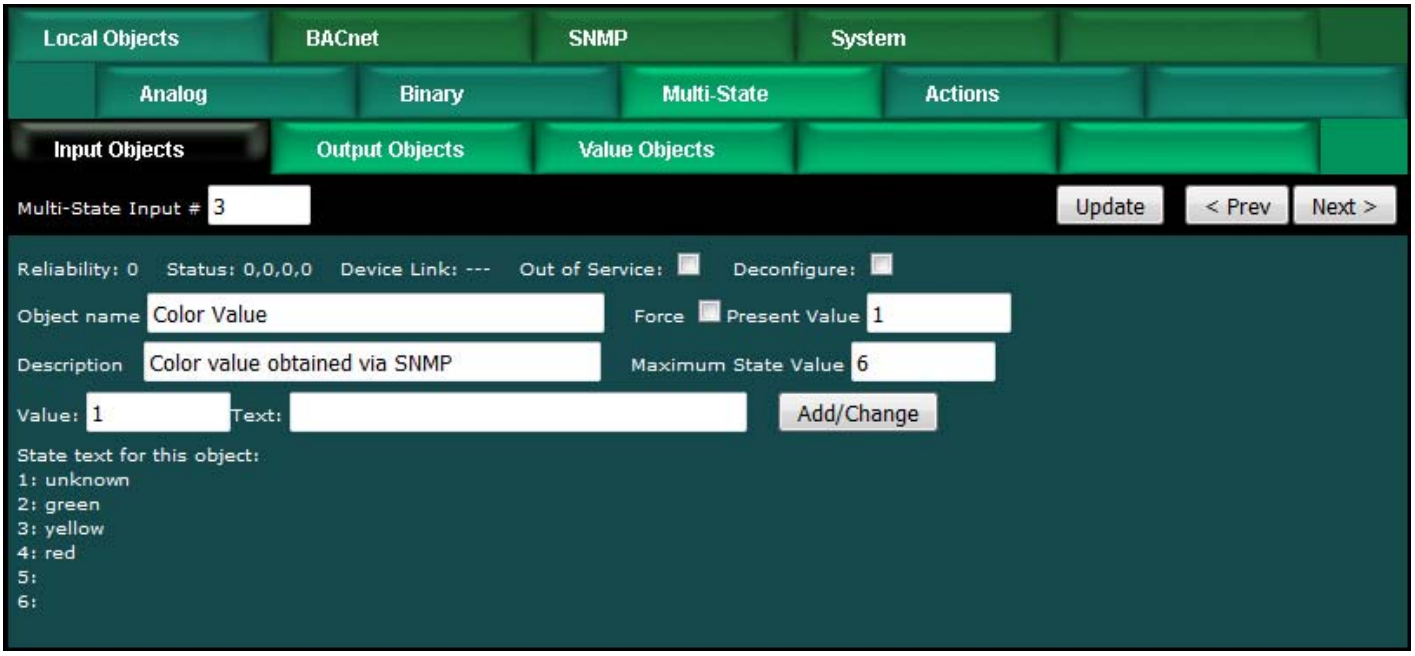
Basic goes down a list of expected strings and produces a corresponding number. That number is then placed into the BACnet object Multistate Input 3.



The resulting value in the Multistate object reflects the color provided by the SNMP query.

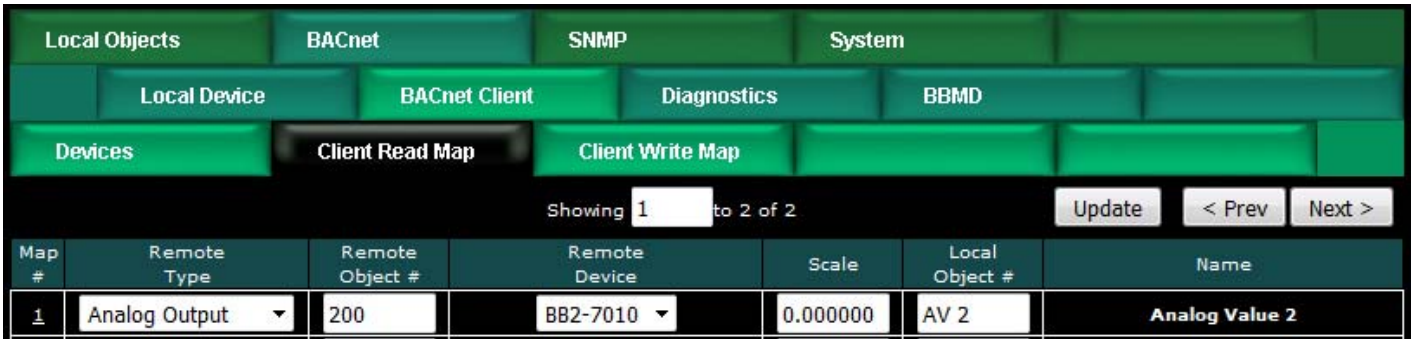


This translation happens because we have defined the states to be the color names as illustrated below. You would typically do something more meaningful than pick colors, but this simple example shows the concept of converting non-standard Octet Strings into BACnet objects. The SNMP data type known as an Octet String can be any string of bytes including raw binary or ASCII character strings.

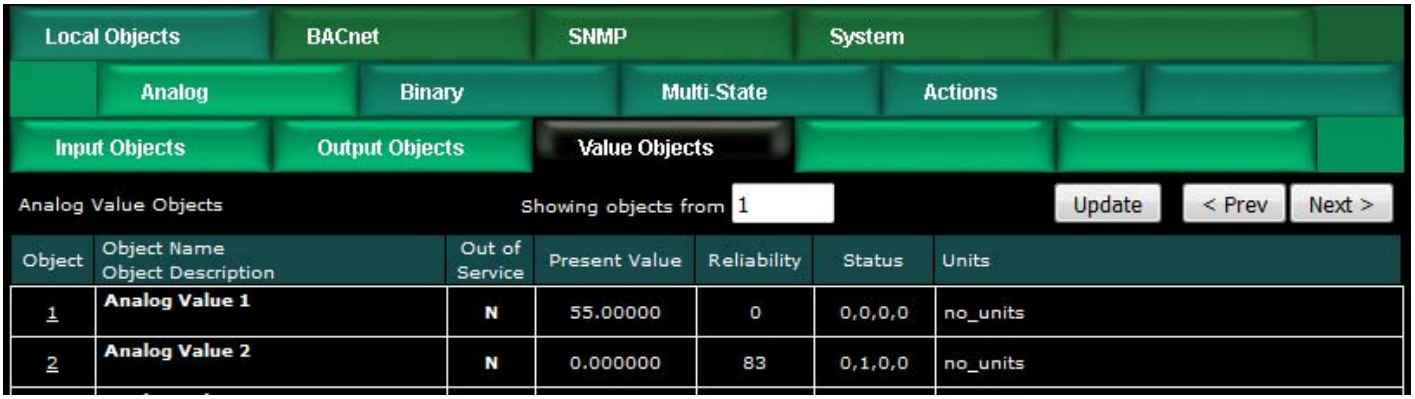


13.8 Example: Device and Rule Diagnostics

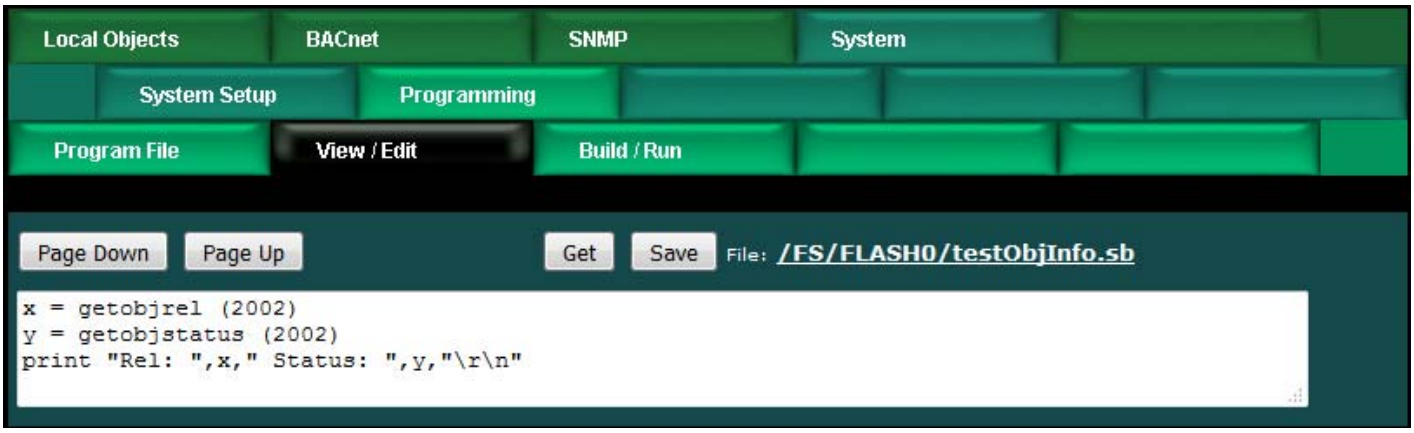
The special functions of greatest interest are probably the SNMP functions, but here is a short example illustrating the diagnostic functions also available. In this example, we are checking on our Babel Buster Pro's attempts to query a BACnet device. We have created a BACnet Client Read Map that will attempt to query an object we know does not exist.



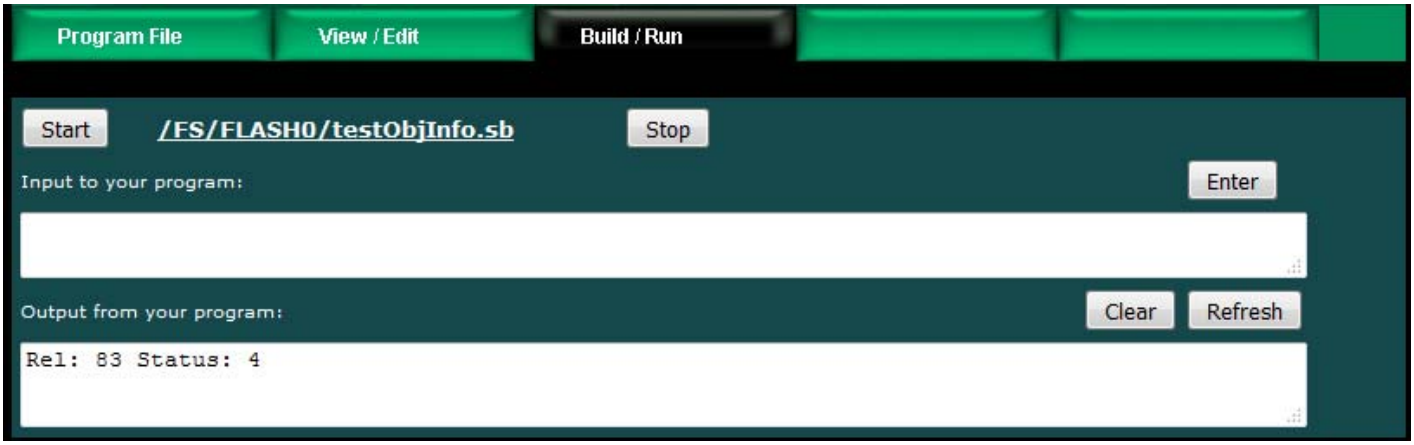
The resulting reliability code will be 83, error code returned by the device, and status will indicate a fault.



The functions for retrieving this status information from Basic are illustrated below. For test purposes, we are only displaying the results in the Virtual Terminal. Your program would take some specific action based on these results.



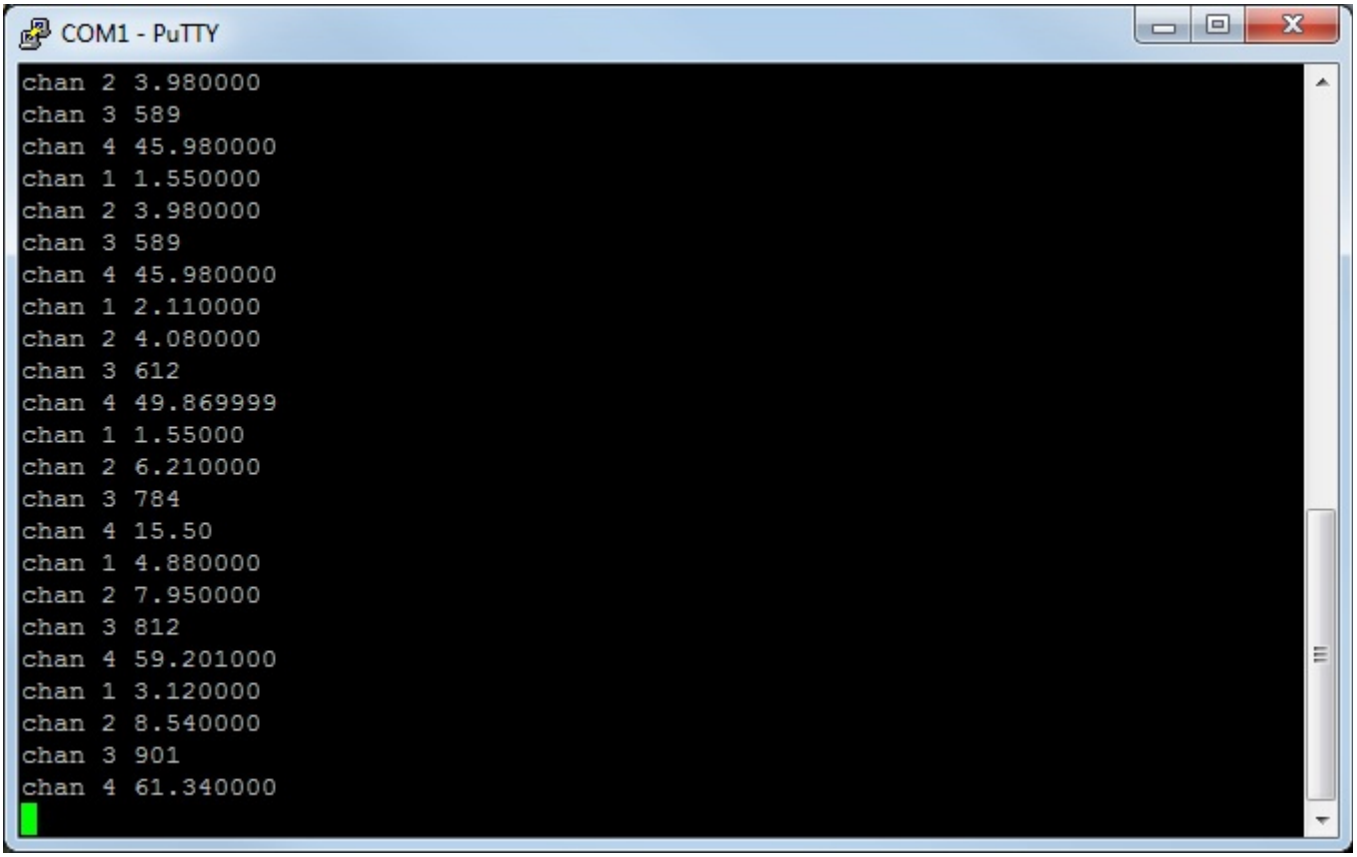
When the above program is run with the status as displayed above, the program results are illustrated below.



13.9 Example: BACnet Enabling a Serial Device

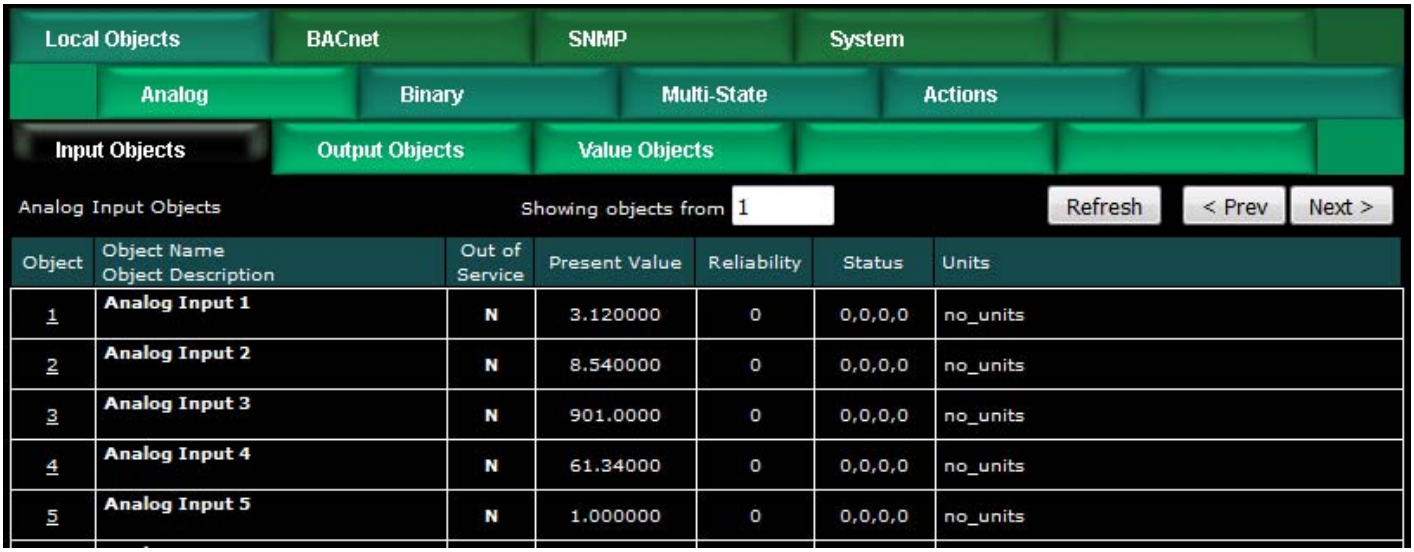
A very useful feature available with the Serial Port version of the Babel Buster Pro V230 is the ability to use the serial port for a proprietary serial protcol instead of MS/TP. Both RS-485 and RS-232 options are available, but you must order the -SP version of the PRO V230 in order to use this feature. MS/TP is disabled in the V230-SP. Instead, Basic can directly communicate with the serial port.

Our example will capture data from a data logger type device that periodically sends strings of data. In this case, it is simply a channel number and that channel's data value. This device is a 4-channel device, and its output is illustrated here by simply connecting it to a PC (via RS485 to RS232 adapter if applicable) and running PuTTY to see its output.



The program to capture data from this device and store the results in local objects is illustrated below. The key line to notice here is the "split" where the data line is effectively parsed. The "toss$" is going to end up holding the string "chan" which we don't care about. The numbers representing channel number and that channel's data value will end up in the variables "chan" and "data". Then, based on channel number, we calculate an object number. This example makes the object calculation easy, it is using Analog Input objects. We would add 2000 to the channel number to use Analog Value objects instead.

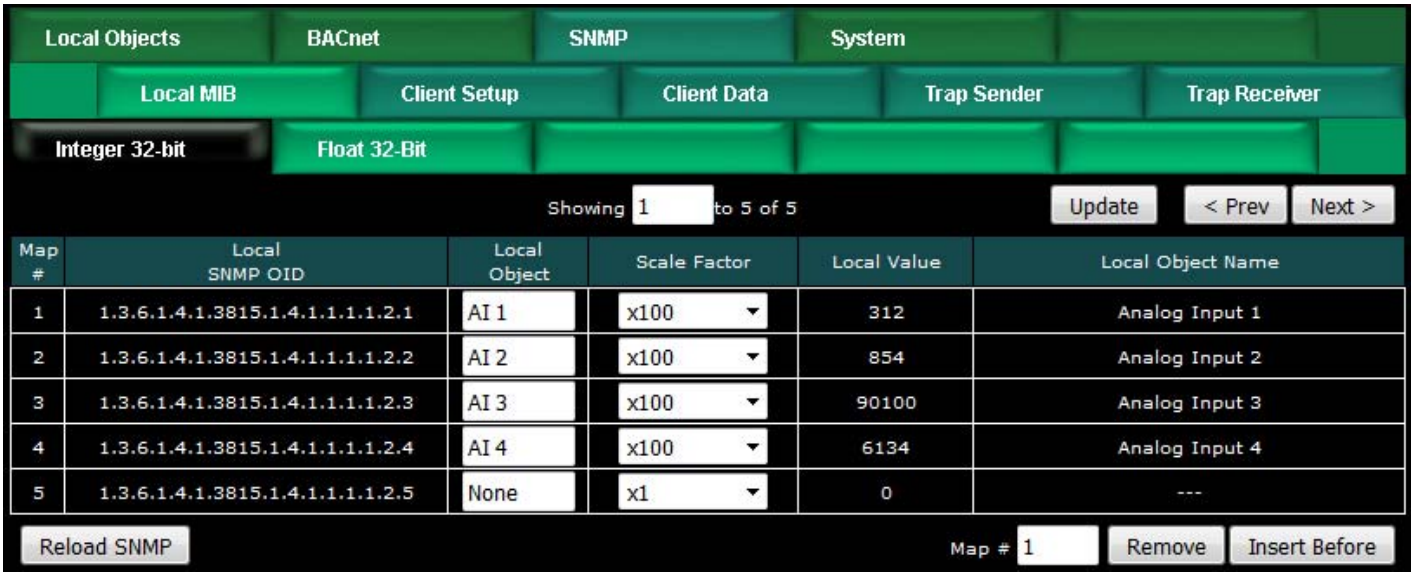
The local object values are going to continue changing as new data is received. The screen shot below illustrates the most recently received data in this example.



Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Units
1	**Analog Input 1**	N	3.120000	0	0,0,0,0	no_units
2	**Analog Input 2**	N	8.540000	0	0,0,0,0	no_units
3	**Analog Input 3**	N	901.0000	0	0,0,0,0	no_units
4	**Analog Input 4**	N	61.34000	0	0,0,0,0	no_units
5	**Analog Input 5**	N	1.000000	0	0,0,0,0	no_units

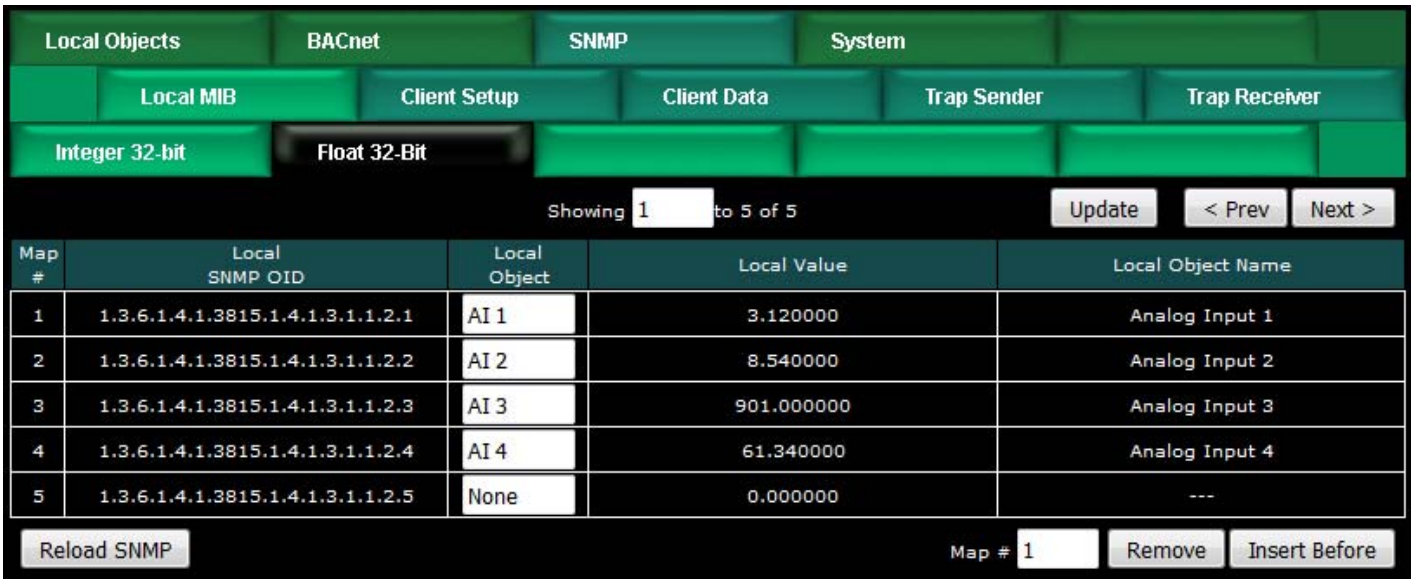
13.10 Example: SNMP Enabling a Serial Device

SNMP enabling a serial device using the BBPRO-V230-SP starts with BACnet enabling the serial device as illustrated in the previous example. From there, we simply assign the objects to the MIB as illustrated below.



Map #	Local SNMP OID	Local Object	Scale Factor	Local Value	Local Object Name
1	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.1	AI 1	x100	312	Analog Input 1
2	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.2	AI 2	x100	854	Analog Input 2
3	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.3	AI 3	x100	90100	Analog Input 3
4	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.4	AI 4	x100	6134	Analog Input 4
5	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.5	None	x1	0	---

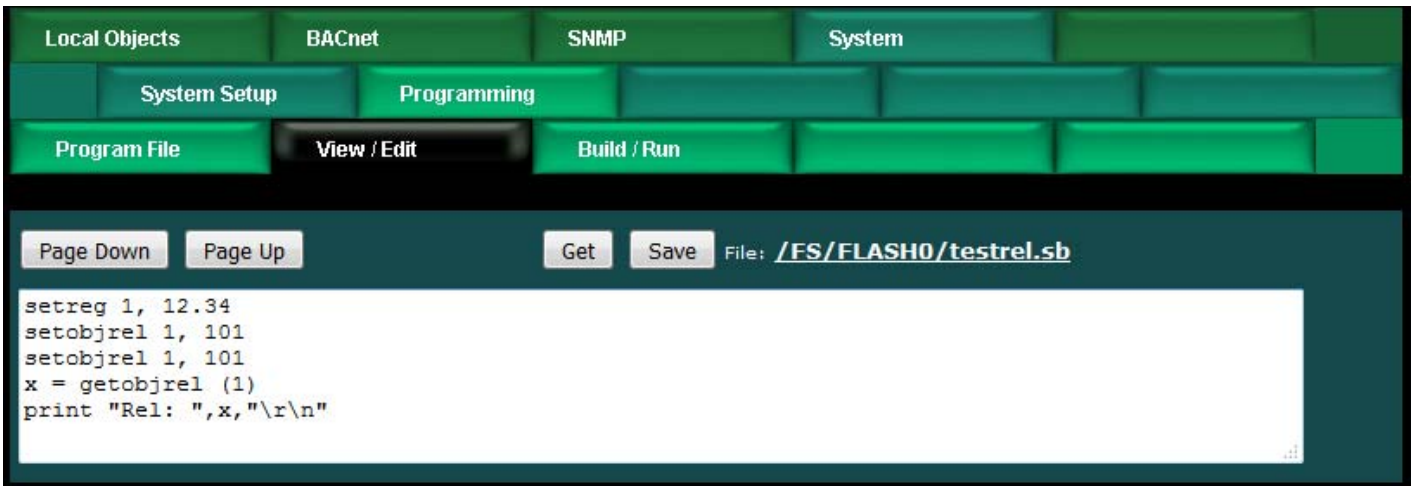
The above example assigns our objects to the Integer branch of the MIB with scaling. If you wanted to assign them to the Floating Point MIB branch instead, you would assign them as follows (and you can assign the same objects to both MIB branches at the same time if you wish).



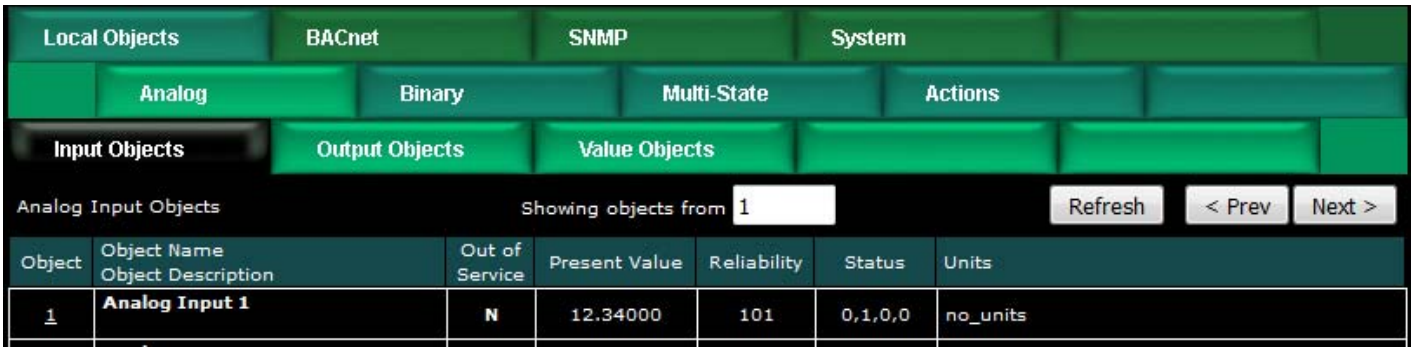
13.11 Example: Getting, Setting Object Status from Basic

Section 13.8 above showed an example of obtaining object status in Basic. This applies when the BACnet or SNMP client has detected a problem and we want Basic to know about it. But what if we want to set object status telling the outside world that our program has detected a problem? Basic can do that, and object reliability codes 101 through 199 have been reserved (for Basic) in the Babel Buster Pro for just that reason.

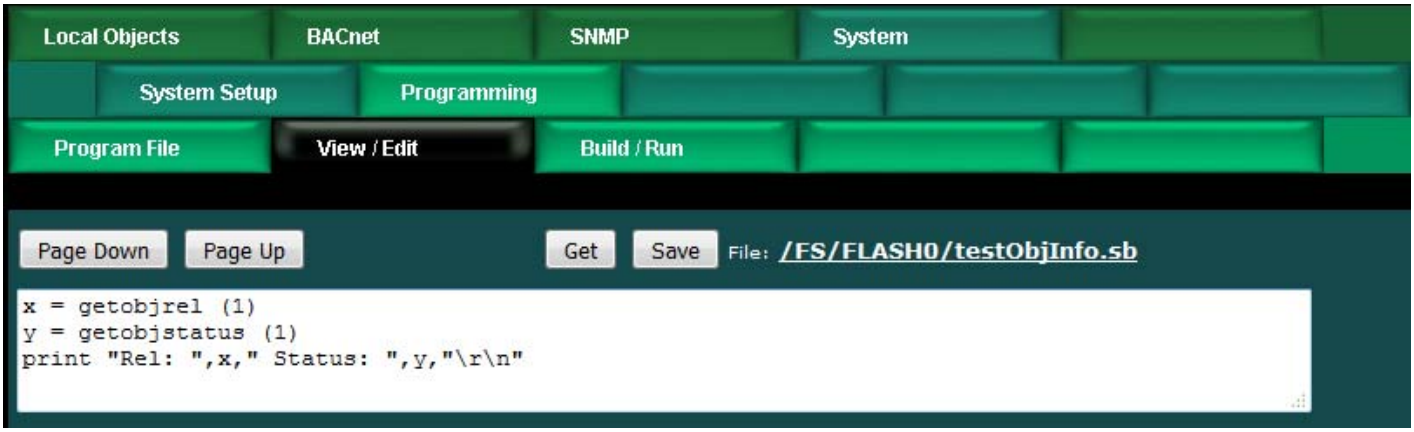
The program below will set the Present Value to 12.34 and reliability code to 101 for Analog Input 1. Then it will get the object's reliability code and print it to the virtual terminal just for test purposes.



Note that there are two consecutive calls to 'setobjrel'. The BACnet objects in the BBPRO-V230 need to see the same reliability code twice before it will be registered as the offical code. This is to avoid triggering action on spurious events like a one-time communication hiccup. If you are setting the reliability code each time through an indefinite loop, only one call will suffice since the same persistent problem will result in repeated calls with the same reliability code.

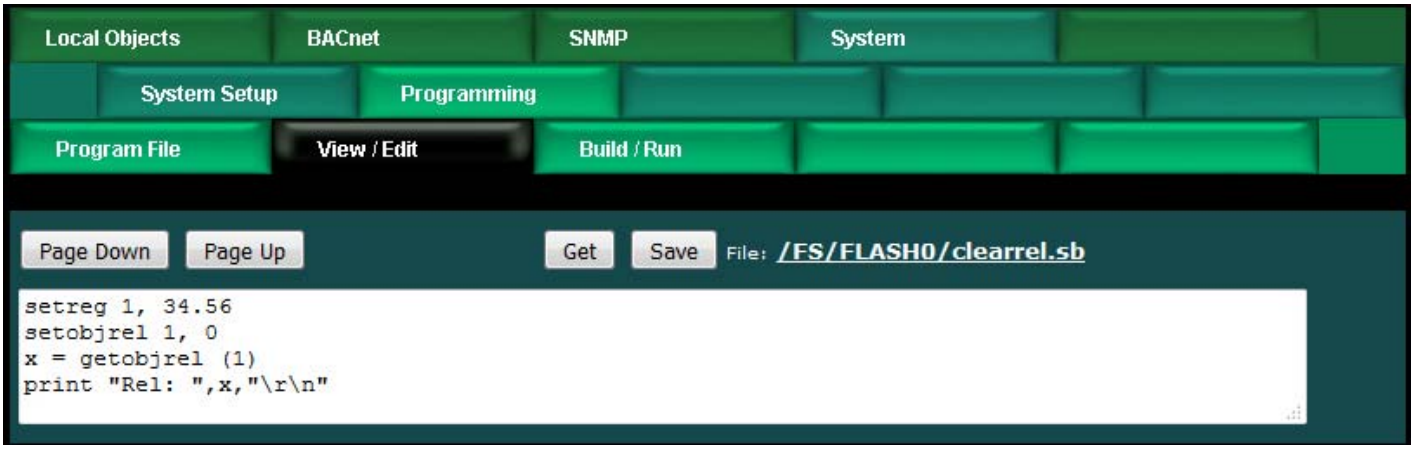


The result of running the 'testrel.sb' program should be as illustrated above.

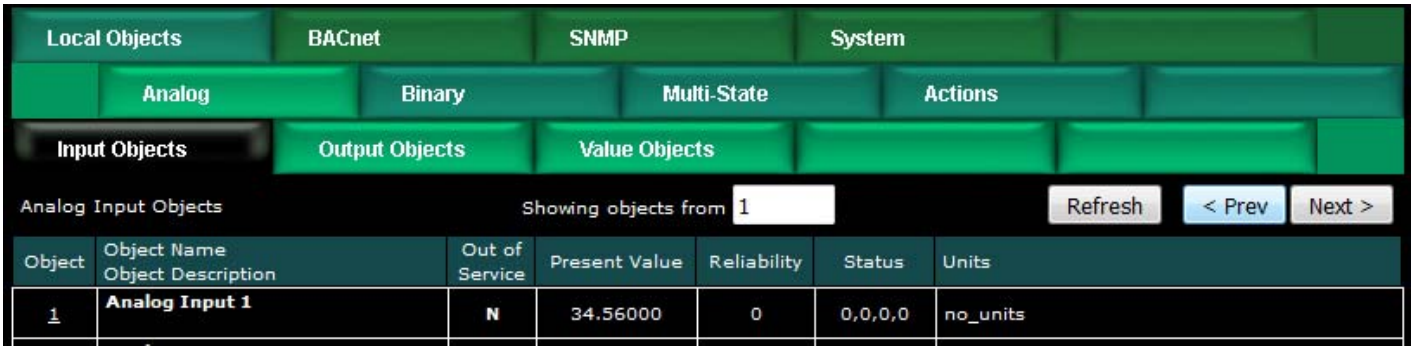


The above program appeared earlier. This time, after running the first example calling 'setobjrel, we will get the following results:

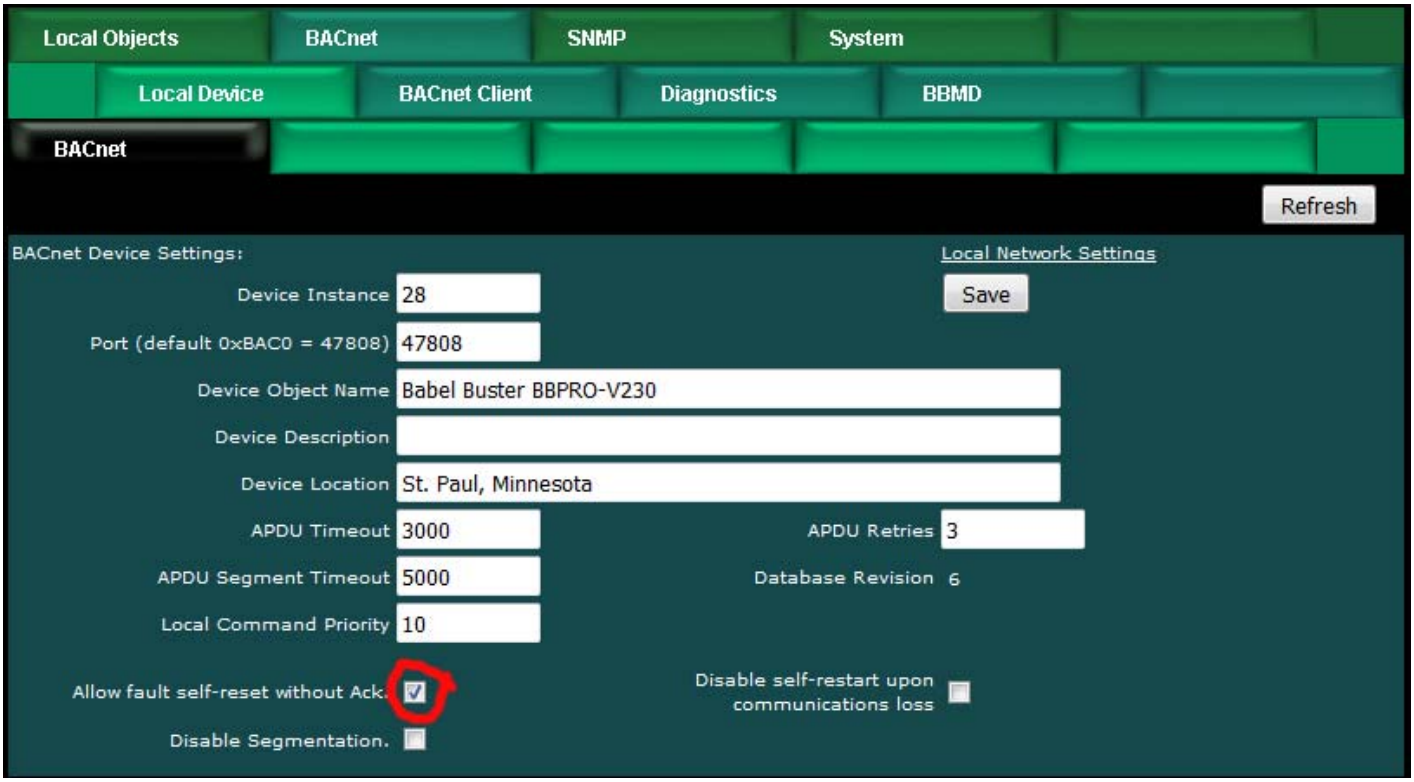
To clear the object reliability, you only need to call setobjrel once with a code of zero. The fault status is automatically set any time the reliability code is nonzero, and cleared when the reliability code is zero (meaning no errors).



The result should be as follows.



You may find that the reliability code does not clear to zero simply by calling 'setobjrel' with a code of zero. Reliability codes are latching in nature to better allow diagnosing problems. The code will not be allowed to "unlatch" until it has been read by a BACnet client. To override this latching, go to the BACnet page and check the box marked "Allow fault self-reset without Ack" and click Save.


Local Objects
BACnet
SNMP
System
Local Device
BACnet Client
Diagnostics
BBMD
BACnet
Refresh
BACnet Device Settings:
Local Network Settings
Save
Device Instance 28
Port (default 0xBAC0 = 47808) 47808
Device Object Name Babel Buster BBPRO-V230
Device Description
Device Location St. Paul, Minnesota
APDU Timeout 3000
APDU Retries 3
APDU Segment Timeout 5000
Database Revision 6
Local Command Priority 10
Allow fault self-reset without Ack.
Disable self-restart upon communications loss
Disable Segmentation.

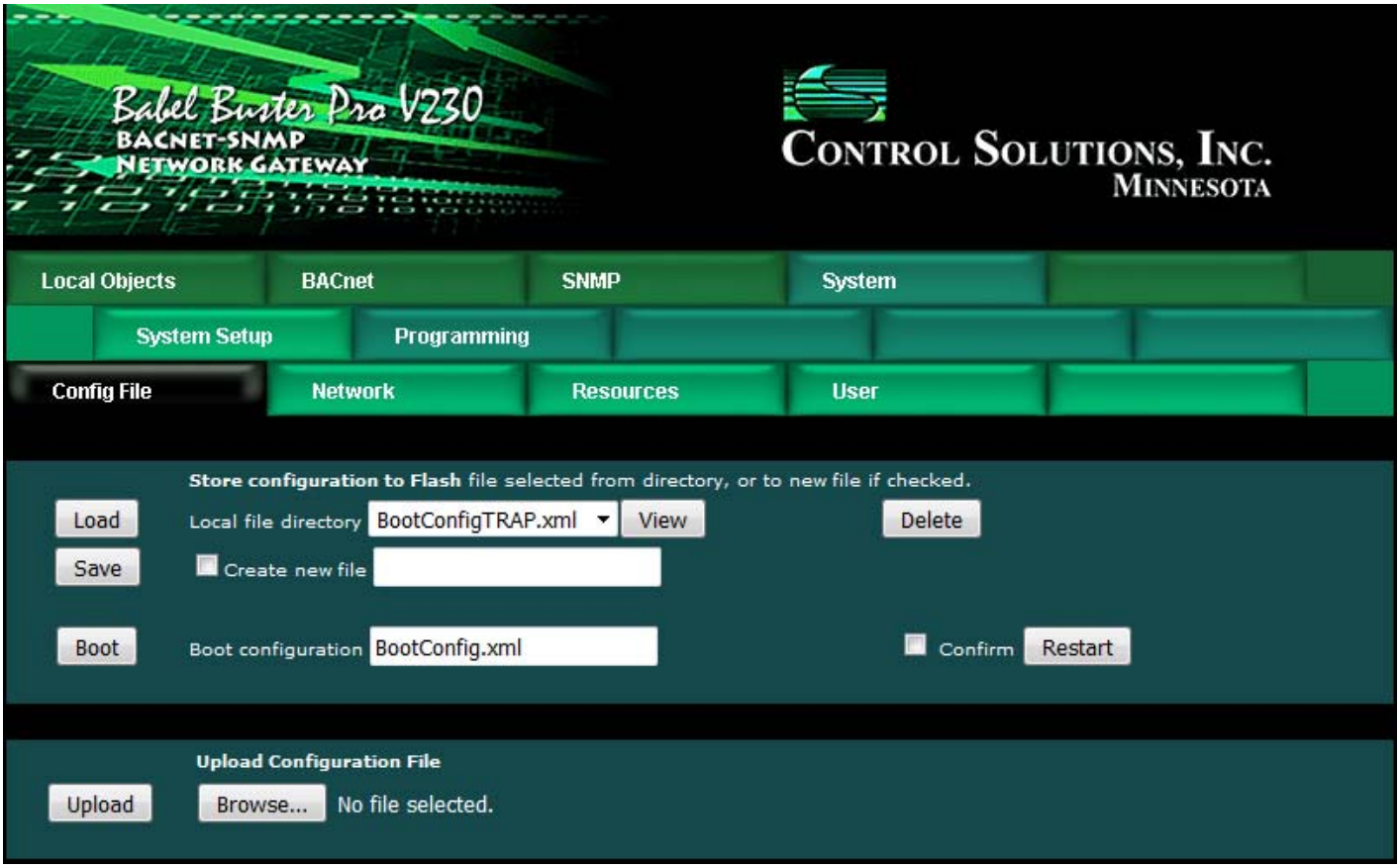
14. System Configuration and Resources

14.1 Configuration Files and Restoring Default Settings

The Config File page is probably one of the most important pages to know about. This is where you tell the gateway to save all of the changes you have made. The various "Update" buttons on the many pages in the web user interface only copy your configuration from your PC's browser to temporary memory in the gateway. To retain those changes indefinitely (i.e. through restart or power cycle), you need to tell the gateway to save those changes in a configuration file.

The configuration files are stored in non-volatile (Flash) memory. The process of reprogramming the Flash takes a little time. It would be cumbersome to rewrite that file every time you made a minor change. Therefore, in the interest of being more responsive, and in the interest of extending the life of the Flash, configuration is only saved to Flash when you direct it to do so.

This is also where you tell the Babel Buster Pro what configuration to automatically load upon startup. To set the startup configuration, select the file from the list, and click the Boot button. The name of the startup file, along with a few other important things like the gateway's own IP address, are stored in a different area of Flash that is not part of the file system. There will be a delay while this area of Flash is reprogrammed after clicking the Boot button.



There is a means of restoring the Babel Buster Pro to "manufacturer's default settings". First of all, make sure that the Boot configuration file is set to "BootConfig.xml". Then, after selecting this file as the boot file, delete it. Now restart the gateway. Upon restart, and upon finding that the boot configuration name is BootConfig.xml, and it does not exist, the gateway will automatically create one with default parameters. The automatic creation of a default file will not occur with any other file name.

To create a new configuration file, enter a name in the new file window (making sure it has a .xml suffix), check the "create new" box, and then click Save.

It is possible to manually edit the XML file outside of the gateway. However, doing so is very prone to errors. If there are errors in the XML file, it will not load successfully on startup. If the configuration does not load on startup, none of the scanners will begin scanning. Because they are all blocked by configuration failure, entering new configuration via the web pages will not result in functionality being restored. You must successfully load a configuration file before the gateway will become functional. To check for errors, select the file here, and click the Load button. Error messages that would have been discarded by the automatic loading at startup will now be displayed on an error page if there are any.

To save a copy of the configuration to your PC, select the file and click the View button. Your browser will now display the XML file. DO NOT do a text copy/paste to try to create an XML file - doing so will result in an invalid file format that cannot be loaded again. You must use the browser's "save as" or "save page" function. The

browser should default to wanting to save a file with a .xml suffix. If correctly saved on your PC, you should be able to double click on the saved file and it will result in opening the file automatically in your browser. It was saved correctly if the browser does not give any error messages when displaying the XML (which should now look exactly as it did when you first clicked the View button).

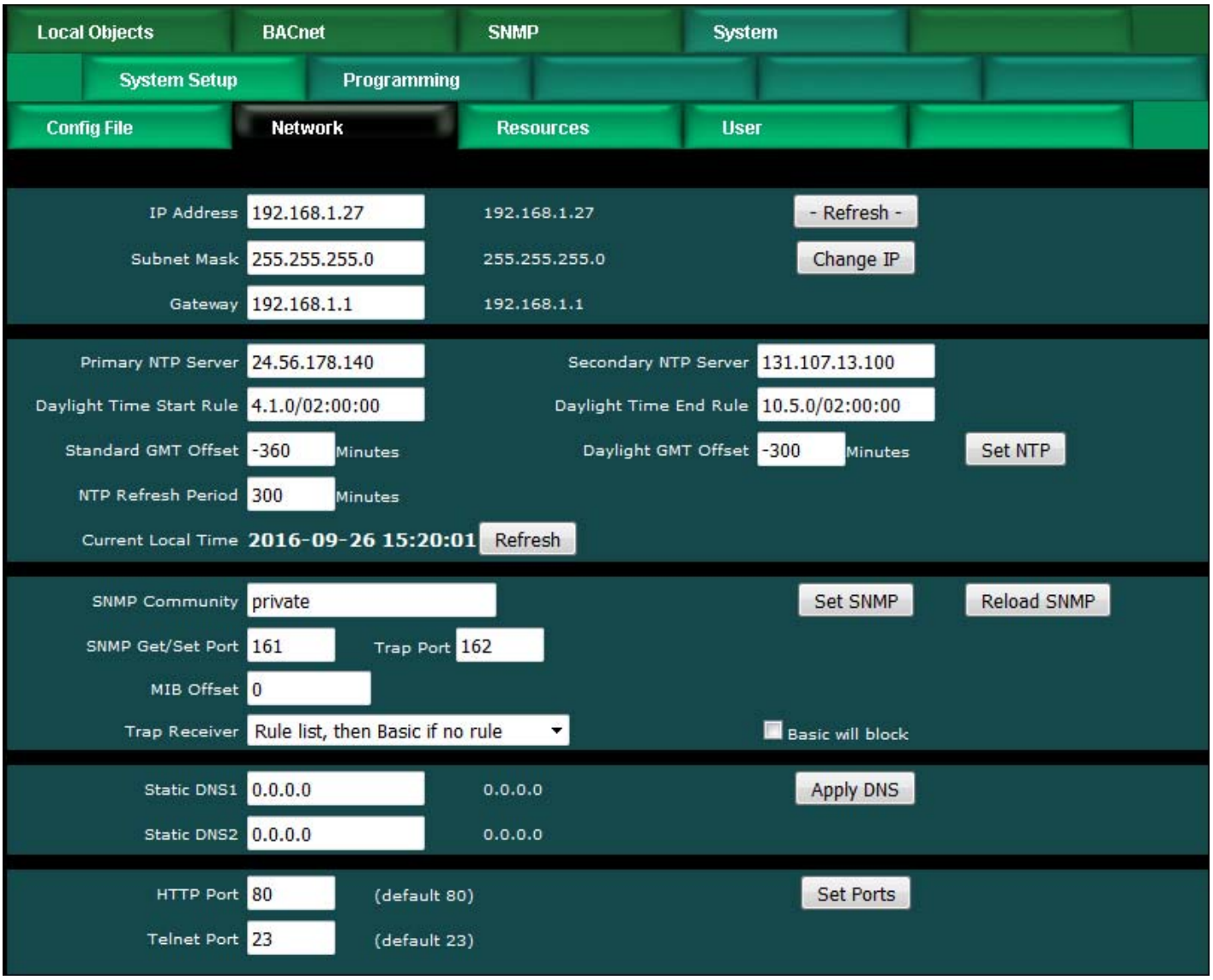
To upload a configuration file from your PC to the gateway, use the Browse button to find the file on your PC, open the file in the PC's file dialog box, and then click Upload. Saving the configuration file to your PC, and then uploading on a different device, is a quick and easy way to configure two Babel Buster Pros the same way.

The other thing found on the Config File page is the means to restart the Babel Buster Pro without a power cycle. Check the restart confirmation box, and click Restart. After a couple of minutes, you will be able to resume browsing the web user interface.

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.

14.2 Network Configuration

The Network Configuration page is where you set the Babel Buster Pro's IP address as well as a few other important things.



To change the Babel Buster Pro's IP address, enter the IP address, subnet mask, and gateway IP, then click Change IP. The new IP address will not take effect until the device is restarted. The configure IP address is shown in the input window. The actual IP address currently in use is displayed next to the input window.

If you want the Babel Buster Pro to request a dynamic IP address from a DHCP server upon startup, enter 255.255.255.255 for IP address. This is the "flag" that the Babel Buster should use DHCP. However, if no DHCP server is present, the Babel Buster will not boot up. If you need to switch from DHCP back to static IP, you must first boot up with DHCP in order to make that change.

The Babel Buster Pro maintains time and date via SNTP services. If you are writing a Script Basic program that wants to know time or date, you will need to provide NTP server addresses, set the daylight savings rules, and provide GMT offset.

To set up SNTP, enter a primary and secondary IP address of NTP servers, such as 24.56.178.140 for wwv.nist.gov (go to http://tf.nist.gov/tf-cgi/servers.cgi to find more). Enter daylight start/end rules, and offset in minutes from GMT for both

standard and daylight time. Offset is a negative number in the western hemisphere. Enter an NTP update time in minutes. Do not set NTP to update too frequently or you risk being denied service by the NTP server. Click the Set NTP button after all settings have been made. The Flash update will take several seconds. You will need to restart the gateway before the new settings will take effect.

Daylight savings time start/end rules consist of "date/time" where the date (m.n.d) indicates the day when summer time starts or ends, and time (hour:min:sec) is the current local time when summer time starts/ends. The date portion of the rule is formatted as follows:

- m indicates the month (1 <= m <= 12)
- n indicates which week of the month (1 <= n <= 5). 5 = the last week in the month.
- d indicates what day of the week (0 <= d <= 6). 0 = Sunday

For example: Start "4.1.0/02:00:00", end "10.5.0/02:00:00" means summer time starts at 2am on the first Sunday in April and ends at 2am on last Sunday in October. That was the old US rule. The new US rule is start "3.2.0/02:00:00" and end "11.1.0/02:00:00", which is start at 2am on the second Sunday in March, end at 2am on the first Sunday in November.

The SNMP section is used to enter the SNMP community that will be used by the Babel Buster Pro as its own. Community strings needed to access other devices are provided in the SNMP client configuration. The ports are normally 161 for SNMP Get/Set and 162 for SNMP traps.

The MIB offset lets you effectively move the entire MIB. This is required if more than one Babel Buster Pro are going to be used on the same network but their configuration is different. Most SNMP managers do not know how to deal with MIBs that have the same set of OIDs but which mean different things in different devices using the "same" MIB. If the offset is changed, you will need to click Reload SNMP to cause it to take effect.

To save changes made on this page, click Set SNMP. To reload the Babel Buster Pro's MIB tables because they have changed (on other configuration pages) click the Reload SNMP button. The Reload SNMP button is also replicated on those pages where you would make changes that will require reloading.

DNS servers will need to be provided if you reference any devices in your configuration by domain name rather than static IP address. Enter the static IP addresses of the DNS servers and click Apply DNS.

The HTTP port for browsing the user interface can be moved away from the default HTTP port 80. Select a different port, click Set Ports, and then restart the gateway to make that new port take effect. Don't forget to append the port number to the gateway's IP address when attempting to browse the web user interface if it has been moved from port 80.

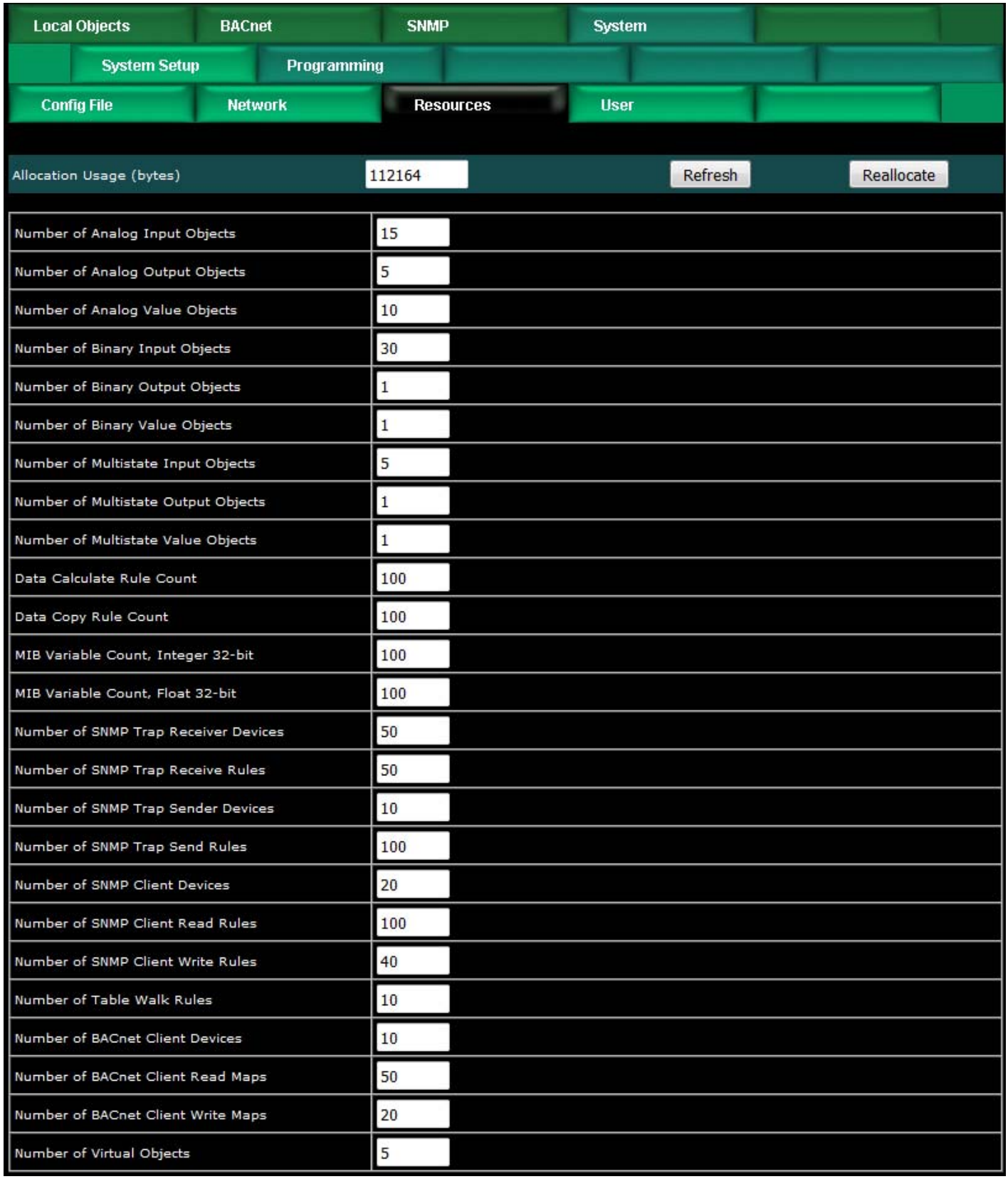
Telnet is disabled by default (port set to zero). Enable it by setting the port that Telnet should use, then restart the device to activate Telnet. The only meaningful thing you can do via Telnet if needed is to restore the MAC address in the unlikely event it has become corrupted.

The one other SNMP setting that was not mentioned above was enabling the trap receiver. Do this in conjunction with configuring the trap receiver (see section 12). The options for enabling the trap receiver are illustrated above. The description of what these mean is found in section 12.

14.3 Resource Allocation

Historically, Control Solutions gateways had a fixed set of BACnet objects and other resources to work with. Invariably, there was always somebody that wanted less of this and more of that. Therefore, while there are still maximums imposed, you can now shift resources around as needed. An example is shown below.

IMPORTANT: To change allocation values, enter the new values, then click Reallocate. If you click Refresh instead, the original numbers will be restored. To complete the reallocation process, after entering new numbers and clicking Reallocate, you must go to the Config File page, Save your configuration, and then click Load to reload the file just saved. This reloading of the configuration will reconfigure the device using the new allocation numbers.



There is no magic formula for determining the "right" size configuration. All resources are allocated from "free memory". This free memory is also used by Script Basic for its variables. It is also used by the SNMP engine for temporary variable buffering while an external SNMP client/manager is sending Get/Set requests to the Babel Buster Pro. With the numbers of objects and various rules provided by default in older Control

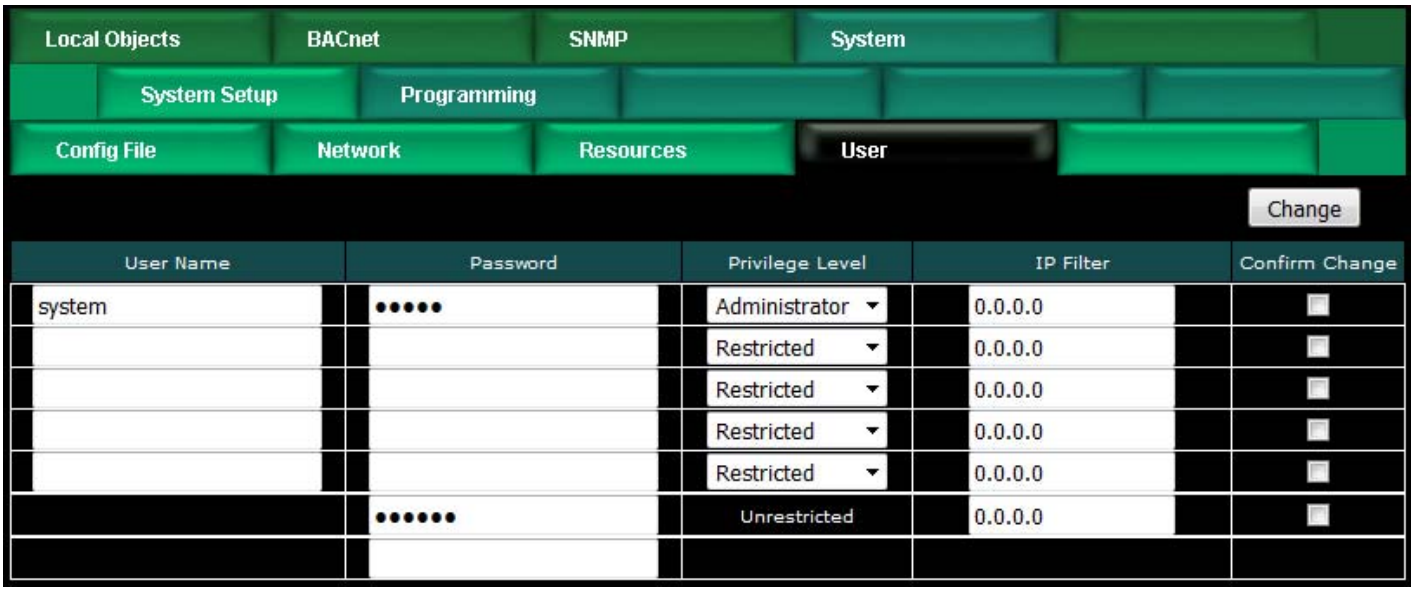
Solutions gateways, the remaining memory allowed successful MIB walks of up to approximately 400 MIB variables. If you are starting to run short on memory, a failed MIB walk will usually be the first place you see this.

14.4 User Login Passwords

There are two default logins provided initially. They are username "root" with password "buster", and username "system" with password "admin". When logged in as anybody other than "root", you can only change your own password. But if you are logged in as the root user, you can change all passwords and add user names.

The privilege level Administrator lets that user see and change anything. The privilege level Maintenance allows the user to log in and see (and change) values in the local objects via the Local Objects page, but cannot access any other pages. The Restricted level has no meaning in the Babel Buster Pro (other than block access to everything) since it does not operate as a user defined web server.

You also have the option of IP filtering. If set, then the user can only access Babel Buster Pro's web pages from that IP address. Leave set to 0.0.0.0 to disable filtering.



15. Trouble Shooting

15.1 BACnet MS/TP Trouble Shooting

15.1.1 Most Common Problems

The most common problems with completely failing to communicate on the MS/TP link are either RS-485 wiring, or port settings. Any incorrect setting of baud rate, Max Masters, or MAC address will result in communication failure - in some cases not just of the BBPRO-V230, but the entire network when you connect the BBPRO-V230.

If you connect two MS/TP devices with mismatched baud rates, one will chatter constantly polling for masters while the other is silent indefinitely. MS/TP devices are required to be silent until their chance to talk is heard. It is only a race to see who starts first and that one will be the one to chatter constantly. Since the silent one has no chance of hearing its chance to talk when running at a different baud rate, it will be silent forever.

Mismatched Max Master settings can result in erratic behavior on the network. The Max Master setting tells a device what the highest expected MAC address should be. A mismatched setting can result in some devices never getting polled. A safe setting, and one usually used as the default, is 127 - this is the highest possible MAC address, and nobody will ever get left out this way.

The only incorrect setting for a MAC address is to use one that is already in use by another device. Now two devices will attempt to respond at the same time when that address is polled, and neither reply is likely to be received correctly.

When it comes to wiring, remember that RS-458 is NOT truly a 2-wire interface as it is commonly referred to. Refer to the RS-485 FAQ under Support at csimn.com if you have questions or concerns about wiring. Refer also to Appendix A in this user guide for more wiring information.

Once the BBPRO-V230 is communicating MS/TP, then next area for possible concern is with the BACnet client. If the gateway is supposed to be polling other MS/TP devices, but the data does not appear correct, the first thing to check is the reliability code. Any reliability code other than zero is a problem. Refer to the list at the bottom of any of the Data Objects pages for explanation of the non-zero codes. If the reliability code indicates that an error was returned by the server (meaning the other BACnet device

you are trying to query), then refer to the BACnet Diagnostics page for additional error information.

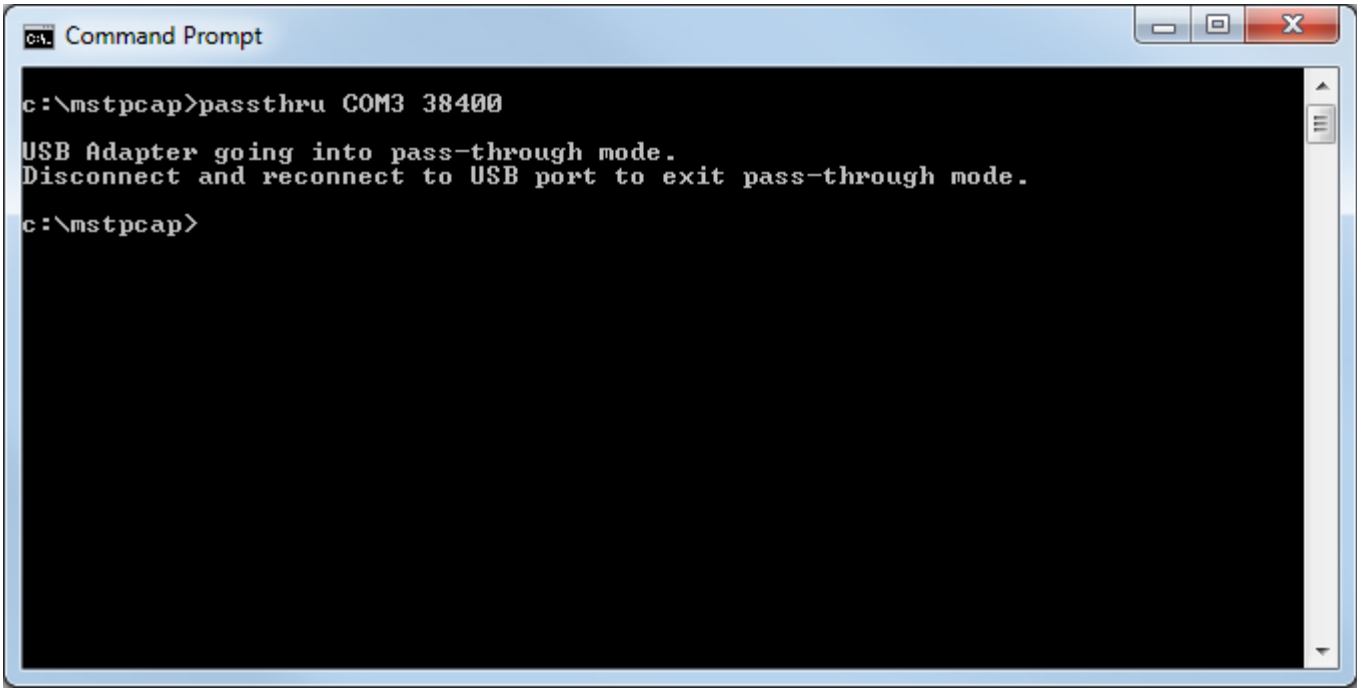
15.1.2 Using Wireshark for MS/TP

Problems are sometimes not obvious and you will want to see what is actually going out over the network. Most people are already with using Wireshark to capture network traffic on Ethernet, but you can also use Wireshark to analyze data captured on MS/TP. The capture is not live like it is for Ethernet, but analysis with Wireshark can be very helpful.

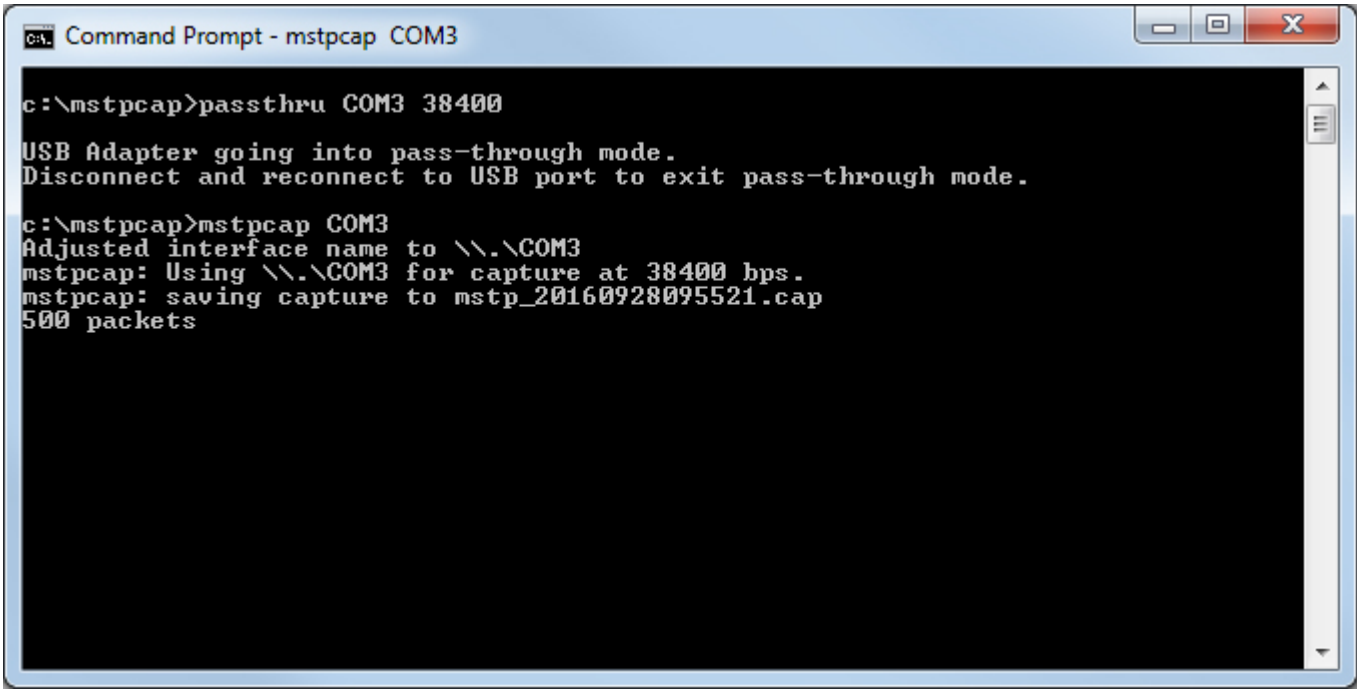
Control Solutions has created an MS/TP data capture utility that works in conjunction with the MTX002 MS/TP to USB adapter. This is not a generic RS-485 adapter. The MTX002 is an intelligent device that is itself an MS/TP device. A special driver has been included in the data capture utility to recognize MS/TP packets sent via USB by the MTX002.

Start by downloading and installing the USB driver for the MTX002. Do not plug in the MTX002 until you have installed the correct USB driver. The driver installation package is found on the product page for the MTX002 at csimn.com.

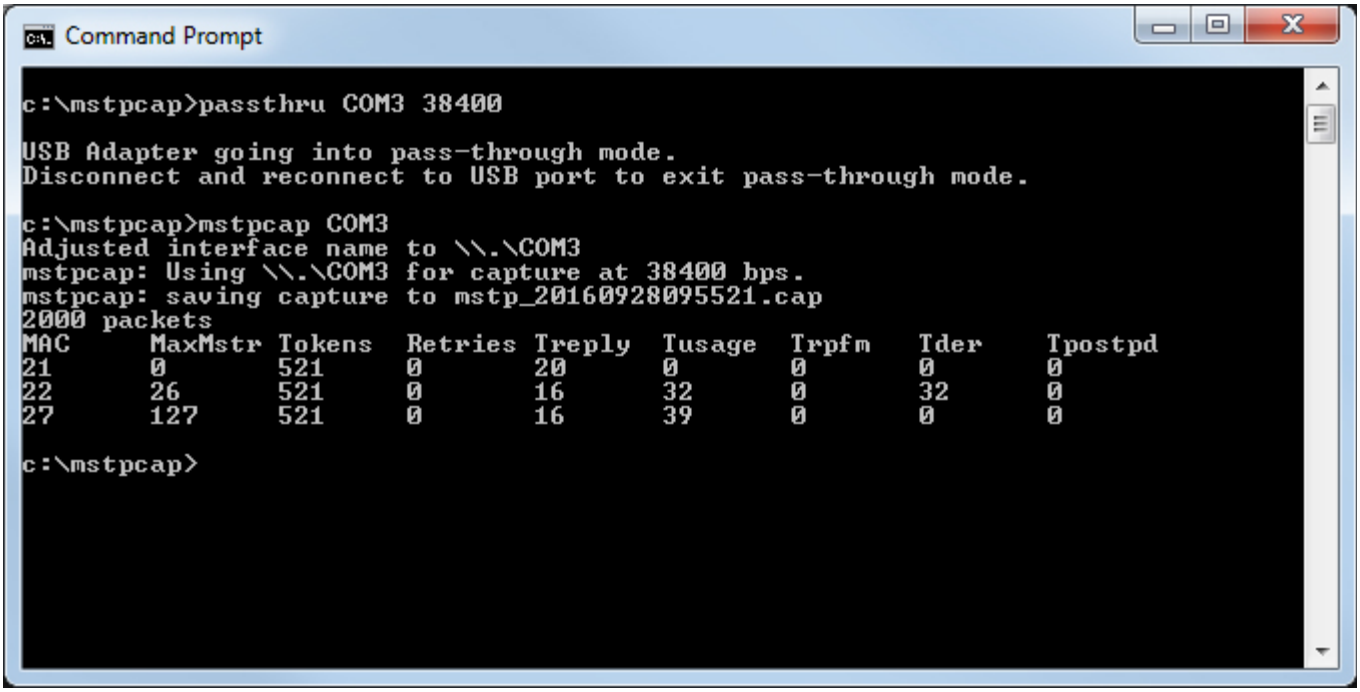
Download the MS/TP packet capture utiility from the Tool Links page at csimn.com. To run the capture utility, start by putting the MTX002 in pass-through mode. Refer to your PC's device manager to see where the MTX002 was installed, and refer to that COM port in the passthru command. Select the baud rate that matches your network.



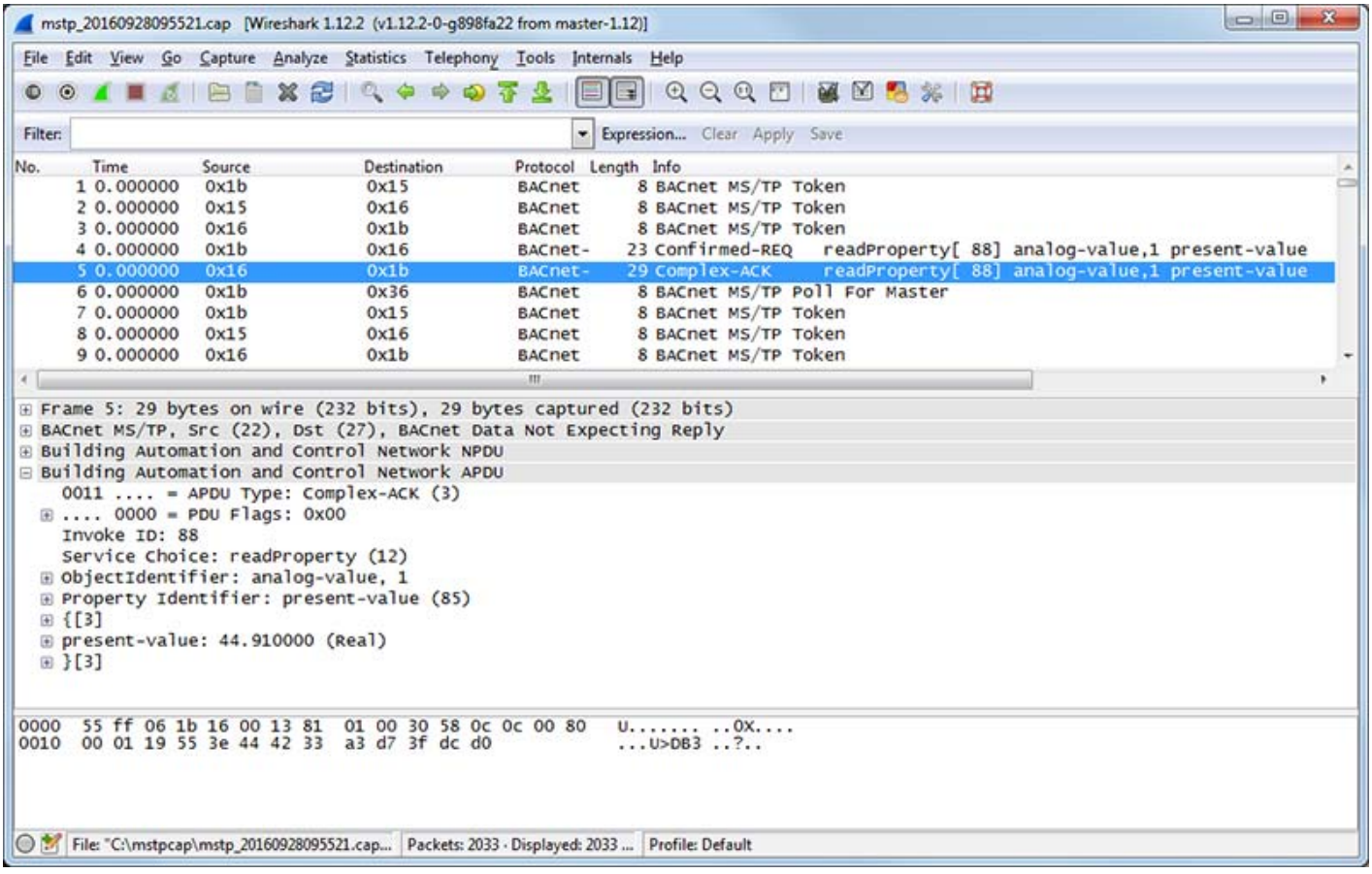
Now run mstpcap referring to the COM port that the MTX002 is on. Type Ctrl-C to stop capture.



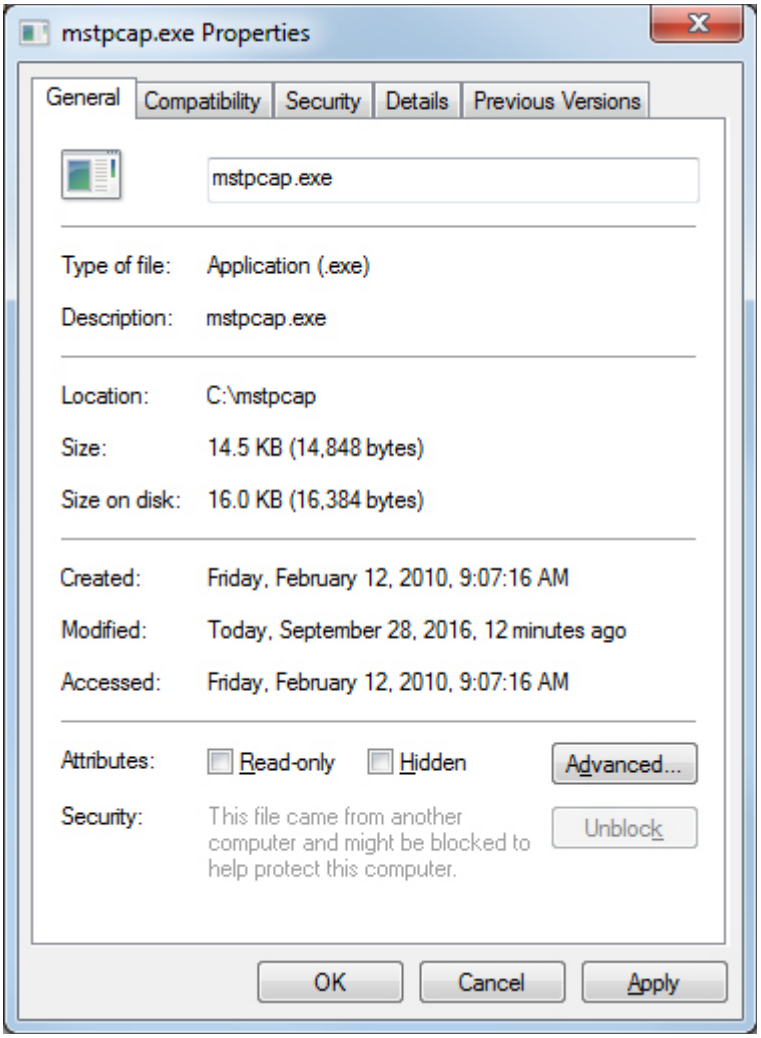
When capture is stopped, you will get the capture summary that looks something like the illustration below. Note the file name that starts with "mstp_" and ends with .cap. Find this file and double click it (assuming you have Wireshark installed on your PC).



Double clicking the .cap file created will automatically open it in Wireshark and display packets as illustrated below.



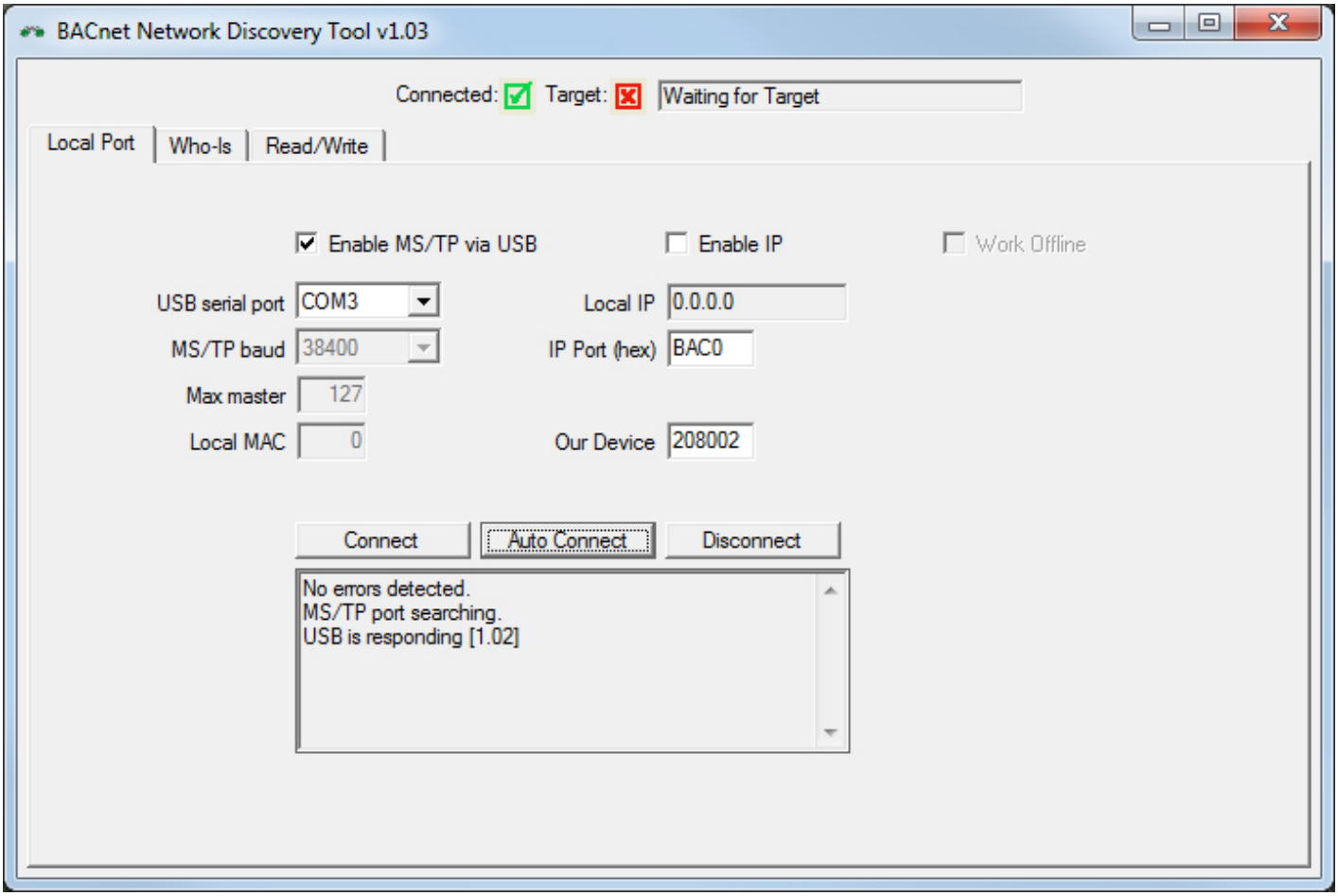
If mstpcap says it saved a file but you cannot find it, check to see that mstpcap.exe is not blocked. It will appear to run but not be allowed to save a file on your PC if blocked. Click Unblock if necessary.



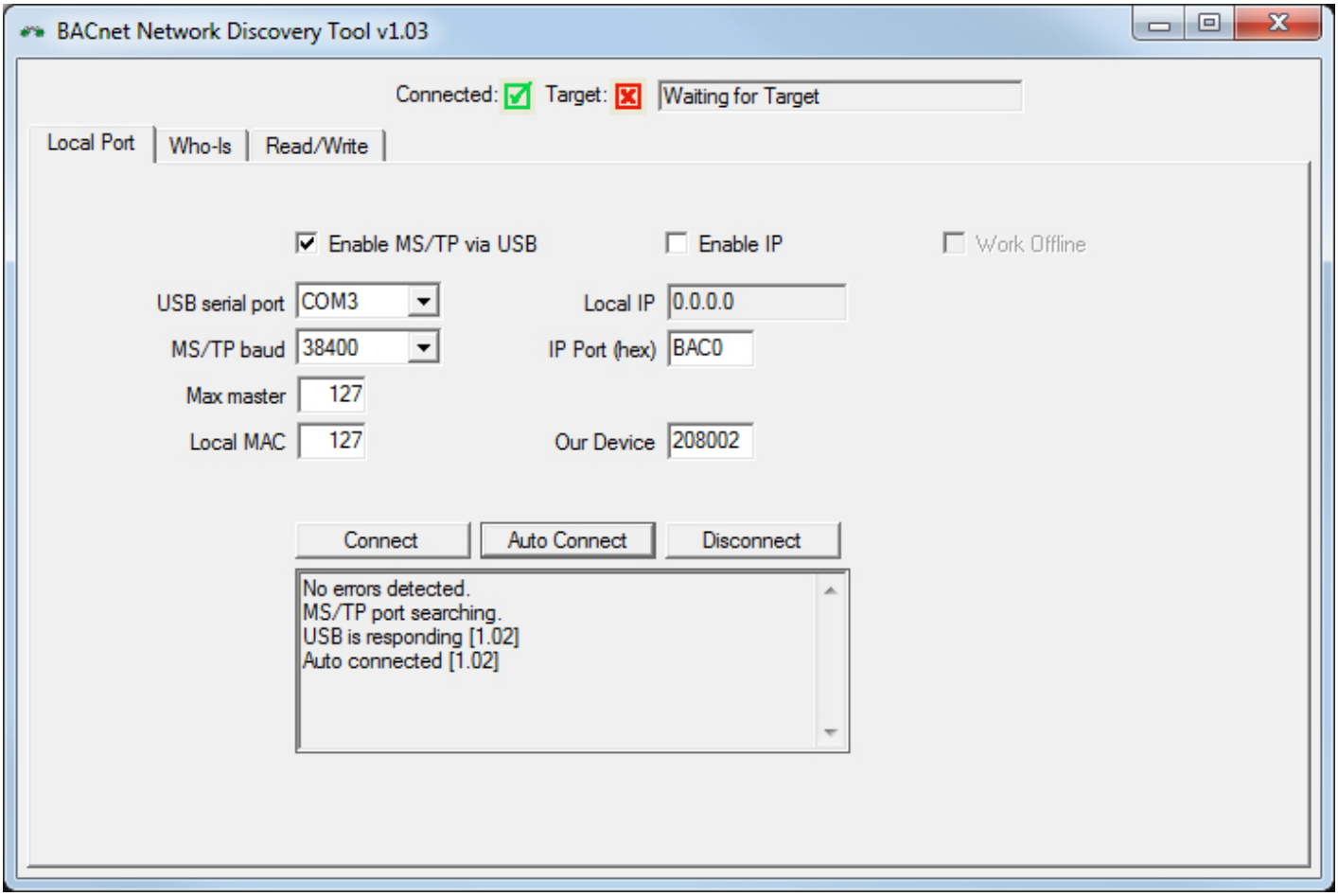
15.1.3 Using Network Discovery Tool

Control Solutions has created a Network Discovery Tool to perform simple diagnostics on BACnet devices and networks. It works with MS/TP using the MTX002 talked about above.

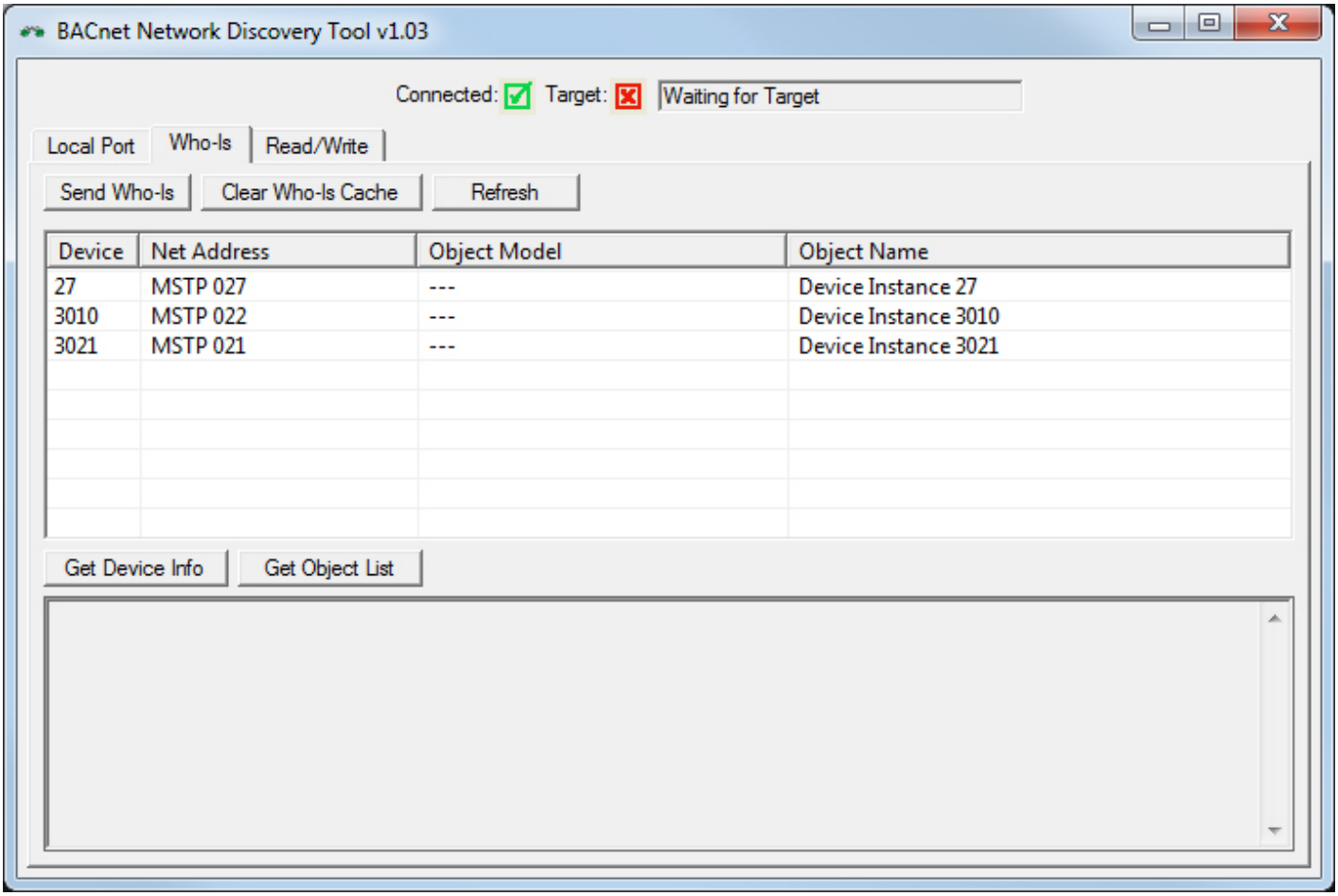
If you know the network's baud rate and Max Master setting, select those, and simply click Connect. If you are uncertain or don't even know the baud rate, select the COM port the MTX002 is on, click Enable MS/TP via USB, and then click Auto Connect.



Auto Connect will take a little while to complete. First it analyzes the baud rate. Once locked onto baud rate, it then listens to the network for a while to learn what the Max Master setting is (which should be identical in all devices on the network). It will also check to see which MAC addresses are in use, and set its own MAC to something not in use starting from the top of the range.

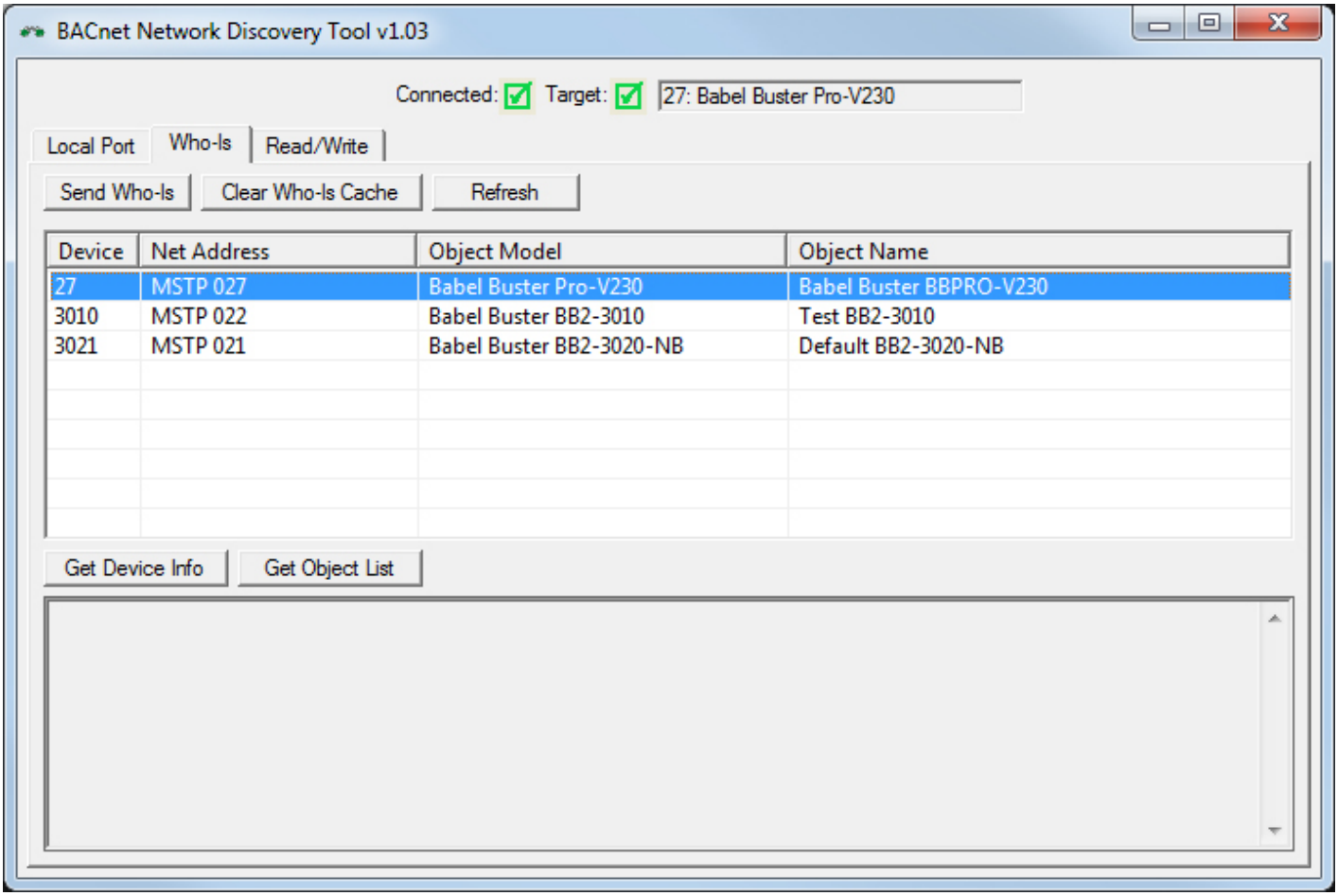


Once connected, go to the Who-Is page. Usually, by the time you get there, the results of the first automatic Who-Is are already displayed.



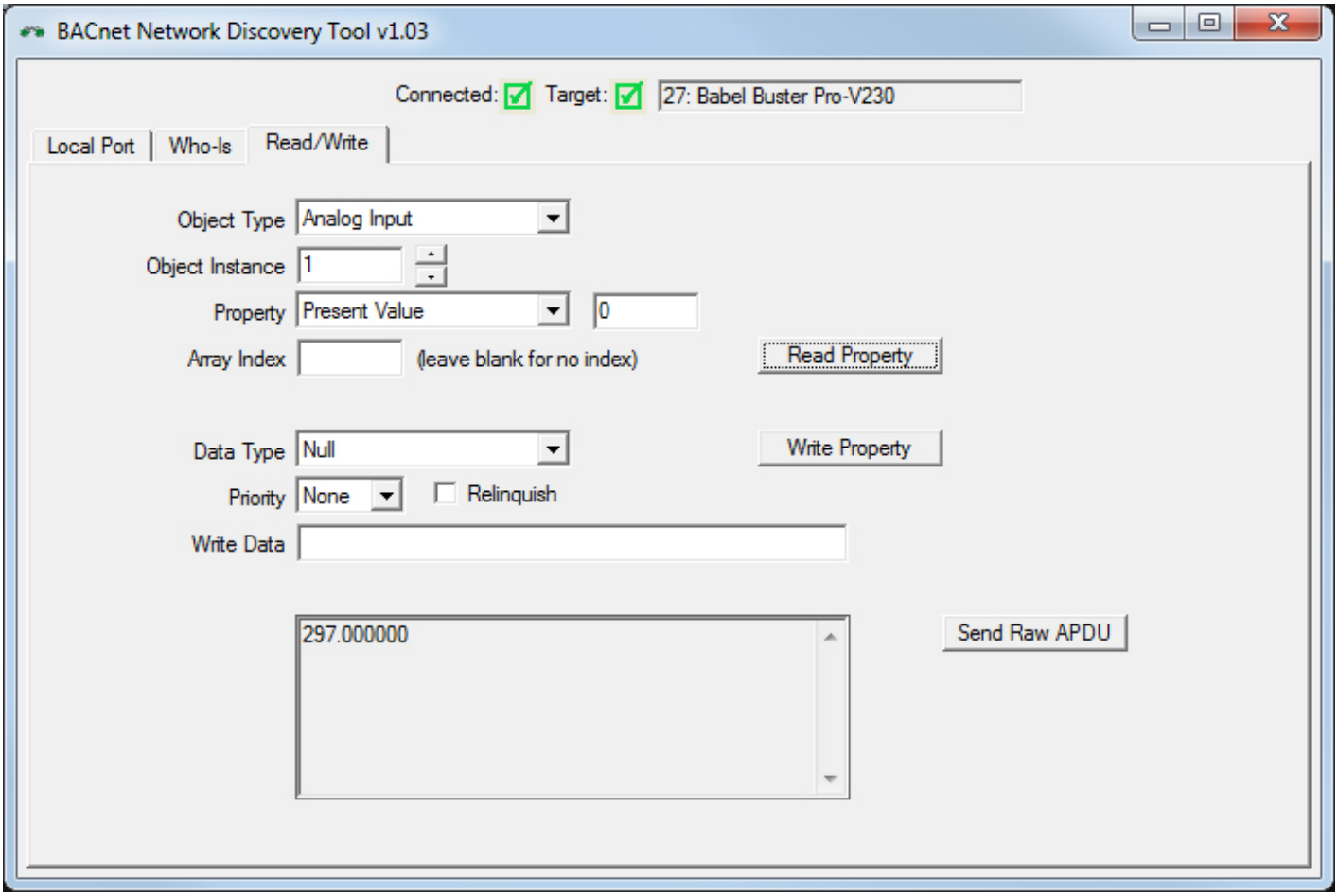
Click the Refresh button to cause the discovery tool to query every responding device to read object model and device oblect name from each of them.

Double click on the device you with to query further. It will now appear as the Target.



You can read and write properties in any of the standard objects typically used in any Control Solutions device and in most other devices. Select object type, instance, and property to read data by clicking the Read Property button.

In addition to selecting the same parameters you would for reading, select data type, priority if writing to a commandable object, and data value to write that property by clicking the Write Property button.



15.2 BACnet IP Trouble Shooting

15.2.1 Most Common Problems

BACnet IP is typically easier to get running than MS/TP just because Ethernet is pretty straight forward. The most frequent problem is "no response" or timeout. The most common cause of this problem for BACnet IP is a network configuration problem, such as incorrect IP address or IP address that cannot be reached as configured. The problem sometimes lies outside the Babel Buster and may require consulting with the IT personnel responsible for the network if on a large network.

The subnet mask determines what part of the IP address constitutes the domain, and all devices on the same network must be on the same domain before they can communicate.

Obviously two devices being assigned the same IP address is going to cause trouble. If you can communicate at all with a device having a duplicate IP address, it will be intermittent, and potentially erratic as the other device having the same IP address may be responding to your queries.

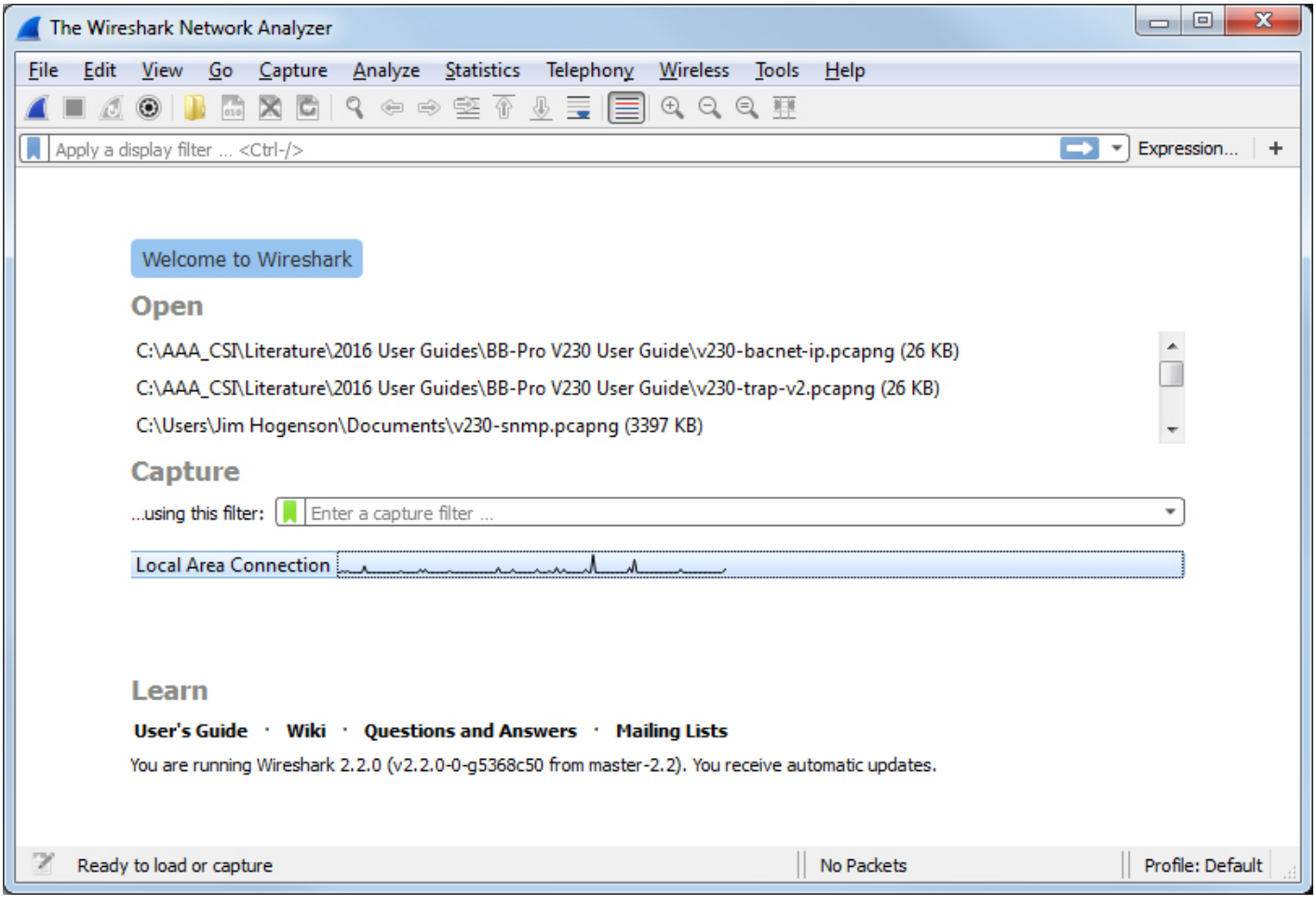
If you are connecting via one or more routers, then everything that applies to routing issues will apply to your device. A complete discussion of NAT routing, BACnet routing, etc, is beyond the scope of this document - you should refer these questions to your

IT administrator when applicable.

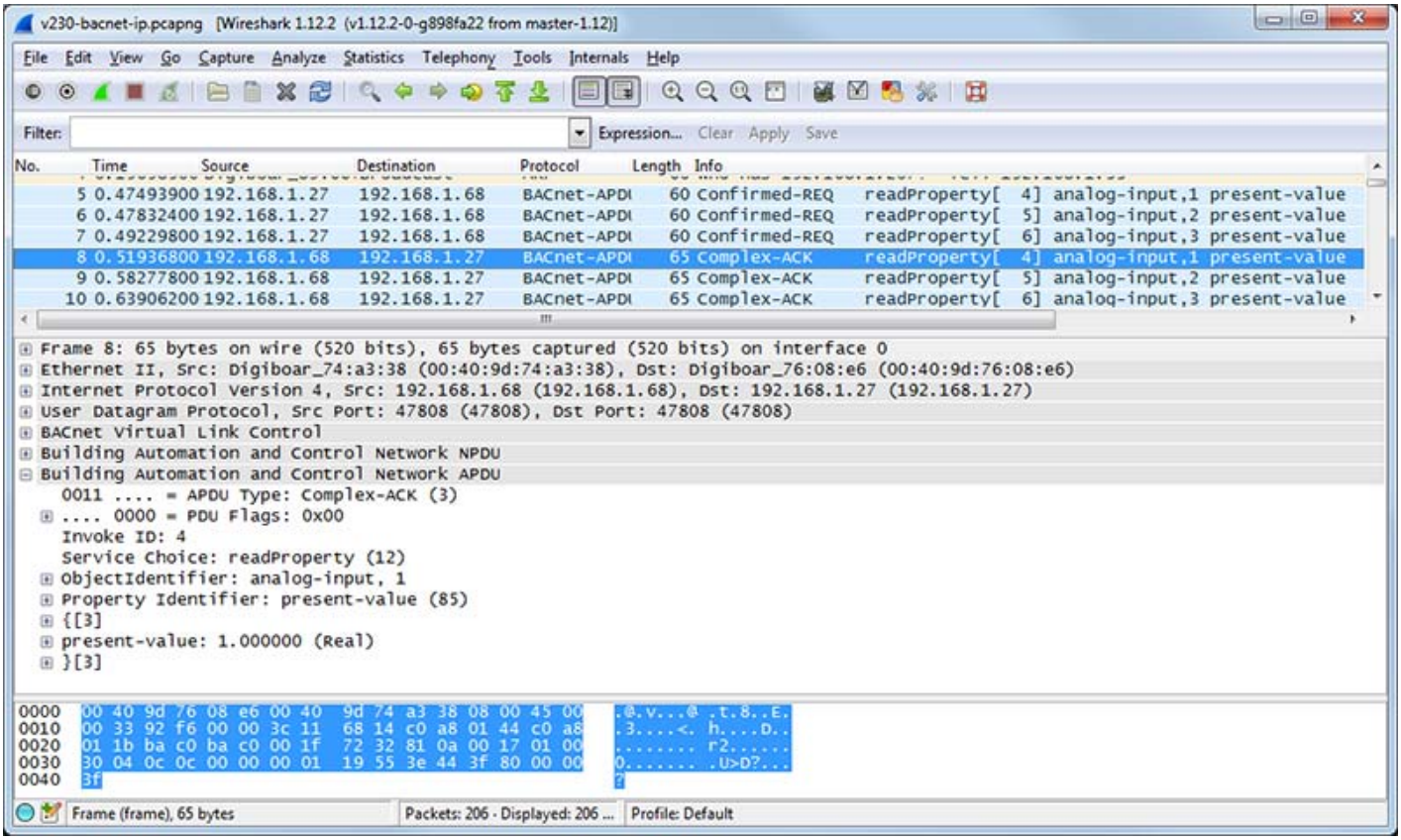
Once the BBPRO-V230 is communicating BACnet IP, then next area for possible concern is with the BACnet client. If the gateway is supposed to be polling other IP devices, but the data does not appear correct, the first thing to check is the reliability code. Any reliability code other than zero is a problem. Refer to the list at the bottom of any of the Data Objects pages for explanation of the non-zero codes. If the reliability code indicates that an error was returned by the server (meaning the other BACnet device you are trying to query), then refer to the BACnet Diagnostics page for additional error information.

15.2.2 Using Wireshark

One of the most useful tools for diagnosing BACnet IP problems is Wireshark. You can get a free copy at www.wireshark.org. Additional important information about Wireshark can be found in Appendix G of this user guide. When you start Wireshark, the startup screen appears as follows (as of this writing). Click on Local Area Connection to begin captureing traffic.



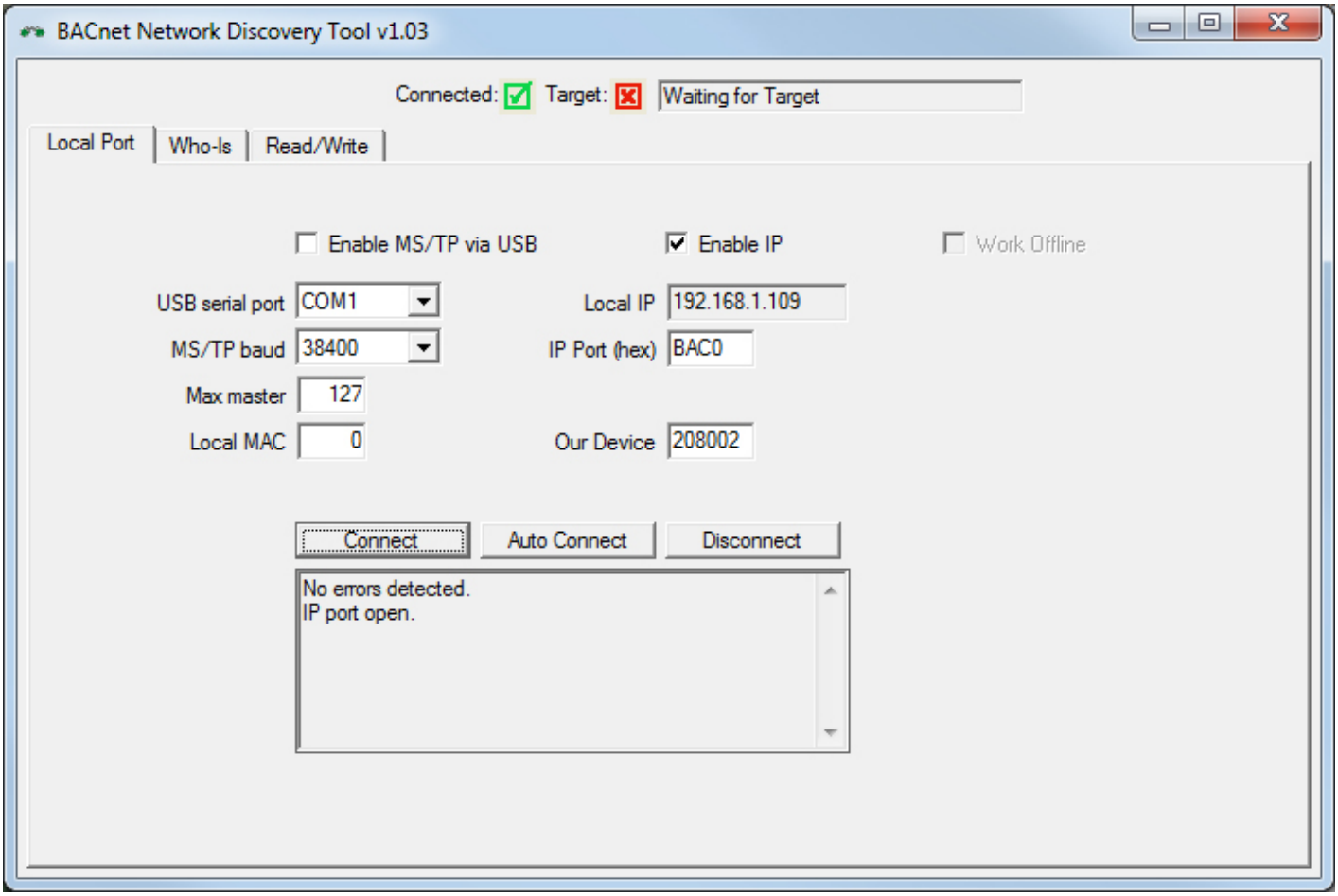
Network packet capture will be live with Ethernet. Click on any packet of interest, and you can expand the tree structure to see the full content of the packet. Wireshark includes complete decoding for BACnet protocol - a very useful feature.



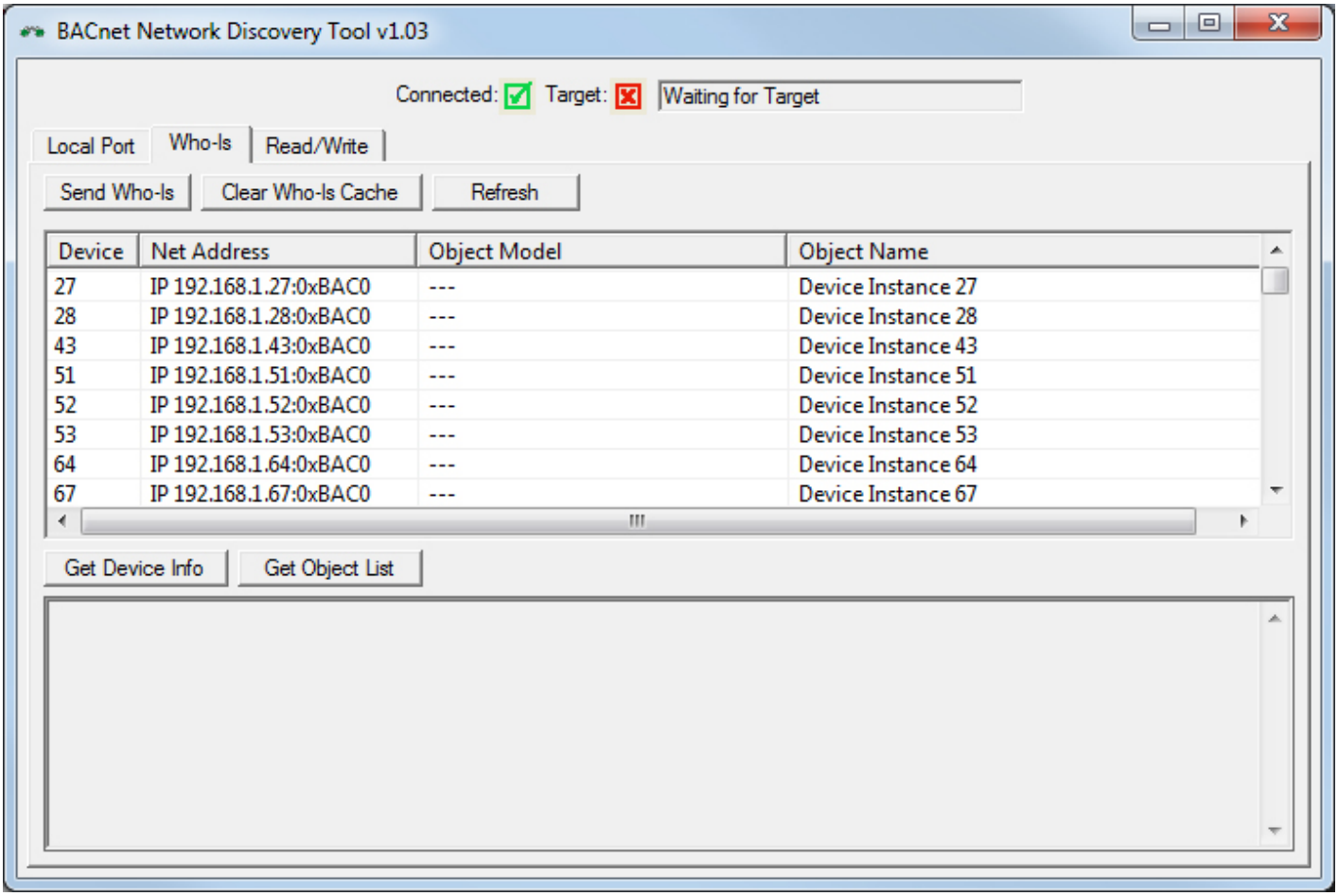
15.2.3 Using Network Discovery Tool

Control Solutions has created a Network Discovery Tool to perform simple diagnostics on BACnet devices and networks. It works with BACnet IP using your PC's Ethernet connection - assuming your PC is connected to the BACnet IP network.

Simple check Enable IP, and click Connect to begin.

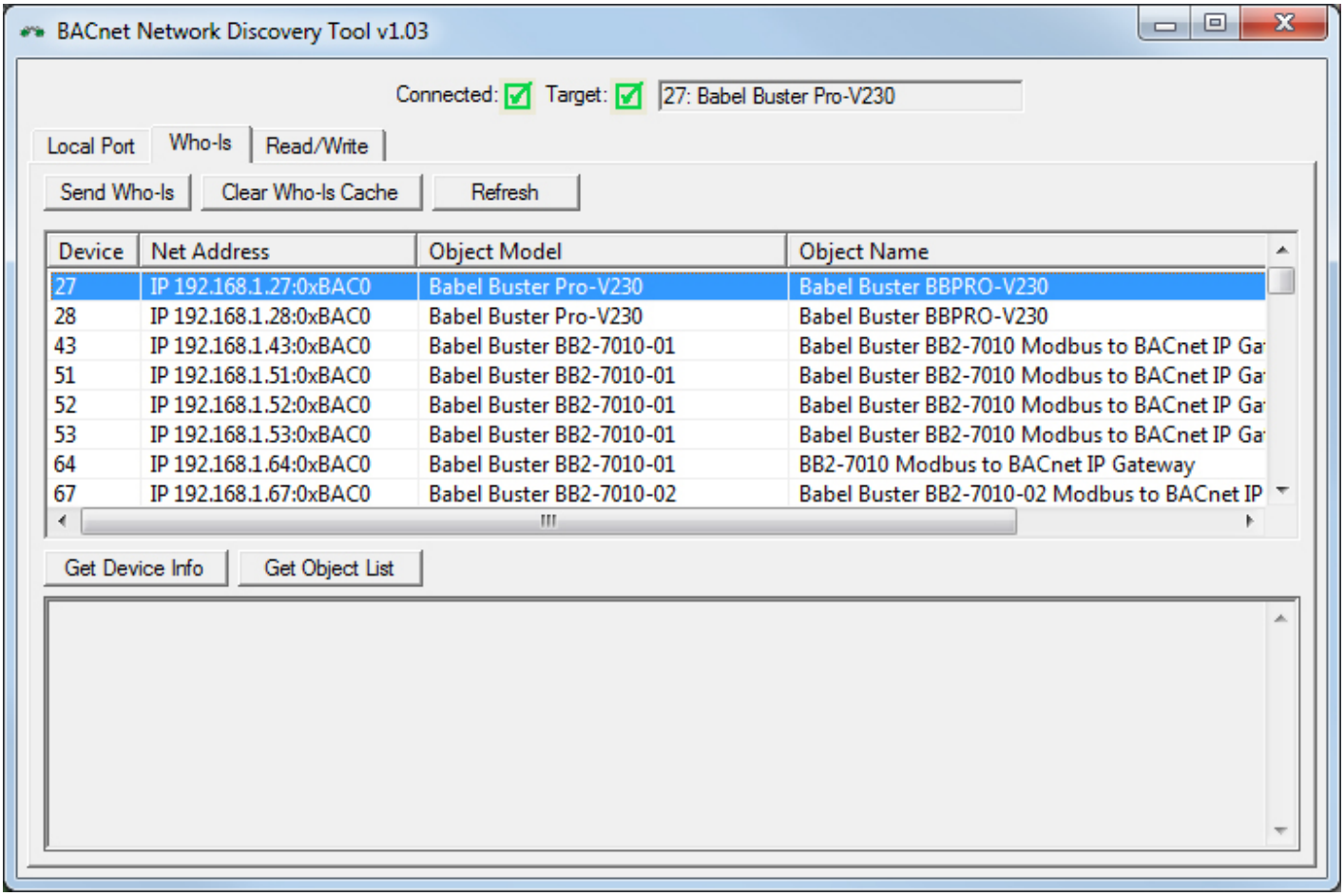


Once connected, go to the Who-Is page. Usually, by the time you get there, the results of the first automatic Who-Is are already displayed.

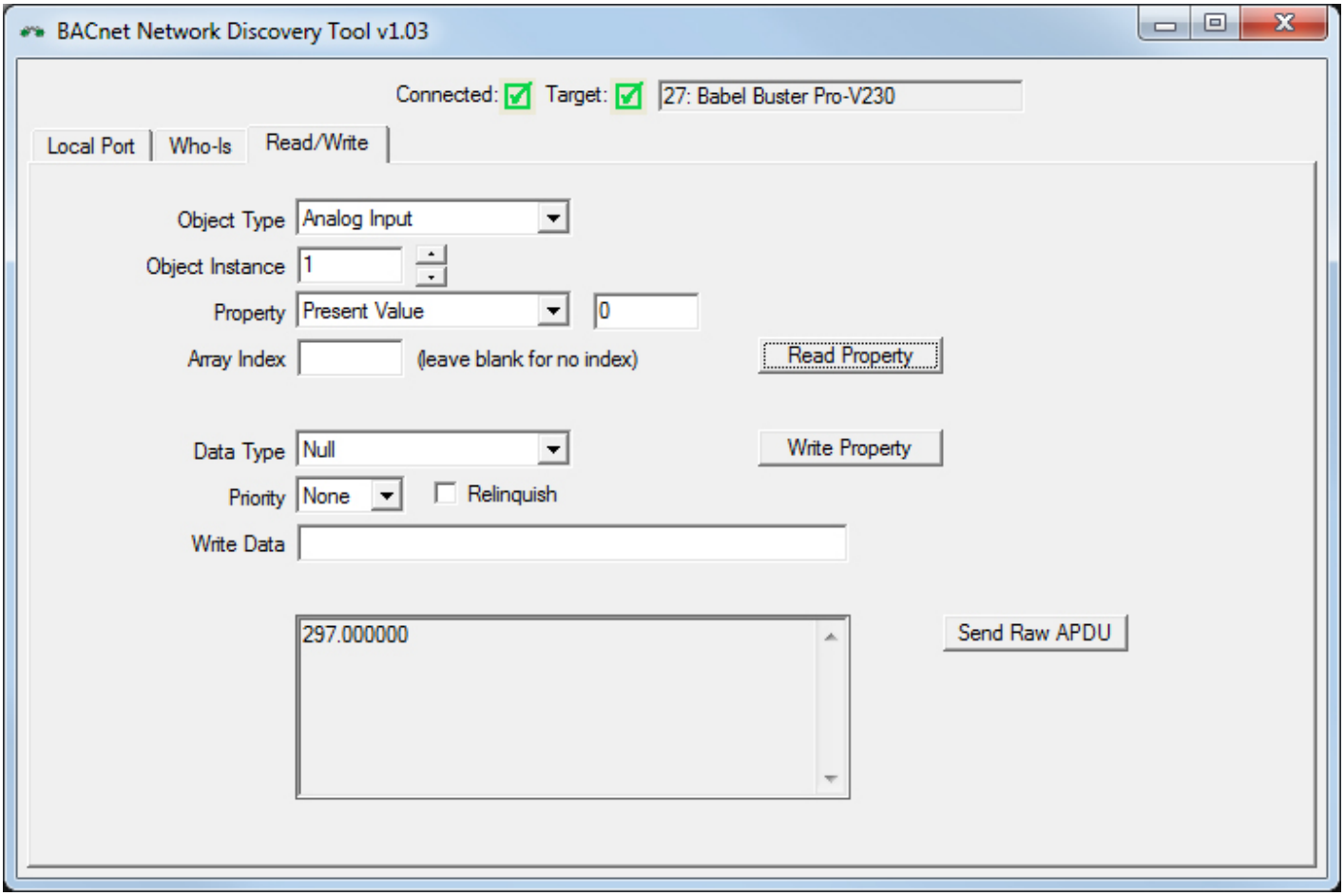


Click the Refresh button to cause the discovery tool to query every responding device to read object model and device oblect name from each of them.

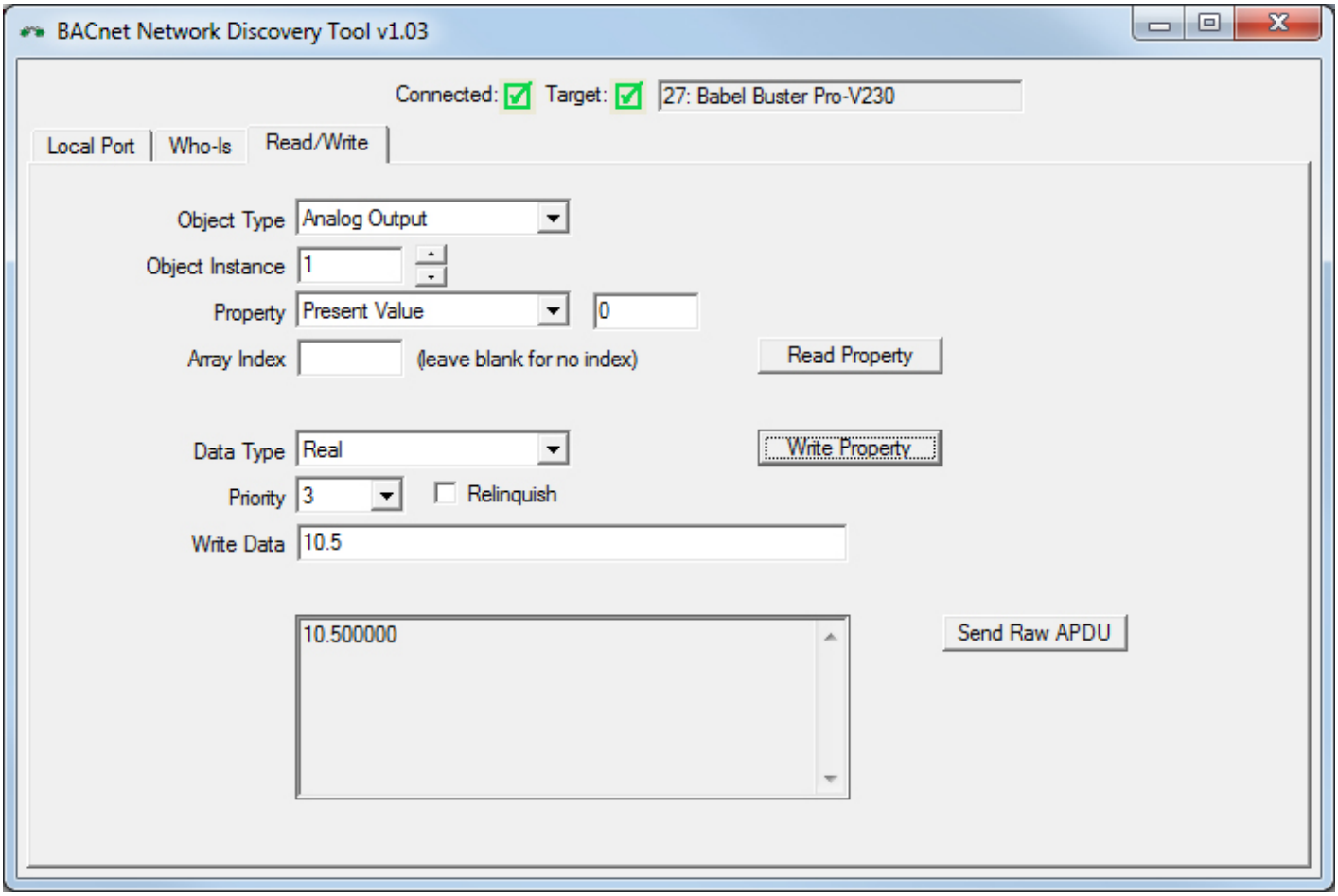
Double click on the device you with to query further. It will now appear as the Target.



You can read and write properties in any of the standard objects typically used in any Control Solutions device and in most other devices. Select object type, instance, and property to read data by clicking the Read Property button.



In addition to selecting the same parameters you would for reading, select data type, priority if writing to a commandable object, and data value to write that property by clicking the Write Property button.



15.3 SNMP Trouble Shooting

15.3.1 Most Common Problems

Assuming you have IP addresses configured correctly and the SNMP ports are open through any routers and firewalls between devices, the most common cause of not communicating is a mismatching community string. This is sort of like a password. Without the correct community string, most devices will simply ignore the request, making it look as if there is no connection when in fact there is nothing wrong with the connection.

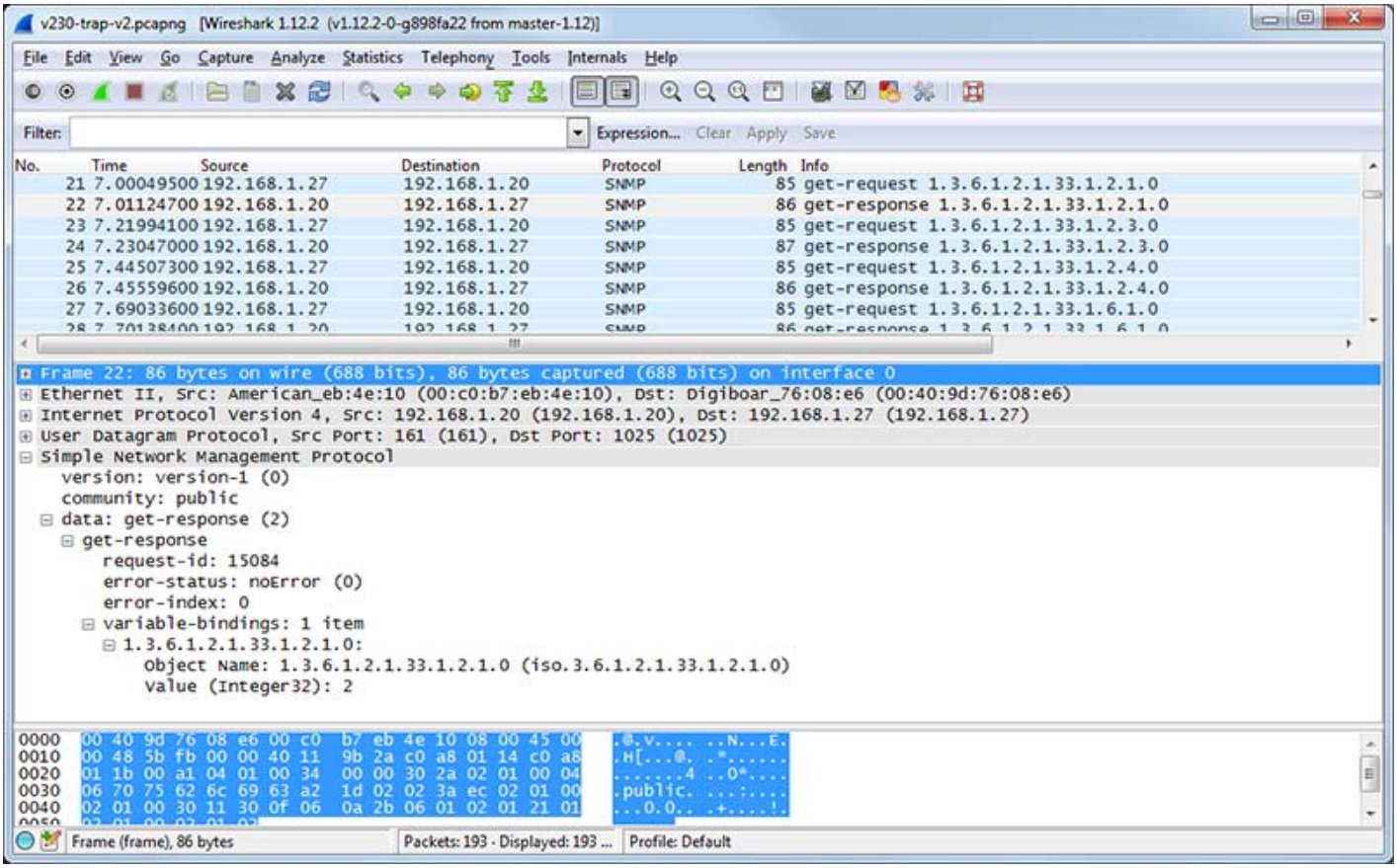
Another common oversight is that when adding local objects to the local MIB, you need to click the Reload SNMP button at the bottom of the Local MIB pages to cause SNMP to reload its internal tables with the new configuration you just entered. The Local MIB pages are effectively a list of instructions for loading the SNMP MIB, but the MIB is not automatically rebuilt every time you add another line to the Local MIB pages. To reload the MIB according to the list of instructions you provided on the Local MIB web pages, you need to click Reload SNMP. (Note: The Reload SNMP button is found on multiple pages, but they all perform the same function and any of the Reload SNMP buttons will reload all branches of the MIB.)

If you are working with the SNMP Client, working to Get or Set MIB variables in other SNMP devices, the list of possible errors is found in section 9.7 of this user guide. The

most common problem is no response due to incorrect IP address, unable to reach the IP address via the network as configured, or mismatching community string. The next most common problem is "unable to interpret application data". This means the SNMP device did respond, but data could not be interpreted - meaning the data was not numeric in nature and could not be converted to anything meaningful to BACnet. If this happens, you have the option of customizing the data interpretation as talked about in section 9.6 and illustrated in section 13.7.

15.3.2 Using Wireshark

One of the most useful tools in trouble shooting SNMP is Wireshark. The general discussion about Wireshark above and in Appendix G will not be repeated here. For reference, the following screen shot illustrates Wireshark decoding an SNMP packet.



15.3.3 Using ManageEngine MIB Browser

The other most useful tool in trouble shooting SNMP is a MIB browser.

A variety of tools are available for browsing an SNMP MIB and receiving SNMP Traps. The tool used in the examples in this user guide is the iReasoning MIB Browser. Refer to the Tools section under Support at csimn.com for more information about SNMP tools.

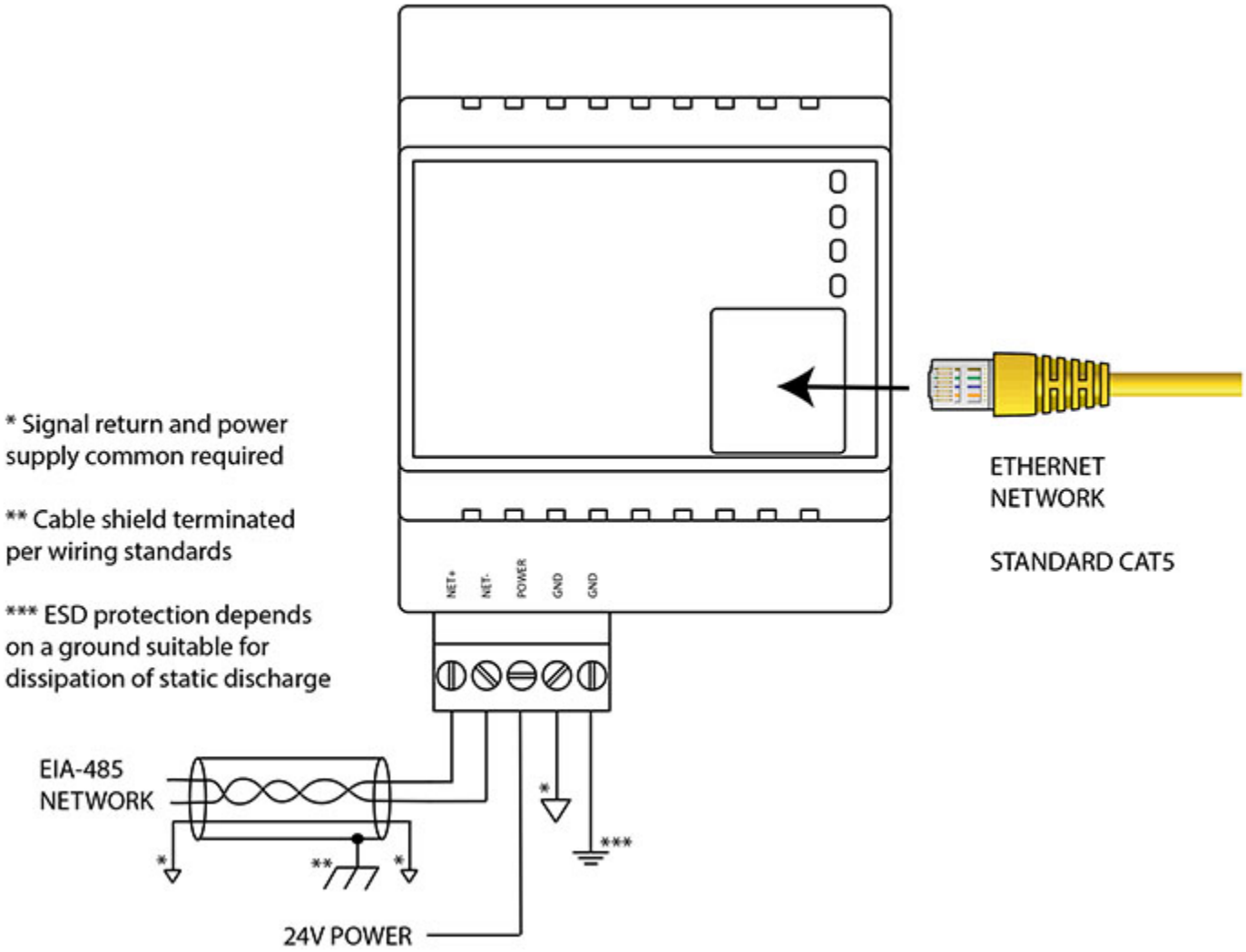
The discussion in previous sections about using the MIB browser to trouble shoot

SNMP will not be repeated here. Refer to Section 8.3 regarding MIB browsing, and Section 11.5 on trouble shooting SNMP Traps.

Appendix A Hardware Details

A.1 Wiring

Wiring for the Babel Buster BBPRO-V230 is illustrated below.



Wire the gateway as illustrated. Follow all conventional standards for wiring of EIA-485 networks when connecting the BACnet MS/TP EIA-485 (RS485) network. This includes use and termination of shield, termination of the network, and grounding.

IMPORTANT: Although EIA-485 (RS485) is thought of as a 2-wire network, you MUST include a third conductor connected to GND or common at each device so that all

devices are operating at close to the same ground potential. Proper grounding of equipment should ensure proper operation without the third conductor; however, proper grounding often cannot be relied upon. If large common mode voltages are present, you may even need to insert optically isolated repeaters between EIA-485 devices.

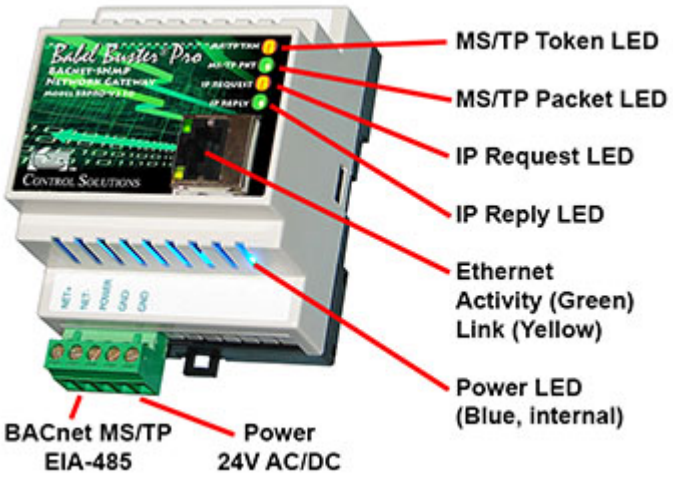
Use standard CAT5 cables for Ethernet connections. Use control wire as applicable for local electrical codes for connecting the 24V (AC or DC) power supply.

Note that in addition to connecting power supply common to a GND terminal, you must also connect a GND terminal to earth ground in order to ensure proper ESD protection.

BBPRO-V230-SP232: The standard BBPRO-V230 BACnet MS/TP port uses RS-485. The RS-232 version, -SP option, replaces the RS-485 transceiver with an RS-232 transceiver. The NET+/NET- terminals are replaced by TXD and RXD on the -SP232 version. TXD is data out from the BBPRO-V230-SP232, and RXD is data in to the gateway. Hardware handshake is not supported.

A.2 Front Panel LED Indicators

Power-up LED behavior: On power up, the column of LEDs above the RJ45 connector will all turn on yellow or red, followed by green. The MS/TP LEDs will exhibit their startup display immediately while the IP LEDs will exhibit their startup display after a server boot-up delay. Then they will proceed to indicate as normally defined for the indicators.



Babel Buster BBPRO-V230 MS/TP LEDs reflect MS/TP, the IP LEDs reflect BACnet IP traffic, and the Ethernet activity LED will indicate network traffic in general.

Babel Buster BBPRO-V230 LEDs indicate as follows (LEDs are bi-color):

MS/TP TKN (MS/TP Token)	Flashes green each time the MS/TP token is passed, and also flashes yellow when the gateway polls for a master.

	(Inactive on -SP version)
MS/TP PKT (MS/TP Packet)	Flashes green when a packet is sent or received and the packet is data other than token pass or poll for master. (Inactive on -SP version)
IP REQUEST	Flashes yellow when a packet is received addressed to this device (or broadcast), or when a packet is sent by the gateway.
IP REPLY	Flashes green when a good packet is received, flashes red when a received packet contains an error. Also flashes red if a client request times out.
Ethernet Activity	Green LED is on solid during portions of the boot-up process, and then flashes briefly when Ethernet network traffic is detected.
Ethernet Link	Yellow LED indicates an Ethernet link is present. This indicator will light if a link is present regardless of processor or network activity. If not lit, check network wiring.
Status	Blue LED (internal) on any time power is present and internal power supply is functioning. The following additional internal LEDs are present only on the standard MS/TP version of the BBPRO-V230, and are not present on BBPRO-V230-SP: A green LED (internal) near the blue LED will be blinking at a slow rate indicating the communications coprocessor is running. It should be blinking any time power is applied. A red LED (internal) is also located next to the green LED. The red should never light (except for a couple seconds at power-up), but if it does, it indicates a serious internal problem. It may be flashing a code consisting of several flashes followed by a pause. If so, count the flash code and report it to tech support. When the BBPRO-V230 is powered up, there is a pause of a couple of seconds while the red LED is on. Red at power up means the window is open for coprocessor firmware update. If it remains solid red at power up, it means the application image has become corrupt (contact tech support).

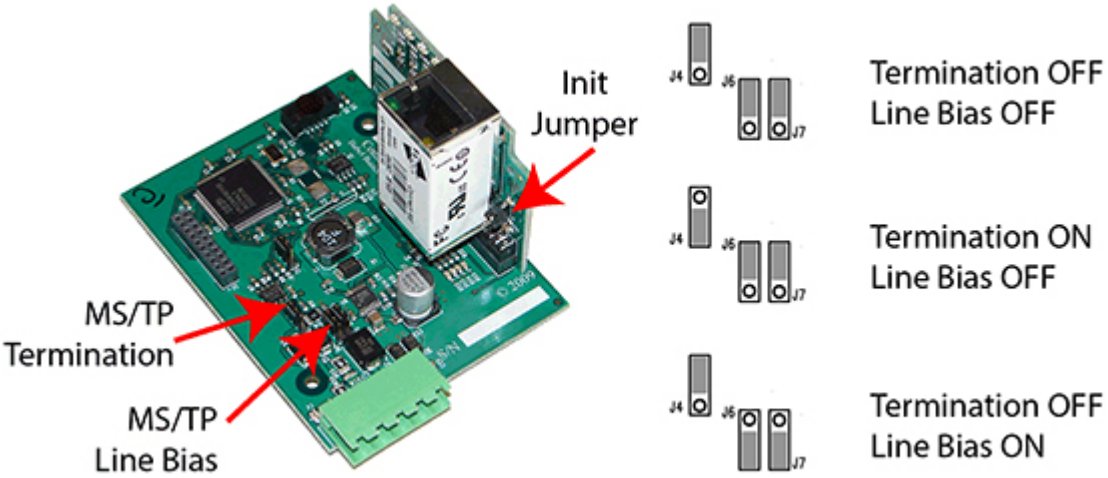
A.3 RS-485 Line Termination & Bias

Enable line termination only when this device is placed at the end of the network. Termination should only be enabled at two points on the network, and these two points must be specifically the end points.

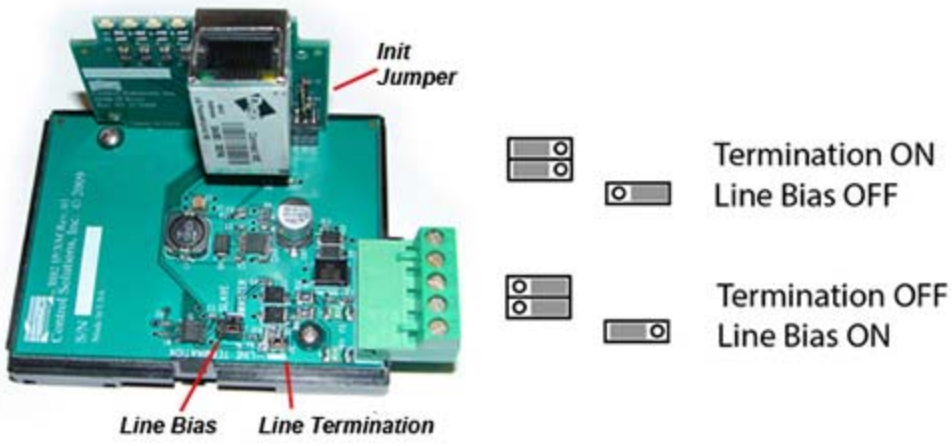
Enable line bias when needed. Line bias should only be enabled at one point on the network, and does not have to be the end point. Line bias holds the line in a known neutral state when no devices are transmitting. Without bias, the transition from offline to online by a transmitter can look like a false start bit and cause loss of communication.

The line conditioning options are enabled when the respective shunt is moved to the position indicated by the diagrams below.

Jumper locations for Babel Buster BBPRO-V230:



Jumper locations for Babel Buster BBPRO-V230-SP:



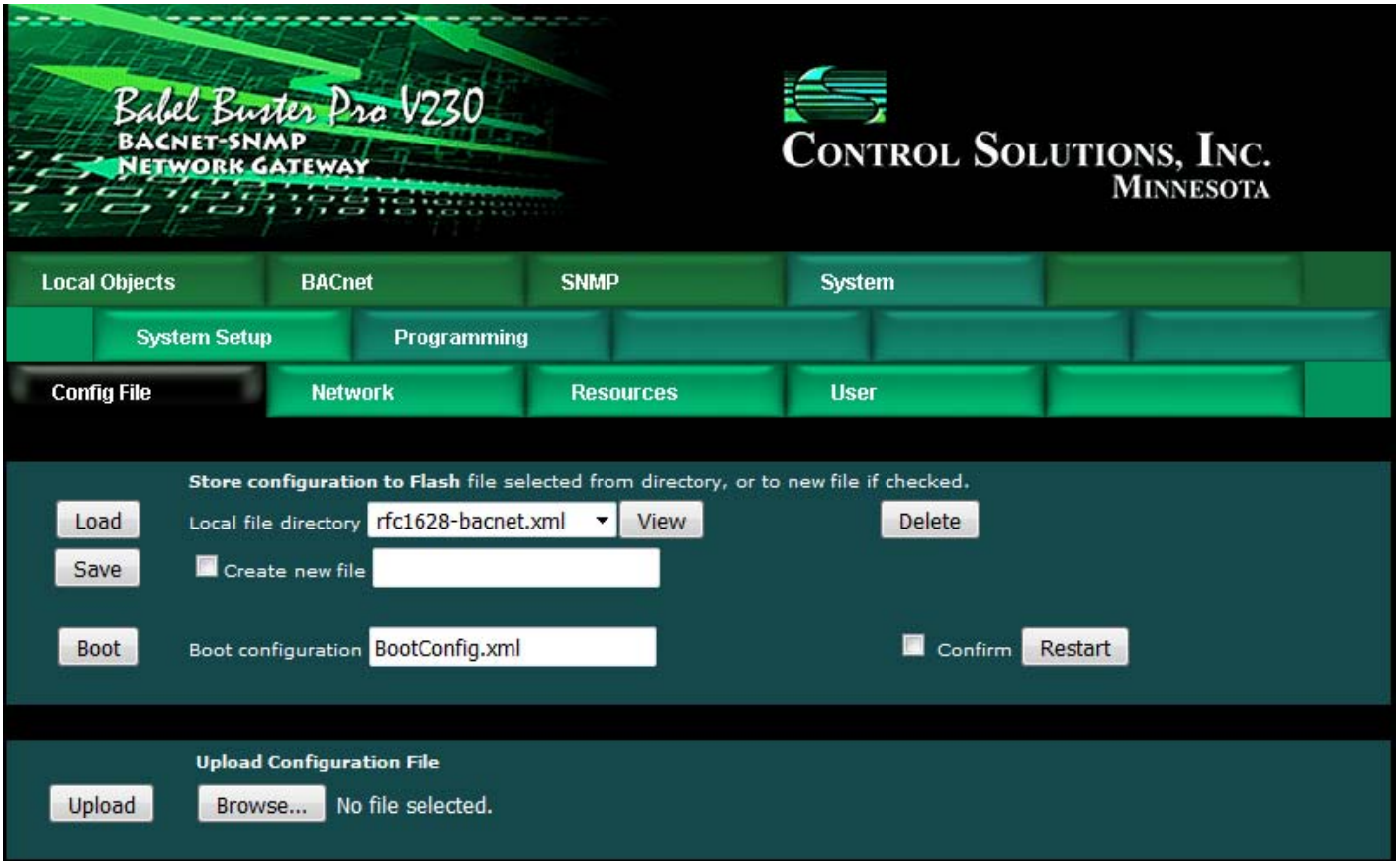
The "Init" jumper on the server module should only be used when advised by tech support. Installing this jumper prior to power-up causes the server to go into firmware update mode.

Appendix B Example Application: RFC 1628 UPS

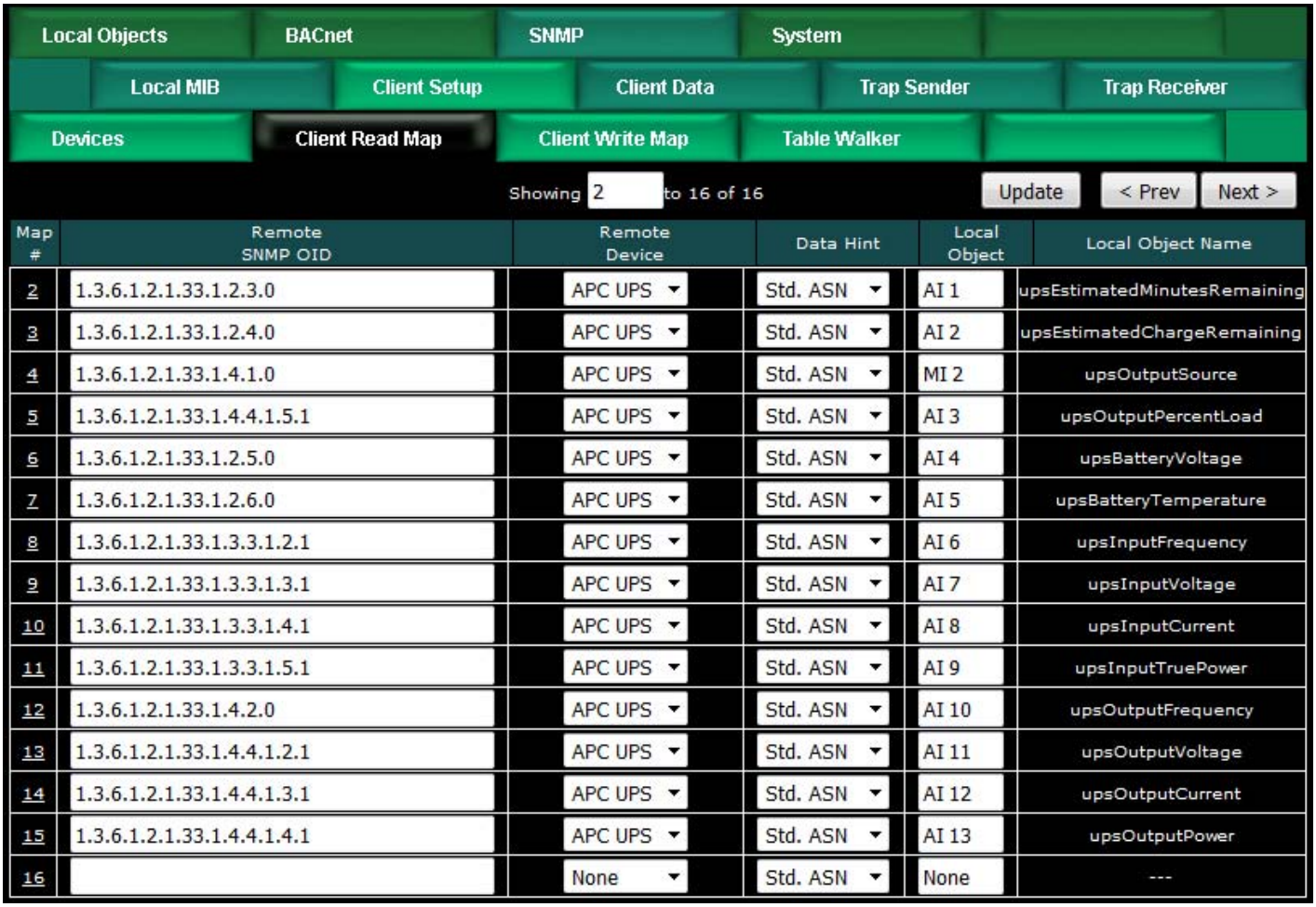
B.1 MIB Query

One of the most useful features of the Babel Buster Pro is its ability to interface to a UPS that implements RFC 1628. The MIB in this type of UPS can be queried to get data like voltage and current, but interpreting its alarms is a bit more complex. There is no way to query some particular OID and see that alarm condition exists. RFC 1628 implements an alarm table that does not lend itself well to simple polling like a gateway would normally like to do.

A configuration file for the Babel Buster Pro named "rfc1628-bacnet.xml" is available for download at csimn.com. This provides at least a good starting point for getting your UPS connected to BACnet, and may even be all you need. If your UPS is a more complex multi-line UPS, then you will need to use this configuration as an example to build out the full configuration, but if the UPS is a more common single line (single output) UPS, then this configuration will work as is.



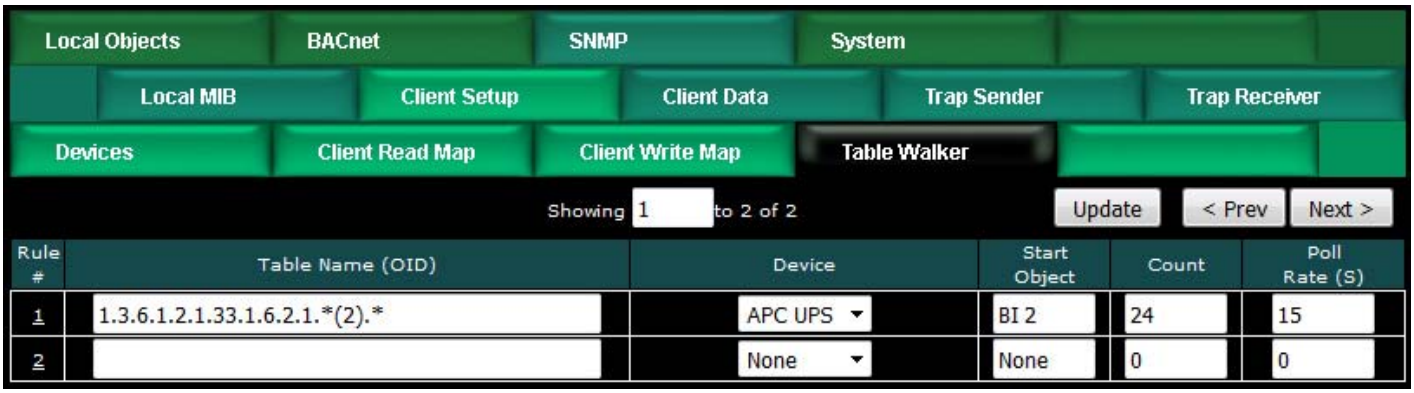
The values available in the MIB that can simply be polled are found in the SNMP Client Read Map list illustrated here.



Map #	Remote SNMP OID	Remote Device	Data Hint	Local Object	Local Object Name
2	1.3.6.1.2.1.33.1.2.3.0	APC UPS	Std. ASN	AI 1	upsEstimatedMinutesRemaining
3	1.3.6.1.2.1.33.1.2.4.0	APC UPS	Std. ASN	AI 2	upsEstimatedChargeRemaining
4	1.3.6.1.2.1.33.1.4.1.0	APC UPS	Std. ASN	MI 2	upsOutputSource
5	1.3.6.1.2.1.33.1.4.4.1.5.1	APC UPS	Std. ASN	AI 3	upsOutputPercentLoad
6	1.3.6.1.2.1.33.1.2.5.0	APC UPS	Std. ASN	AI 4	upsBatteryVoltage
7	1.3.6.1.2.1.33.1.2.6.0	APC UPS	Std. ASN	AI 5	upsBatteryTemperature
8	1.3.6.1.2.1.33.1.3.3.1.2.1	APC UPS	Std. ASN	AI 6	upsInputFrequency
9	1.3.6.1.2.1.33.1.3.3.1.3.1	APC UPS	Std. ASN	AI 7	upsInputVoltage
10	1.3.6.1.2.1.33.1.3.3.1.4.1	APC UPS	Std. ASN	AI 8	upsInputCurrent
11	1.3.6.1.2.1.33.1.3.3.1.5.1	APC UPS	Std. ASN	AI 9	upsInputTruePower
12	1.3.6.1.2.1.33.1.4.2.0	APC UPS	Std. ASN	AI 10	upsOutputFrequency
13	1.3.6.1.2.1.33.1.4.4.1.2.1	APC UPS	Std. ASN	AI 11	upsOutputVoltage
14	1.3.6.1.2.1.33.1.4.4.1.3.1	APC UPS	Std. ASN	AI 12	upsOutputCurrent
15	1.3.6.1.2.1.33.1.4.4.1.4.1	APC UPS	Std. ASN	AI 13	upsOutputPower
16		None	Std. ASN	None	---

B.2 Alarm Table Walker

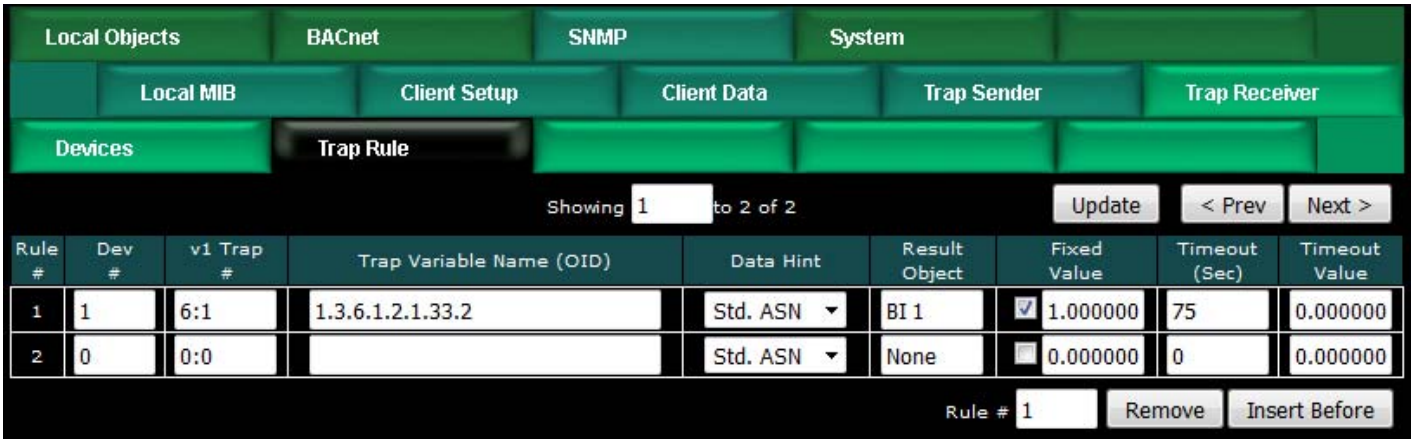
Finding active alarms in an RFC 1628 UPS is a bit trickier than just polling a given MIB. You need to walk the alarm table. This is discussed in much greater detail earlier in this user guide. The particular table walk needed to find alarms in the UPS is illustrated here and included in the rfc1628-bacnet.xml configuration.



Rule #	Table Name (OID)	Device	Start Object	Count	Poll Rate (S)
1	1.3.6.1.2.1.33.1.6.2.1.*(2).*	APC UPS	BI 2	24	15
2		None	None	0	0

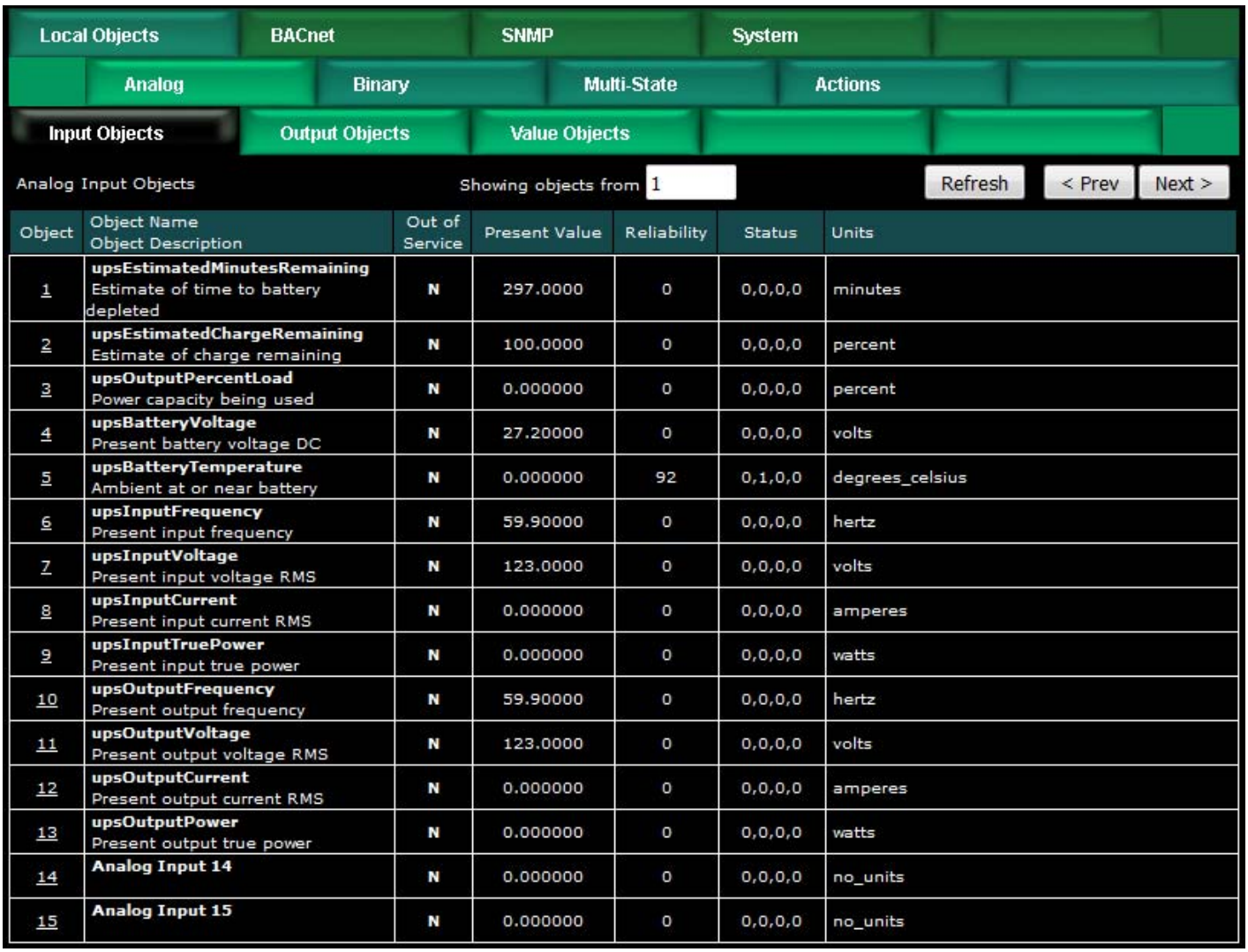
B.3 Trap Receiver

The rfc1628-bacnet.xml configuration also includes a trap receiver. This trap receiver may be considered redundant with the alarm table walker, but is included as an alternate alarm indication method that may be used if desired.



B.4 Example Data

The polled data when the UPS has normal utility power available is illustrated here.



The table walker in this case is setting a series of 24 Binary Inputs starting at BI 2 to the Active state if the corresponding alarm is active. These objects will return to Inactive when the alarm condition clears because the rfc1628-bacnet.xml table walk configuration includes resetting the objects to a default value of Inactive after a timeout. If you change the table walk poll rate, be sure to change the timeout to some value greater than the poll rate.

Local Objects | BACnet | SNMP | System

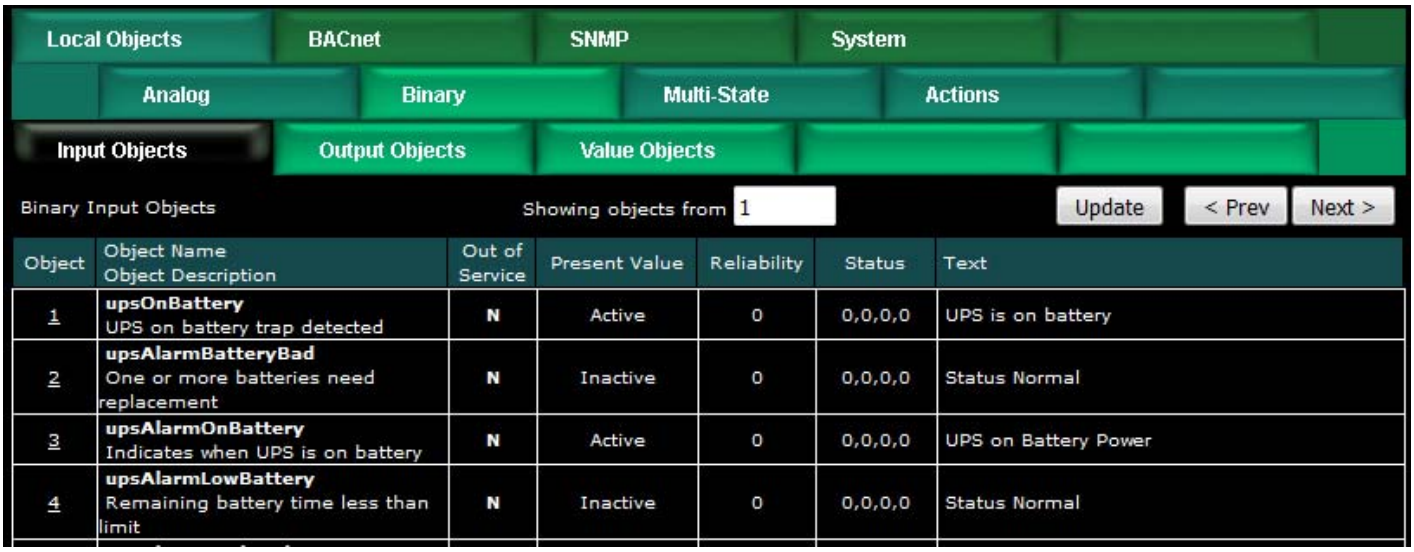
Analog | Binary | Multi-State | Actions

Input Objects | Output Objects | Value Objects

Binary Input Objects Showing objects from 1 Update < Prev Next >

Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Text
1	**upsOnBattery** UPS on battery trap detected	N	Inactive	0	0,0,0,0	UPS no longer on battery
2	**upsAlarmBatteryBad** One or more batteries need replacement	N	Inactive	0	0,0,0,0	Status Normal
3	**upsAlarmOnBattery** Indicates when UPS is on battery	N	Inactive	0	0,0,0,0	UPS on Utility Power
4	**upsAlarmLowBattery** Remaining battery time less than limit	N	Inactive	0	0,0,0,0	Status Normal
5	**upsAlarmDepletedBattery** UPS will be unable to sustain load	N	Inactive	0	0,0,0,0	Status Normal
6	**upsAlarmTempBad** Temperature out of tolerance	N	Inactive	0	0,0,0,0	Status Normal
7	**upsAlarmInputBad** An input condition is out of tolerance	N	Inactive	0	0,0,0,0	Status Normal
8	**upsAlarmOutputBad** Output condition out of tolerance	N	Inactive	0	0,0,0,0	Status Normal
9	**upsAlarmOutputOverLoad** Output exceeds UPS capacity	N	Inactive	0	0,0,0,0	Status Normal
10	**upsAlarmOnBypass** Bypass is engaged on UPS	N	Inactive	0	0,0,0,0	Status Normal
11	**upsAlarmBypassBad** Bypass is out of tolerance	N	Inactive	0	0,0,0,0	Status Normal
12	**upsAlarmOutputOffAsRequested** UPS has shut down as requested	N	Inactive	0	0,0,0,0	Status Normal
13	**upsAlarmUpsOffAsRequested** Entire UPS has shut down as requested	N	Inactive	0	0,0,0,0	Status Normal
14	**upsAlarmChargerFailed** Problem detected with charger system	N	Inactive	0	0,0,0,0	Status Normal
15	**upsAlarmUpsOutputOff** UPS output is in the Off state	N	Inactive	0	0,0,0,0	Status Normal

The screen shot above shows the Binary Inputs in their non-alarming state. The screen shot below shows the some of the Binary Input states when the UPS is running on battery.



Object	Object Name Object Description	Out of Service	Present Value	Reliability	Status	Text
1	**upsOnBattery** UPS on battery trap detected	**N**	Active	0	0,0,0,0	UPS is on battery
2	**upsAlarmBatteryBad** One or more batteries need replacement	**N**	Inactive	0	0,0,0,0	Status Normal
3	**upsAlarmOnBattery** Indicates when UPS is on battery	**N**	Active	0	0,0,0,0	UPS on Battery Power
4	**upsAlarmLowBattery** Remaining battery time less than limit	**N**	Inactive	0	0,0,0,0	Status Normal

Appendix C Script Basic Quick Reference

C.1 Accessing BACnet Objects

You may read any of the local objects using the getreg function, and write them using the setreg command. Note that getreg is a function while setreg is not; it is a command and therefore requires no parenthesis.

Usage is:

```
data = getreg (x)
setreg x, data
```

where 'x' is the BACnet object number encoded as indicated below, and 'data' is the Present Value of the BACnet object. Consider the following example:

```
MyData = getreg (22)
MyData = MyData * 2.5
setreg 2024, MyData
```

The above example will get the Present Value of AI 22 and place it in the variable MyData, then multiply it by 2.5, then place the resulting value in the Present Value of AV 24. Variables may be used in place of the constants used in the above example.

BACnet objects are accessed via the generic Basic "register" interface. Register numbers are BACnet object type multiplied times 1,000 plus object number starting at #1. Register numbers corresponding to BACnet objects are as follows:

Object Type	Starting Object Number	Starting Register Number
Analog Input	AI 1	1
Analog Output	AO 1	1001
Analog Value	AV 1	2001
Binary Input	BI 1	3001
Binary Output	BO 1	4001
Binary Value	BV 1	5001
Multi-State Input	MI 1	13001
Multi-State Output	MO 1	14001

Multi-State Value	MV 1	19001

C.2 Syntax Summary

The following summary provides an overview of most-used features. Some examples are shown in all upper case. Script BASIC is not case sensitive, therefore keywords may be upper or lower case. This also applies to variable names. Therefore, myvar and MYVAR are the same variable.

A complete reference for Script Basic is available online at www.csimn.com.

LET Statement

The "LET" keyword is archaic, and not used in Script Basic. Line numbers are also not used. Simply begin an assignment line with the variable you wish to assign. For example:

```
OurAverage = (YourVariable + MyVariable) / 2
A = B + C
phi = sin (theAngle)
```

Time Functions

The pre-defined variable NOW will provide the present real time in seconds since January 1, 1970 (Unix style) if real time clock support is present, or seconds since boot-up otherwise. Saving the value of NOW in a variable and calculating the difference some time later will allow measurement of real time. You use NOW in the same manner as any other variable except that you cannot assign a new value to NOW.

The function SLEEP will suspend program execution for some number of seconds, or generate a delay. Examples:

```
sleep (5)
sleep (0.25)
```

The first example will pause program execution for 5 seconds. The second example will pause for a quarter of a second.

Math Functions & Operators

The following is a summary of math operators and functions recognized in Script Basic:

+	Addition
-	Subtraction
*	Multiplication

/	Division
\	Integer Division
%	Modulus
^	a^b produces a raised to the power of b
ABS(n)	Returns absolute value of 'n'
ACOS(n)	Calculates arc cosine of 'n'
ASIN(n)	Calculates arc sine of 'n'
ATAN(n)	Calculates arc tangent of 'n'
COS(n)	Calculates cosine of 'n'
FIX(n)	Returns integral part of argument, rounding toward zero, i.e., int(-3.3) = -3
FRAC(n)	Returns the fractional part of the argument
INT(n)	Returns integral part of argument, rounding down, i.e., int(-3.3) = -4
LOG(n)	Produces the natural logarithm of n
LOG10(n)	Products the logarithm of n
PI	Produces the constant pi
POW(n)	Produces 10 raised to the power of n
RND	Provides a random number (use RANDOMIZE(n) to seed random number generator)
SIN(n)	Calculates sine of 'n'
SQR(n)	Calculates square root of 'n'
TAN(n)	Calculates tangent of 'n'

Logic Operators

The following is a summary of logic operators recognized in Script Basic:

AND	Logical AND - bitwise for values, or logical in "if" statements
NOT	Logical NOT - bitwise for values, or logical in "if" statements
OR	Logical OR - bitwise for values, or logical in "if" statements
XOR	Logical Exclusive OR - bitwise for values

Be sure to use parenthesis to establish precedence as necessary.

IF THEN ELSE Statement

There are two different ways to use this command: single line IF and multi line IF. (IF/THEN and other keywords are shown in upper case in these examples for clairty; however, they may be either upper or lower case.)

A single line IF has the form

```
IF condition THEN command
```

There is no way to specify any ELSE part for the command in the single line version. If you need the ELSE command you have to use the multi line IF.

The multi line IF should not contain any command directly after the keyword THEN. It should have the format:

```
IF condition THEN
 commands
ELSE
 commands
END IF
```

The ELSE part of the command is optional, thus the command can have the format

```
IF condition THEN
 commands
END IF
```

The condition is any valid comparison or expression. Examples include:

```
if a > b then print "greater"
if a <> b then print "not equal"
if getreg(3) then print "is enabled"
```

Contitional operators are:

=	Equal
<>	Not equal
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal

FOR NEXT Statement

Implements a FOR loop. The variable var gets the value of the start expression exp_start, and after each execution of the loop body it is incremented or decrement by the value exp_step until it reaches the stop value exp_stop.

```
FOR var= exp_start TO exp_stop [ STEP exp_step]
 ...
 commands to repeat
 ...
NEXT var
```

The STEP part of the command is optional. If this part is missing then the default value to increment the variable is 1.

If

- the expression exp_start is larger than the expression exp_stop and exp_step is positive or if
- the expression exp_start is smaller than the expression exp_stop and exp_step is negative

then the loop body is not executed even once and the variable retains its old value.

When the loop is executed at least once the variable gets the values one after the other and after the loop exists the loop variable holds the last value for which the loop already did not execute.

DO WHILE Statement

This command implements a looping construct that loops the code between the line DO WHILE and LOOP util the expression following the keywords on the loop starting line becomes false.

Practically this command is same as the command WHILE with a different syntax to be compatible with different BASIC implementations.

```
do while
 ...
loop
```

You can use the construct that creates more readable code.

WHILE Statement

Implements the 'while' loop as it is usually done in most basic implementations. The loop starts with the command while and finished with the line containing the keyword wend. The keyword while is followed by an expression and the loop is executes so long as long the expression is evaluated true.

```
while expression
 ...
 commands to repeat
 ...
wend
```

The expression is evaluated when the looping starts and each time the loop is restarted. It means that the code between the while and wend lines may be skipped totally if the expression evaluates to some false value during the first evaluation before the execution starts the loop.

In case some condition makes it necessary to exit the loop from its middle then the command GOTO can be used.

DO UNTIL Statement

This command implements a looping construct that loops the code between the line DO UNTIL and LOOP util the expression following the keywords on the loop starting line becomes true.

```
DO UNTIL expression
 ...
 commands to repeat
 ...
LOOP
```

The expression is evaluated when the looping starts and each time the loop is restarted. It means that the code between the DO UNTIL and LOOP lines may be skipped totally if the expression evaluates to some TRUE value during the first evaluation before the execution starts the loop.

This command is practically equivalent to the construct

```
WHILE NOT expression
 ...
 commands to repeat
 ...
WEND
```

You can and you also should use the construct that creates more readable code.

User Functions

The FUNCTION command is used to define a function. A function is a piece of code that can be called by the BASIC program from the main part or from a function or subroutine.

```
FUNCTION fun(a,b,c)
 ...
 fun = returnvalue
 ...
END FUNCTION
```

The end of the function is defined by the line containing the keywords END FUNCTION. This function would be called by the line x=fun(a,b,c) and the result would be placed in 'x'.

The SUB command should be used to define a subroutine. A subroutine is a piece of code that can be called by the BASIC program from the main part or from a function or subroutine.

```
SUB sub(a,b,c)
 ...
END SUB
```

The end of the subroutine is defined by the line containing the keywords END SUB.

Note that functions and subroutines are not really different in ScriptBasic. ScriptBasic allows you to return a value from a subroutine and to call a function using the command CALL. It is just a convention to have separately SUB and FUNCTION declarations. You would call this subroutine with the line CALL sub.

Labels and GOTO Statement

The statement GOTO is the most famous statement of all BASIC languages. Many program theorists say that you should not ever use GOTO and they may be right. Even so, the statement GOTO is part of most programming languages and ScriptBasic is no exception.

Using the statement GOTO you can alter the execution order of the statements. GOTO statements use labels and the labels can identify program lines. The form of a GOTO statement is

```
GOTO label
```

where the label is a label, which is defined somewhere in a program.

Labels are local within functions and subroutines. You can not jump into a subroutine or jump out of it. Labels begin at the start of a line, are the only thing on the line, and end with a colon.

```
GOTO mylabel
...
commands
...
mylabel:
...
```

Use of GOTO is usually discouraged and is against structural programming. Whenever you feel the need to use the GOTO statement (except ON ERROR GOTO) think twice whether there is a better solution without utilizing the statement GOTO.

Typical use of the GOTO statement to jump out of some error condition to the error handling code or jumping out of some loop on some condition.

END Statement

End of the program. Stops program execution.

You should usually use this command to signal the end of a program. Although you can use STOP and END interchangeably this is the convention in BASIC programs to use the command END to note the end of the program and STOP to stop the program execution based on some extra condition inside the code.

Variables and Arrays

Variables may be any name that starts with a letter, and after the first letter, may also contain digits, underscore, and dollar sign (for old style denotation of string variables). Variables are automatically typed as integer, double (floating point), or string according to context in which they are used, and may dynamically change type in the course of program execution.

Arrays are created automatically as soon as you use subscripts. Uninitialized elements of the arrays return "undef". Array subscripts use square brackets (not parenthesis which are easily confused with function calls). For example:

```
a[1] = 45.9
a[2] = 99.8
a[3] = "something else"
a[4,12] = 10
```

These are all valid elements of the same array. Normally the index values must be integer. However, Script Basic also supports associative arrays. Associative arrays use "curly" brackets. For example:

```
animal{"cat"} = "Garfield"
animal{"dog"} = "Snoopy"
animal{"tiger"} = "Tony"
```

Print Statement

You may use a Print statement to show output in the virtual terminal window. Although difficult to work with via HTML, you can also do a LINE INPUT statement in the input window of the virtual terminal when the program is run in the foreground (Start button). No part of the virtual terminal is available to the auto run program. An example of a print statement might be:

```
print "Register #5 is ", getreg(5), "\n"
```

An example of the line input statement is:

```
line input myline$
```

The use of "$" as part of a string variable name is optional, and would be included only for readability. Any valid variable name may contain an integer, a floating point value, or a string, and will be dynamically typed according to the context.

Rem Statement

Remarks or comments are any line starting with REM or a single quote. Everything to the end of the line will be ignored. Remarks may not be added to the end of a valid line of code.

Appendix D BACnet, SNMP Extensions for Script Basic

D.1 BACnet Funtions

The Babel Buster Pro includes several functions that are unique to its implementation of Script Basic. The language in general is summarized in Appendix C. A much more detailed collection of help may be found in the external compiled help file that may be downloaded at www.csimn.com. The compiled help includes reference guides for all of the string manipulation functions. The compiled help does not include the special functions documented here.

BACnet Object Status and Reliability Codes

```
n = getobjstatus (x)
```

Returns 4-bit value representing the 4 bits indicated for status for BACnet object 'x' as displayed on the object data web pages. A value of 0 means there are no faults and the object is not out of service. The object number 'x' is encoded as for the 'setreg' command.

```
n = getobjrel (x)
```

Returns reliability code for BACnet object 'x'. A value of 0 means there are no faults. The object number 'x' is encoded as for the 'setreg' command. The meaning of the reliability codes in the range of 1-63 is defined by the BACnet protocol standard. The meaning of codes 64-100 are displayed as applicable on the various rule pages in the web UI for this device. The meaning of reliability codes 101-199 are defined by the user's Script Basic program if applicable.

```
setobjrel x, y
```

Sets the reliability code for BACnet object 'x'. A value of 0 means there are no faults. The object number 'x' is encoded as for the 'setreg' command. The meaning of the reliability codes in the range of 1-63 is defined by the BACnet protocol standard. The meaning of codes 64-100 are displayed as applicable on the various rule pages in the web UI for this device. The meaning of reliability codes 101-199 are defined by the user's Script Basic program if applicable.

The value 'y' must be in the range of 101-199 when your program uses this command,

or it will be ignored. If you set the reliability code to any non-zero value, the object will be considered to be in fault by any BMS system. If you use this feature to indicate the fact that your program has detected a problem, you must also be sure to set the value back to zero when your program detects that the problem no longer exists.

Note that 'segobjrel' is a command, not a function, and therefore you do not enclose the parameters in parenthesis.

You will need to make two consecutive calls to 'setobjrel' before the value will appear in the objects reliability code. The BACnet objects in the BBPRO-V230 need to see the same reliability code twice before it will be registered as the offical code. This is to avoid triggering action on spurious events like a one-time communication hiccup. If you are setting the reliability code each time through an indefinite loop, only one call will suffice since the same persistent problem will result in repeated calls with the same reliability code.

You may find that the reliability code does not clear to zero simply by calling 'setobjrel' with a code of zero. Reliability codes are latching in nature to better allow diagnosing problems. The code will not be allowed to "unlatch" until it has been read by a BACnet client. To override this latching, go to the BACnet page and check the box marked "Allow fault self-reset without Ack" and click Save.

D.2 SNMP Functions

Trap Information Retrieval

```
n = gettrapinfo (x)
```

This function is used first of all to see if any trap has been received. Then, if a trap has been received, it is used to retrieve several pieces of information about that trap. Each time gettrapinfo(0) or gettrapinfo(-1) returns a non-zero value indicating a trap was received, the previously retained trap is discarded. Until gettrapinfo(0) or gettrapinfo(-1) is called again, the same received trap will be retained for all subsequent trap related function calls.

-1 = same as 0, but suspend until there is a trap to return
0 = return trap version if new trap (0=no new trap, 1=v1, 2=v2). Each gettrapinfo(0) releases the previous trap, and queues up a new trap, if any, for subsequent calls using x=1..6
1 = return trap generic number if v1, or 0
2 = return trap specific number if v1, or 0
3 = return number of varbinds in trap
4 = return IP address of sender as char string
5 = return uptime from trap as integer
6 = return enterprise OID from trap as string
For gettrapinfo(x), the various pieces of information are requested using selector 'x'

using one of:

```
n = gettrapname (x)
```

Returns var name as string for varbind 'x', where x is 1..max as returned by gettrapinfo(3).

```
n = gettrapdata (x,y)
```

Returns var value for varbind 'x', where x is 1..max as returned by gettrapinfo(3). Format of data is specified by format 'y' where format is:
0 = want character string
1 = want integer value (convert as necessary)
2 = want floating point (convert as necessary) (also indicates RFC 6340 expected if octet string)

Virtual Object Access

The BBPRO-V230 contains a set of "virtual objects" that provide a data connection between SNMP and Basic without going through BACnet. This is sometimes needed because the SNMP data type known as "Octet String" is defined only as a string of bytes and those bytes can contain anything, including raw binary data. The role of the Virtual Object is to allow transferring of data directly between SNMP and Basic so that Basic can then make customized translations as necessary to derive meaningful BACnet data.

Virtual Objects are referenced as "XV n" in the device's web pages, where 'n' counts from 1 to the maximum allocated on the Resources page. In Script Basic, they are accessed using the 'getvar' and 'setvar' functions. Usage is as follows

```
MyString = getvar (x)
setvar x, MyString
```

where 'x' is the virtual object number (e.g. 1 if referring to XV 1), and MyString is a string variable. All of the Script Basic string functions may be used on the MyString variable (which may be any variable name).

The SNMP Client read and write maps are used to interface with SNMP devices to move the data between that device and the virtual object. Script Basic is used to translate between Octet Strings and BACnet object types. The 'getreg' and 'setreg' functions are used to access data in BACnet objects (see Appendix C).

D.3 Other Functions

Error Information Retrieval

```
n = geterr (x,y)
```

Returns n=0 for no error, otherwise returns error code or status for item specified by

'x', item number 'y' (1..N). Error codes returned are those displayed on the respective web pages as error codes or device status.

Item 'x' may be:
1 = SNMP client device
2 = SNMP trap sender device
3 = (not used)
4 = BACnet IP client device
5 = (not used)
6 = SNMP client read map
7 = SNMP client write map
8 = SNMP table walk rule
9 = (not used)
10 = BACnet client read rule
11 = BACnet client write rule

IP Address Retrieval

```
ip$ = getipaddr (x,y)
```

Returns IP address for item type 'x', item number 'y' (1..N), formatted as character string.

Item 'x' may be:
0 = Our own IP address
1 = SNMP client device
2 = SNMP trap sender device
3 = SNMP trap receiver device
4 = BACnet IP device

Comm Port Timeout:

This discussion of "comm port" only applies to the -SP version of BBPRO-V230.

```
timeout (n)
```

Sets timeout in seconds that the "line input #n" request for input from the communiation port will wait before returning an empty string if nothing was received on the communication port. Otherwise, the line input will wait indefinitely for a line that ends in a line end character.

The line end character is otherwise known as line feed or "\n". The carriage return will be ignored, unless it is specified to be the line end instead.

To open the communication port (-SP version only) as file #2 for example, you would use:

```
open "COM:9600" for comm as 2
```

To open the comm port using the carriage return as line end instead of the (Linux) line end, you would use:

```
open "COM:9600,CR" for comm as 2
```

Many devices return both carriage return and line feed at the ends of lines. The sole line end character recognized as the end of line for the comm port will end the line while the other will be discarded and not returned in the string that gets placed in a variable for Basic.

Appendix E BACnet Object Properties

E.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_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 IP client, device timeout (82) BACnet IP client, error returned by server (83) SNMP client, no response from agent (91) SNMP client, agent returned 'no such name' error (92) SNMP client, unable to parse data returned (93) SNMP client, reply did not match request (94) SNMP client, invalid table walk configuration (95) SNMP client, unable to parse PDU returned (96) SNMP client, table walk came up short on data (97) SNMP client, agent returned error other than 'no such name' (98)
Out_Of_Service (81) (W)	BOOLEAN
COV_Increment (22) (W)	REAL (analog objects only)
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)

E.2 Device Object Properties

The following properties are found in the Device object of the BBPRO-V230. In addition to standard Device properties.

Property	Encoding
Object_Identifier (75)	BACnetObjectIdentifier
Object_Name (77)	CharacterString

Object_Type (79)	BACnetObjectType ENUMERATED: device (8)
System_Status (112)	BACnetDeviceStatus
Vendor_Name (121)	CharacterString
Vendor_Identifier (120)	Unsigned16 (should always return 208)
Model_Name (70)	CharacterString
Fimrware_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

Appendix F BACnet Codes

F.1 BACnet Object Property Codes

BACnet property type codes may be found in your copy of the BACnet protocol specification, ANSI/ASHRAE Standard 135. That document is copyrighted, but the C enumeration shown below for reference is taken from open source code available under GPL at http://sourceforge.net, and provides essentially the same information (copyrighted by Steve Karg, licensed under GPL as noted at http://sourceforge.net).

```
typedef enum {
    PROP_ACKED_TRANSITIONS = 0,
    PROP_ACK_REQUIRED = 1,
    PROP_ACTION = 2,
    PROP_ACTION_TEXT = 3,
    PROP_ACTIVE_TEXT = 4,
    PROP_ACTIVE_VT_SESSIONS = 5,
    PROP_ALARM_VALUE = 6,
    PROP_ALARM_VALUES = 7,
    PROP_ALL = 8,
    PROP_ALL_WRITES_SUCCESSFUL = 9,
    PROP_APDU_SEGMENT_TIMEOUT = 10,
    PROP_APDU_TIMEOUT = 11,
    PROP_APPLICATION_SOFTWARE_VERSION = 12,
    PROP_ARCHIVE = 13,
    PROP_BIAS = 14,
    PROP_CHANGE_OF_STATE_COUNT = 15,
    PROP_CHANGE_OF_STATE_TIME = 16,
    PROP_NOTIFICATION_CLASS = 17,
    PROP_BLANK_1 = 18,
    PROP_CONTROLLED_VARIABLE_REFERENCE = 19,
    PROP_CONTROLLED_VARIABLE_UNITS = 20,
    PROP_CONTROLLED_VARIABLE_VALUE = 21,
    PROP_COV_INCREMENT = 22,
    PROP_DATE_LIST = 23,
    PROP_DAYLIGHT_SAVINGS_STATUS = 24,
    PROP_DEADBAND = 25,
    PROP_DERIVATIVE_CONSTANT = 26,
    PROP_DERIVATIVE_CONSTANT_UNITS = 27,
    PROP_DESCRIPTION = 28,
    PROP_DESCRIPTION_OF_HALT = 29,
```

```
PROP_DEVICE_ADDRESS_BINDING = 30,
PROP_DEVICE_TYPE = 31,
PROP_EFFECTIVE_PERIOD = 32,
PROP_ELAPSED_ACTIVE_TIME = 33,
PROP_ERROR_LIMIT = 34,
PROP_EVENT_ENABLE = 35,
PROP_EVENT_STATE = 36,
PROP_EVENT_TYPE = 37,
PROP_EXCEPTION_SCHEDULE = 38,
PROP_FAULT_VALUES = 39,
PROP_FEEDBACK_VALUE = 40,
PROP_FILE_ACCESS_METHOD = 41,
PROP_FILE_SIZE = 42,
PROP_FILE_TYPE = 43,
PROP_FIRMWARE_REVISION = 44,
PROP_HIGH_LIMIT = 45,
PROP_INACTIVE_TEXT = 46,
PROP_IN_PROCESS = 47,
PROP_INSTANCE_OF = 48,
PROP_INTEGRAL_CONSTANT = 49,
PROP_INTEGRAL_CONSTANT_UNITS = 50,
PROP_ISSUE_CONFIRMED_NOTIFICATIONS = 51,
PROP_LIMIT_ENABLE = 52,
PROP_LIST_OF_GROUP_MEMBERS = 53,
PROP_LIST_OF_OBJECT_PROPERTY_REFERENCES = 54,
PROP_LIST_OF_SESSION_KEYS = 55,
PROP_LOCAL_DATE = 56,
PROP_LOCAL_TIME = 57,
PROP_LOCATION = 58,
PROP_LOW_LIMIT = 59,
PROP_MANIPULATED_VARIABLE_REFERENCE = 60,
PROP_MAXIMUM_OUTPUT = 61,
PROP_MAX_APDU_LENGTH_ACCEPTED = 62,
PROP_MAX_INFO_FRAMES = 63,
PROP_MAX_MASTER = 64,
PROP_MAX_PRES_VALUE = 65,
PROP_MINIMUM_OFF_TIME = 66,
PROP_MINIMUM_ON_TIME = 67,
PROP_MINIMUM_OUTPUT = 68,
PROP_MIN_PRES_VALUE = 69,
PROP_MODEL_NAME = 70,
PROP_MODIFICATION_DATE = 71,
PROP_NOTIFY_TYPE = 72,
PROP_NUMBER_OF_APDU_RETRIES = 73,
PROP_NUMBER_OF_STATES = 74,
PROP_OBJECT_IDENTIFIER = 75,
PROP_OBJECT_LIST = 76,
PROP_OBJECT_NAME = 77,
PROP_OBJECT_PROPERTY_REFERENCE = 78,
PROP_OBJECT_TYPE = 79,
PROP_OPTIONAL = 80,
```

```
    PROP_OUT_OF_SERVICE = 81,
    PROP_OUTPUT_UNITS = 82,
    PROP_EVENT_PARAMETERS = 83,
    PROP_POLARITY = 84,
    PROP_PRESENT_VALUE = 85,
    PROP_PRIORITY = 86,
    PROP_PRIORITY_ARRAY = 87,
    PROP_PRIORITY_FOR_WRITING = 88,
    PROP_PROCESS_IDENTIFIER = 89,
    PROP_PROGRAM_CHANGE = 90,
    PROP_PROGRAM_LOCATION = 91,
    PROP_PROGRAM_STATE = 92,
    PROP_PROPORTIONAL_CONSTANT = 93,
    PROP_PROPORTIONAL_CONSTANT_UNITS = 94,
    PROP_PROTOCOL_CONFORMANCE_CLASS = 95,       /* deleted in version 1
revision 2 */
    PROP_PROTOCOL_OBJECT_TYPES_SUPPORTED = 96,
    PROP_PROTOCOL_SERVICES_SUPPORTED = 97,
    PROP_PROTOCOL_VERSION = 98,
    PROP_READ_ONLY = 99,
    PROP_REASON_FOR_HALT = 100,
    PROP_RECIPIENT = 101,
    PROP_RECIPIENT_LIST = 102,
    PROP_RELIABILITY = 103,
    PROP_RELINQUISH_DEFAULT = 104,
    PROP_REQUIRED = 105,
    PROP_RESOLUTION = 106,
    PROP_SEGMENTATION_SUPPORTED = 107,
    PROP_SETPOINT = 108,
    PROP_SETPOINT_REFERENCE = 109,
    PROP_STATE_TEXT = 110,
    PROP_STATUS_FLAGS = 111,
    PROP_SYSTEM_STATUS = 112,
    PROP_TIME_DELAY = 113,
    PROP_TIME_OF_ACTIVE_TIME_RESET = 114,
    PROP_TIME_OF_STATE_COUNT_RESET = 115,
    PROP_TIME_SYNCHRONIZATION_RECIPIENTS = 116,
    PROP_UNITS = 117,
    PROP_UPDATE_INTERVAL = 118,
    PROP_UTC_OFFSET = 119,
    PROP_VENDOR_IDENTIFIER = 120,
    PROP_VENDOR_NAME = 121,
    PROP_VT_CLASSES_SUPPORTED = 122,
    PROP_WEEKLY_SCHEDULE = 123,
    PROP_ATTEMPTED_SAMPLES = 124,
    PROP_AVERAGE_VALUE = 125,
    PROP_BUFFER_SIZE = 126,
    PROP_CLIENT_COV_INCREMENT = 127,
    PROP_COV_RESUBSCRIPTION_INTERVAL = 128,
    PROP_CURRENT_NOTIFY_TIME = 129,
    PROP_EVENT_TIME_STAMPS = 130,
```

```
PROP_LOG_BUFFER = 131,
PROP_LOG_DEVICE_OBJECT = 132,
/* The enable property is renamed from log-enable in
   Addendum b to ANSI/ASHRAE 135-2004(135b-2) */
PROP_ENABLE = 133,
PROP_LOG_INTERVAL = 134,
PROP_MAXIMUM_VALUE = 135,
PROP_MINIMUM_VALUE = 136,
PROP_NOTIFICATION_THRESHOLD = 137,
PROP_PREVIOUS_NOTIFY_TIME = 138,
PROP_PROTOCOL_REVISION = 139,
PROP_RECORDS_SINCE_NOTIFICATION = 140,
PROP_RECORD_COUNT = 141,
PROP_START_TIME = 142,
PROP_STOP_TIME = 143,
PROP_STOP_WHEN_FULL = 144,
PROP_TOTAL_RECORD_COUNT = 145,
PROP_VALID_SAMPLES = 146,
PROP_WINDOW_INTERVAL = 147,
PROP_WINDOW_SAMPLES = 148,
PROP_MAXIMUM_VALUE_TIMESTAMP = 149,
PROP_MINIMUM_VALUE_TIMESTAMP = 150,
PROP_VARIANCE_VALUE = 151,
PROP_ACTIVE_COV_SUBSCRIPTIONS = 152,
PROP_BACKUP_FAILURE_TIMEOUT = 153,
PROP_CONFIGURATION_FILES = 154,
PROP_DATABASE_REVISION = 155,
PROP_DIRECT_READING = 156,
PROP_LAST_RESTORE_TIME = 157,
PROP_MAINTENANCE_REQUIRED = 158,
PROP_MEMBER_OF = 159,
PROP_MODE = 160,
PROP_OPERATION_EXPECTED = 161,
PROP_SETTING = 162,
PROP_SILENCED = 163,
PROP_TRACKING_VALUE = 164,
PROP_ZONE_MEMBERS = 165,
PROP_LIFE_SAFETY_ALARM_VALUES = 166,
PROP_MAX_SEGMENTS_ACCEPTED = 167,
PROP_PROFILE_NAME = 168,
PROP_AUTO_SLAVE_DISCOVERY = 169,
PROP_MANUAL_SLAVE_ADDRESS_BINDING = 170,
PROP_SLAVE_ADDRESS_BINDING = 171,
PROP_SLAVE_PROXY_ENABLE = 172,
PROP_LAST_NOTIFY_TIME = 173,
PROP_SCHEDULE_DEFAULT = 174,
PROP_ACCEPTED_MODES = 175,
PROP_ADJUST_VALUE = 176,
PROP_COUNT = 177,
PROP_COUNT_BEFORE_CHANGE = 178,
PROP_COUNT_CHANGE_TIME = 179,
```

```
    PROP_COV_PERIOD = 180,
    PROP_INPUT_REFERENCE = 181,
    PROP_LIMIT_MONITORING_INTERVAL = 182,
    PROP_LOGGING_DEVICE = 183,
    PROP_LOGGING_RECORD = 184,
    PROP_PRESCALE = 185,
    PROP_PULSE_RATE = 186,
    PROP_SCALE = 187,
    PROP_SCALE_FACTOR = 188,
    PROP_UPDATE_TIME = 189,
    PROP_VALUE_BEFORE_CHANGE = 190,
    PROP_VALUE_SET = 191,
    PROP_VALUE_CHANGE_TIME = 192,
    /* enumerations 193-206 are new */
    PROP_ALIGN_INTERVALS = 193,
    PROP_GROUP_MEMBER_NAMES = 194,
    PROP_INTERVAL_OFFSET = 195,
    PROP_LAST_RESTART_REASON = 196,
    PROP_LOGGING_TYPE = 197,
    PROP_MEMBER_STATUS_FLAGS = 198,
    PROP_NOTIFICATION_PERIOD = 199,
    PROP_PREVIOUS_NOTIFY_RECORD = 200,
    PROP_REQUESTED_UPDATE_INTERVAL = 201,
    PROP_RESTART_NOTIFICATION_RECIPIENTS = 202,
    PROP_TIME_OF_DEVICE_RESTART = 203,
    PROP_TIME_SYNCHRONIZATION_INTERVAL = 204,
    PROP_TRIGGER = 205,
    PROP_UTC_TIME_SYNCHRONIZATION_RECIPIENTS = 206,
    /* enumerations 207-211 are used in Addendum d to ANSI/ASHRAE
135-2004 */
    PROP_NODE_SUBTYPE = 207,
    PROP_NODE_TYPE = 208,
    PROP_STRUCTURED_OBJECT_LIST = 209,
    PROP_SUBORDINATE_ANNOTATIONS = 210,
    PROP_SUBORDINATE_LIST = 211,
    /* enumerations 212-225 are used in Addendum e to ANSI/ASHRAE
135-2004 */
    PROP_ACTUAL_SHED_LEVEL = 212,
    PROP_DUTY_WINDOW = 213,
    PROP_EXPECTED_SHED_LEVEL = 214,
    PROP_FULL_DUTY_BASELINE = 215,
    /* enumerations 216-217 are used in Addendum i to ANSI/ASHRAE
135-2004 */
    PROP_BLINK_PRIORITY_THRESHOLD = 216,
    PROP_BLINK_TIME = 217,
    /* enumerations 212-225 are used in Addendum e to ANSI/ASHRAE
135-2004 */
    PROP_REQUESTED_SHED_LEVEL = 218,
    PROP_SHED_DURATION = 219,
    PROP_SHED_LEVEL_DESCRIPTIONS = 220,
    PROP_SHED_LEVELS = 221,
```

```
    PROP_STATE_DESCRIPTION = 222,
    /* enumerations 223-225 are used in Addendum i to ANSI/ASHRAE
135-2004 */
    PROP_FADE_TIME = 223,
    PROP_LIGHTING_COMMAND = 224,
    PROP_LIGHTING_COMMAND_PRIORITY = 225,
    /* enumerations 226-235 are used in Addendum f to ANSI/ASHRAE
135-2004 */
    PROP_DOOR_ALARM_STATE = 226,
    PROP_DOOR_EXTENDED_PULSE_TIME = 227,
    PROP_DOOR_MEMBERS = 228,
    PROP_DOOR_OPEN_TOO_LONG_TIME = 229,
    PROP_DOOR_PULSE_TIME = 230,
    PROP_DOOR_STATUS = 231,
    PROP_DOOR_UNLOCK_DELAY_TIME = 232,
    PROP_LOCK_STATUS = 233,
    PROP_MASKED_ALARM_VALUES = 234,
    PROP_SECURED_STATUS = 235,
    /* enumerations 236-243 are used in Addendum i to ANSI/ASHRAE
135-2004 */
    PROP_OFF_DELAY = 236,
    PROP_ON_DELAY = 237,
    PROP_POWER = 238,
    PROP_POWER_ON_VALUE = 239,
    PROP_PROGRESS_VALUE = 240,
    PROP_RAMP_RATE = 241,
    PROP_STEP_INCREMENT = 242,
    PROP_SYSTEM_FAILURE_VALUE = 243,
    /* enumerations 244-311 are used in Addendum j to ANSI/ASHRAE
135-2004 */
    PROP_ABSENTEE_LIMIT = 244,
    PROP_ACCESS_ALARM_EVENTS = 245,
    PROP_ACCESS_DOORS = 246,
    PROP_ACCESS_EVENT = 247,
    PROP_ACCESS_EVENT_AUTHENTICATION_FACTOR = 248,
    PROP_ACCESS_EVENT_CREDENTIAL = 249,
    PROP_ACCESS_EVENT_TIME = 250,
    PROP_ACCESS_RULES = 251,
    PROP_ACCESS_RULES_ENABLE = 252,
    PROP_ACCESS_TRANSACTION_EVENTS = 253,
    PROP_ACCOMPANIED = 254,
    PROP_ACTIVATION_TIME = 255,
    PROP_ACTIVE_AUTHENTICATION_POLICY = 256,
    PROP_ASSIGNED_ACCESS_RIGHTS = 257,
    PROP_AUTHENTICATION_FACTOR_INPUT_LIST = 258,
    PROP_AUTHENTICATION_FACTORS = 259,
    PROP_AUTHENTICATION_POLICY_LIST = 260,
    PROP_AUTHENTICATION_POLICY_NAMES = 261,
    PROP_AUTHORIZATION_MODE = 262,
    PROP_BELONGS_TO = 263,
    PROP_CREDENTIAL_DISABLE = 264,
```

```
    PROP_CREDENTIAL_STATUS = 265,
    PROP_CREDENTIALS = 266,
    PROP_CREDENTIALS_IN_ZONE = 267,
    PROP_DAYS_REMAINING = 268,
    PROP_ENTRY_POINTS = 269,
    PROP_EXIT_POINTS = 270,
    PROP_EXPIRY_TIME = 271,
    PROP_EXTENDED_TIME_ENABLE = 272,
    PROP_FAILED_ATTEMPT_EVENTS = 273,
    PROP_FAILED_ATTEMPTS = 274,
    PROP_FAILED_ATTEMPTS_TIME = 275,
    PROP_FORMAT_CLASS_SUPPORTED = 276,
    PROP_FORMAT_TYPE = 277,
    PROP_LAST_ACCESS_EVENT = 278,
    PROP_LAST_ACCESS_POINT = 279,
    PROP_LAST_CREDENTIAL_ADDED = 280,
    PROP_LAST_CREDENTIAL_ADDED_TIME = 281,
    PROP_LAST_CREDENTIAL_REMOVED = 282,
    PROP_LAST_CREDENTIAL_REMOVED_TIME = 283,
    PROP_LAST_USE_TIME = 284,
    PROP_LOCKDOWN = 285,
    PROP_LOCKDOWN_RELINQUISH_TIME = 286,
    PROP_MASTER_EXEMPTION = 287,
    PROP_MAX_FAILED_ATTEMPTS = 288,
    PROP_MEMBERS = 289,
    PROP_MASTER_POINT = 290,
    PROP_NUMBER_OF_AUTHENTICATION_POLICIES = 291,
    PROP_OCCUPANCY_COUNT = 293,
    PROP_OCCUPANCY_COUNT_ENABLE = 294,
    PROP_OCCUPANCY_COUNT_EXEMPTION = 295,
    PROP_OCCUPANCY_LOWER_THRESHOLD = 296,
    PROP_OCCUPANCY_LOWER_THRESHOLD_ENFORCED = 297,
    PROP_OCCUPANCY_STATE = 298,
    PROP_OCCUPANCY_UPPER_LIMIT = 299,
    PROP_OCCUPANCY_UPPER_LIMIT_ENFORCED = 300,
    PROP_PASSBACK_EXEMPTION = 301,
    PROP_PASSBACK_MODE = 302,
    PROP_PASSBACK_TIMEOUT = 303,
    PROP_POSITIVE_ACCESS_RULES = 304,
    PROP_READ_STATUS = 305,
    PROP_REASON_FOR_DISABLE = 306,
    PROP_THREAT_AUTHORITY = 307,
    PROP_THREAT_LEVEL = 308,
    PROP_TRACE_FLAG = 309,
    PROP_TRANSACTION_NOTIFICATION_CLASS = 310,
    PROP_USER_EXTERNAL_IDENTIFIER = 311,
    /* enumerations 312-313 are used in Addendum k to ANSI/ASHRAE
135-2004 */
    PROP_CHARACTER_SET = 312,
    PROP_STRICT_CHARACTER_MODE = 313,
    /* enumerations 312-313 are used in Addendum k to ANSI/ASHRAE
```

```
135-2004 */
    PROP_BACKUP_AND_RESTORE_STATE = 314,
    PROP_BACKUP_PREPARATION_TIME = 315,
    PROP_RESTORE_PREPARATION_TIME = 316,
    /* enumerations 317-323 are used in Addendum j to ANSI/ASHRAE
135-2004 */
    PROP_USER_INFORMATION_REFERENCE = 317,
    PROP_USER_NAME = 318,
    PROP_USER_TYPE = 319,
    PROP_USES_REMAINING = 320,
    PROP_VENDOR_FORMAT_IDENTIFIER = 321,
    PROP_ZONE_FROM = 322,
    PROP_ZONE_TO = 323,
    /* enumerations 324-325 are used in Addendum i to ANSI/ASHRAE
135-2004 */
    PROP_BINARY_ACTIVE_VALUE = 324,
    PROP_BINARY_INACTIVE_VALUE = 325
        /* The special property identifiers all, optional, and required
*/
        /* are reserved for use in the ReadPropertyConditional and */
        /* ReadPropertyMultiple services or services not defined in this
standard. */
        /* Enumerated values 0-511 are reserved for definition by
ASHRAE.  */
        /* Enumerated values 512-4194303 may be used by others subject to
the  */
        /* procedures and constraints described in Clause 23.  */
} BACNET_PROPERTY_ID;
```

F.2 BACnet Engineering Units Codes

BACnet engineering units codes may be found in your copy of the BACnet protocol specification, ANSI/ASHRAE Standard 135. That document is copyrighted, but the C enumeration shown below for reference is taken from open source code available under GPL at http://sourceforge.net, and provides essentially the same information (copyrighted by Steve Karg, licensed under GPL as noted at http://sourceforge.net).

```
typedef enum {
    /* Acceleration */
    UNITS_METERS_PER_SECOND_PER_SECOND = 166,
    /* Area */
    UNITS_SQUARE_METERS = 0,
    UNITS_SQUARE_CENTIMETERS = 116,
    UNITS_SQUARE_FEET = 1,
    UNITS_SQUARE_INCHES = 115,
    /* Currency */
    UNITS_CURRENCY1 = 105,
    UNITS_CURRENCY2 = 106,
    UNITS_CURRENCY3 = 107,
    UNITS_CURRENCY4 = 108,
```

```
UNITS_CURRENCY5 = 109,
UNITS_CURRENCY6 = 110,
UNITS_CURRENCY7 = 111,
UNITS_CURRENCY8 = 112,
UNITS_CURRENCY9 = 113,
UNITS_CURRENCY10 = 114,
/* Electrical */
UNITS_MILLIAMPERES = 2,
UNITS_AMPERES = 3,
UNITS_AMPERES_PER_METER = 167,
UNITS_AMPERES_PER_SQUARE_METER = 168,
UNITS_AMPERE_SQUARE_METERS = 169,
UNITS_FARADS = 170,
UNITS_HENRYS = 171,
UNITS_OHMS = 4,
UNITS_OHM_METERS = 172,
UNITS_MILLIOHMS = 145,
UNITS_KILOHMS = 122,
UNITS_MEGOHMS = 123,
UNITS_SIEMENS = 173, /* 1 mho equals 1 siemens */
UNITS_SIEMENS_PER_METER = 174,
UNITS_TESLAS = 175,
UNITS_VOLTS = 5,
UNITS_MILLIVOLTS = 124,
UNITS_KILOVOLTS = 6,
UNITS_MEGAVOLTS = 7,
UNITS_VOLT_AMPERES = 8,
UNITS_KILOVOLT_AMPERES = 9,
UNITS_MEGAVOLT_AMPERES = 10,
UNITS_VOLT_AMPERES_REACTIVE = 11,
UNITS_KILOVOLT_AMPERES_REACTIVE = 12,
UNITS_MEGAVOLT_AMPERES_REACTIVE = 13,
UNITS_VOLTS_PER_DEGREE_KELVIN = 176,
UNITS_VOLTS_PER_METER = 177,
UNITS_DEGREES_PHASE = 14,
UNITS_POWER_FACTOR = 15,
UNITS_WEBERS = 178,
/* Energy */
UNITS_JOULES = 16,
UNITS_KILOJOULES = 17,
UNITS_KILOJOULES_PER_KILOGRAM = 125,
UNITS_MEGAJOULES = 126,
UNITS_WATT_HOURS = 18,
UNITS_KILOWATT_HOURS = 19,
UNITS_MEGAWATT_HOURS = 146,
UNITS_BTUS = 20,
UNITS_KILO_BTUS = 147,
UNITS_MEGA_BTUS = 148,
UNITS_THERMS = 21,
UNITS_TON_HOURS = 22,
/* Enthalpy */
```

```
UNITS_JOULES_PER_KILOGRAM_DRY_AIR = 23,
UNITS_KILOJOULES_PER_KILOGRAM_DRY_AIR = 149,
UNITS_MEGAJOULES_PER_KILOGRAM_DRY_AIR = 150,
UNITS_BTUS_PER_POUND_DRY_AIR = 24,
UNITS_BTUS_PER_POUND = 117,
/* Entropy */
UNITS_JOULES_PER_DEGREE_KELVIN = 127,
UNITS_KILOJOULES_PER_DEGREE_KELVIN = 151,
UNITS_MEGAJOULES_PER_DEGREE_KELVIN = 152,
UNITS_JOULES_PER_KILOGRAM_DEGREE_KELVIN = 128,
/* Force */
UNITS_NEWTON = 153,
/* Frequency */
UNITS_CYCLES_PER_HOUR = 25,
UNITS_CYCLES_PER_MINUTE = 26,
UNITS_HERTZ = 27,
UNITS_KILOHERTZ = 129,
UNITS_MEGAHERTZ = 130,
UNITS_PER_HOUR = 131,
/* Humidity */
UNITS_GRAMS_OF_WATER_PER_KILOGRAM_DRY_AIR = 28,
UNITS_PERCENT_RELATIVE_HUMIDITY = 29,
/* Length */
UNITS_MILLIMETERS = 30,
UNITS_CENTIMETERS = 118,
UNITS_METERS = 31,
UNITS_INCHES = 32,
UNITS_FEET = 33,
/* Light */
UNITS_CANDELAS = 179,
UNITS_CANDELAS_PER_SQUARE_METER = 180,
UNITS_WATTS_PER_SQUARE_FOOT = 34,
UNITS_WATTS_PER_SQUARE_METER = 35,
UNITS_LUMENS = 36,
UNITS_LUXES = 37,
UNITS_FOOT_CANDLES = 38,
/* Mass */
UNITS_KILOGRAMS = 39,
UNITS_POUNDS_MASS = 40,
UNITS_TONS = 41,
/* Mass Flow */
UNITS_GRAMS_PER_SECOND = 154,
UNITS_GRAMS_PER_MINUTE = 155,
UNITS_KILOGRAMS_PER_SECOND = 42,
UNITS_KILOGRAMS_PER_MINUTE = 43,
UNITS_KILOGRAMS_PER_HOUR = 44,
UNITS_POUNDS_MASS_PER_SECOND = 119,
UNITS_POUNDS_MASS_PER_MINUTE = 45,
UNITS_POUNDS_MASS_PER_HOUR = 46,
UNITS_TONS_PER_HOUR = 156,
/* Power */
```

```
UNITS_MILLIWATTS = 132,
UNITS_WATTS = 47,
UNITS_KILOWATTS = 48,
UNITS_MEGAWATTS = 49,
UNITS_BTUS_PER_HOUR = 50,
UNITS_KILO_BTUS_PER_HOUR = 157,
UNITS_HORSEPOWER = 51,
UNITS_TONS_REFRIGERATION = 52,
/* Pressure */
UNITS_PASCALS = 53,
UNITS_HECTOPASCALS = 133,
UNITS_KILOPASCALS = 54,
UNITS_MILLIBARS = 134,
UNITS_BARS = 55,
UNITS_POUNDS_FORCE_PER_SQUARE_INCH = 56,
UNITS_CENTIMETERS_OF_WATER = 57,
UNITS_INCHES_OF_WATER = 58,
UNITS_MILLIMETERS_OF_MERCURY = 59,
UNITS_CENTIMETERS_OF_MERCURY = 60,
UNITS_INCHES_OF_MERCURY = 61,
/* Temperature */
UNITS_DEGREES_CELSIUS = 62,
UNITS_DEGREES_KELVIN = 63,
UNITS_DEGREES_KELVIN_PER_HOUR = 181,
UNITS_DEGREES_KELVIN_PER_MINUTE = 182,
UNITS_DEGREES_FAHRENHEIT = 64,
UNITS_DEGREE_DAYS_CELSIUS = 65,
UNITS_DEGREE_DAYS_FAHRENHEIT = 66,
UNITS_DELTA_DEGREES_FAHRENHEIT = 120,
UNITS_DELTA_DEGREES_KELVIN = 121,
/* Time */
UNITS_YEARS = 67,
UNITS_MONTHS = 68,
UNITS_WEEKS = 69,
UNITS_DAYS = 70,
UNITS_HOURS = 71,
UNITS_MINUTES = 72,
UNITS_SECONDS = 73,
UNITS_HUNDREDTHS_SECONDS = 158,
UNITS_MILLISECONDS = 159,
/* Torque */
UNITS_NEWTON_METERS = 160,
/* Velocity */
UNITS_MILLIMETERS_PER_SECOND = 161,
UNITS_MILLIMETERS_PER_MINUTE = 162,
UNITS_METERS_PER_SECOND = 74,
UNITS_METERS_PER_MINUTE = 163,
UNITS_METERS_PER_HOUR = 164,
UNITS_KILOMETERS_PER_HOUR = 75,
UNITS_FEET_PER_SECOND = 76,
UNITS_FEET_PER_MINUTE = 77,
```

```
    UNITS_MILES_PER_HOUR = 78,
    /* Volume */
    UNITS_CUBIC_FEET = 79,
    UNITS_CUBIC_METERS = 80,
    UNITS_IMPERIAL_GALLONS = 81,
    UNITS_LITERS = 82,
    UNITS_US_GALLONS = 83,
    /* Volumetric Flow */
    UNITS_CUBIC_FEET_PER_SECOND = 142,
    UNITS_CUBIC_FEET_PER_MINUTE = 84,
    UNITS_CUBIC_METERS_PER_SECOND = 85,
    UNITS_CUBIC_METERS_PER_MINUTE = 165,
    UNITS_CUBIC_METERS_PER_HOUR = 135,
    UNITS_IMPERIAL_GALLONS_PER_MINUTE = 86,
    UNITS_LITERS_PER_SECOND = 87,
    UNITS_LITERS_PER_MINUTE = 88,
    UNITS_LITERS_PER_HOUR = 136,
    UNITS_US_GALLONS_PER_MINUTE = 89,
    /* Other */
    UNITS_DEGREES_ANGULAR = 90,
    UNITS_DEGREES_CELSIUS_PER_HOUR = 91,
    UNITS_DEGREES_CELSIUS_PER_MINUTE = 92,
    UNITS_DEGREES_FAHRENHEIT_PER_HOUR = 93,
    UNITS_DEGREES_FAHRENHEIT_PER_MINUTE = 94,
    UNITS_JOULE_SECONDS = 183,
    UNITS_KILOGRAMS_PER_CUBIC_METER = 186,
    UNITS_KW_HOURS_PER_SQUARE_METER = 137,
    UNITS_KW_HOURS_PER_SQUARE_FOOT = 138,
    UNITS_MEGAJOULES_PER_SQUARE_METER = 139,
    UNITS_MEGAJOULES_PER_SQUARE_FOOT = 140,
    UNITS_NO_UNITS = 95,
    UNITS_NEWTON_SECONDS = 187,
    UNITS_NEWTONS_PER_METER = 188,
    UNITS_PARTS_PER_MILLION = 96,
    UNITS_PARTS_PER_BILLION = 97,
    UNITS_PERCENT = 98,
    UNITS_PERCENT_OBSCURATION_PER_FOOT = 143,
    UNITS_PERCENT_OBSCURATION_PER_METER = 144,
    UNITS_PERCENT_PER_SECOND = 99,
    UNITS_PER_MINUTE = 100,
    UNITS_PER_SECOND = 101,
    UNITS_PSI_PER_DEGREE_FAHRENHEIT = 102,
    UNITS_RADIANS = 103,
    UNITS_RADIANS_PER_SECOND = 184,
    UNITS_REVOLUTIONS_PER_MINUTE = 104,
    UNITS_SQUARE_METERS_PER_NEWTON = 185,
    UNITS_WATTS_PER_METER_PER_DEGREE_KELVIN = 189,
    UNITS_WATTS_PER_SQUARE_METER_DEGREE_KELVIN = 141,
       ; /* Enumerated values 0-255 are reserved for definition by
ASHRAE. */
        /* Enumerated values 256-65535 may be used by others subject to
```

```
*/
        /* the procedures and constraints described in Clause 23. */
        /* The last enumeration used in this version is 189. */
    MAX_UNITS = 190
} BACNET_ENGINEERING_UNITS;
```

Appendix G Using Wireshark for Trouble Shooting

G.1 Hardware Requirements

There are no particular hardware requirements regarding the PC you run Wireshark on. Basically anything running any version of Windows can run Wireshark. There are also Linux and Mac OS X versions.

The "hardware requirement" that is of most concern is the means of connecting to the network. We typically just connect everything Ethernet to a switch and don't worry about it. However, switches are really unmanaged routers, and they filter traffic. Therefore, your PC will not see traffic passing back and forth between two other devices that are not the PC. In order to see that network traffic using Wireshark, you need to come up with the right kind of network connection.

If your PC itself is one end of the network conversation you wish to capture, for example when running a MIB browser or the Network Discovery Tool, then Wireshark will capture all network traffic to and from the PC however connected. It is when your PC wants to simply "eavesdrop" that you run into problems with the network switch.

A while back, 10BaseT hubs were common. A 10BaseT hub is not as smart as a switch and does not filter traffic. If you have an old 10BaseT hub collecting dust somewhere, you now have a new use for it. It will let Wireshark see all traffic from the PC that goes between any other devices connected to that 10BaseT hub. Beware of devices calling themselves "hubs" but support 100BaseT connections. These are switches.

Since manufacturers of hubs decided nobody should have a use for them anymore, they are generally out of production. Finding a 10BaseT hub for sale is not easy (try eBay). But there are other alternatives.

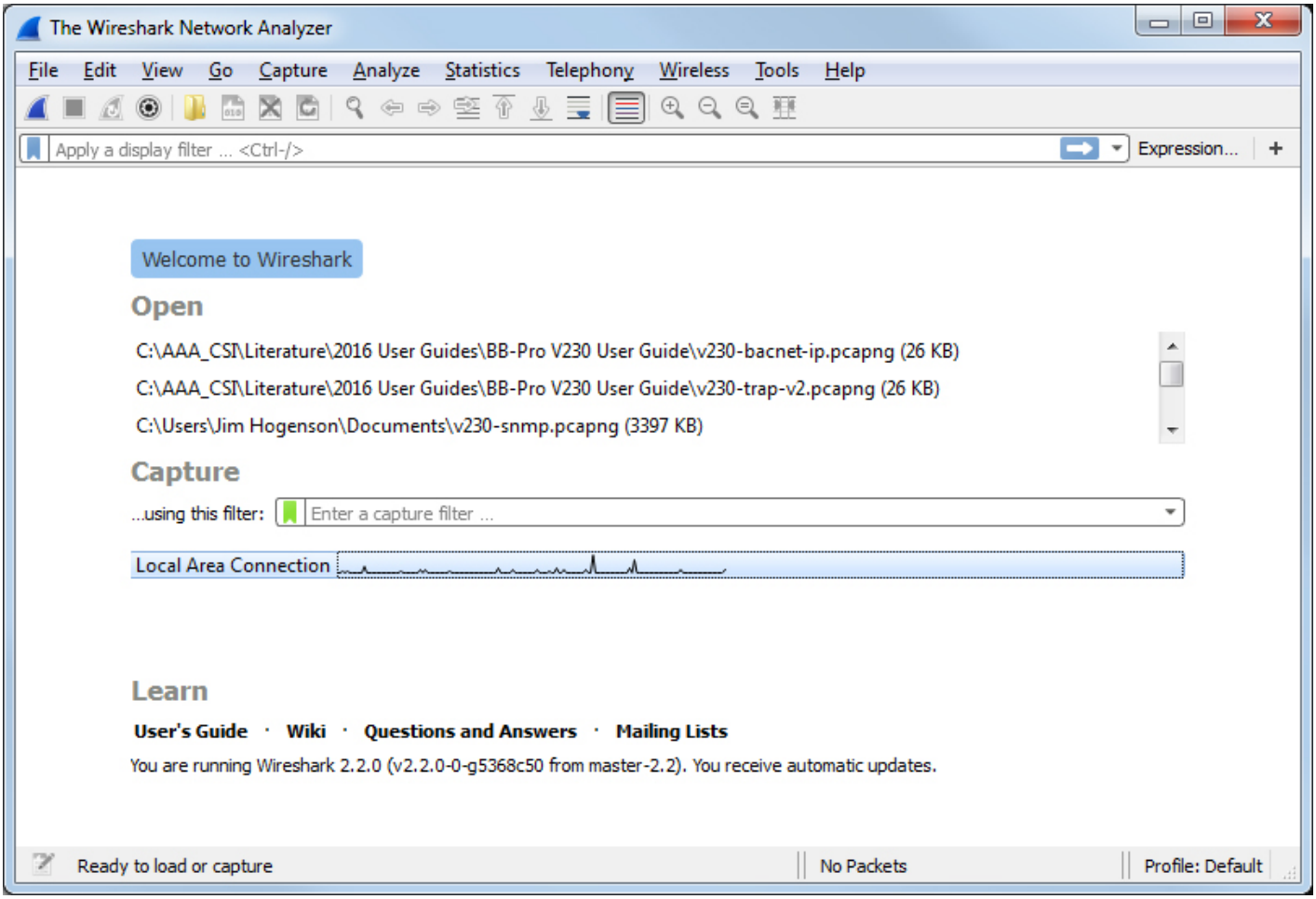
One means of monitoring network traffic is to get a managed switch that supports "port mirroring". One such device we have tested is the TP-LINK model TL-SG105E. Setting it up requires utility software (provided with the switch) and takes a little effort to get configured. But once configured, it works well without any further monkeying around. And it is inexpensive.

The other means of monitoring traffic is with the use of a device made specifically for use with Wireshark. The "SharkTap" provides two connections for the network

pass-through, and a third "tap" connection where you connect your PC running Wireshark. There is no configuration required. It is the simplest way to monitor network traffic, and it is a current production item available on Amazon (as of 2016).

G.2 Example of Using Wireshark

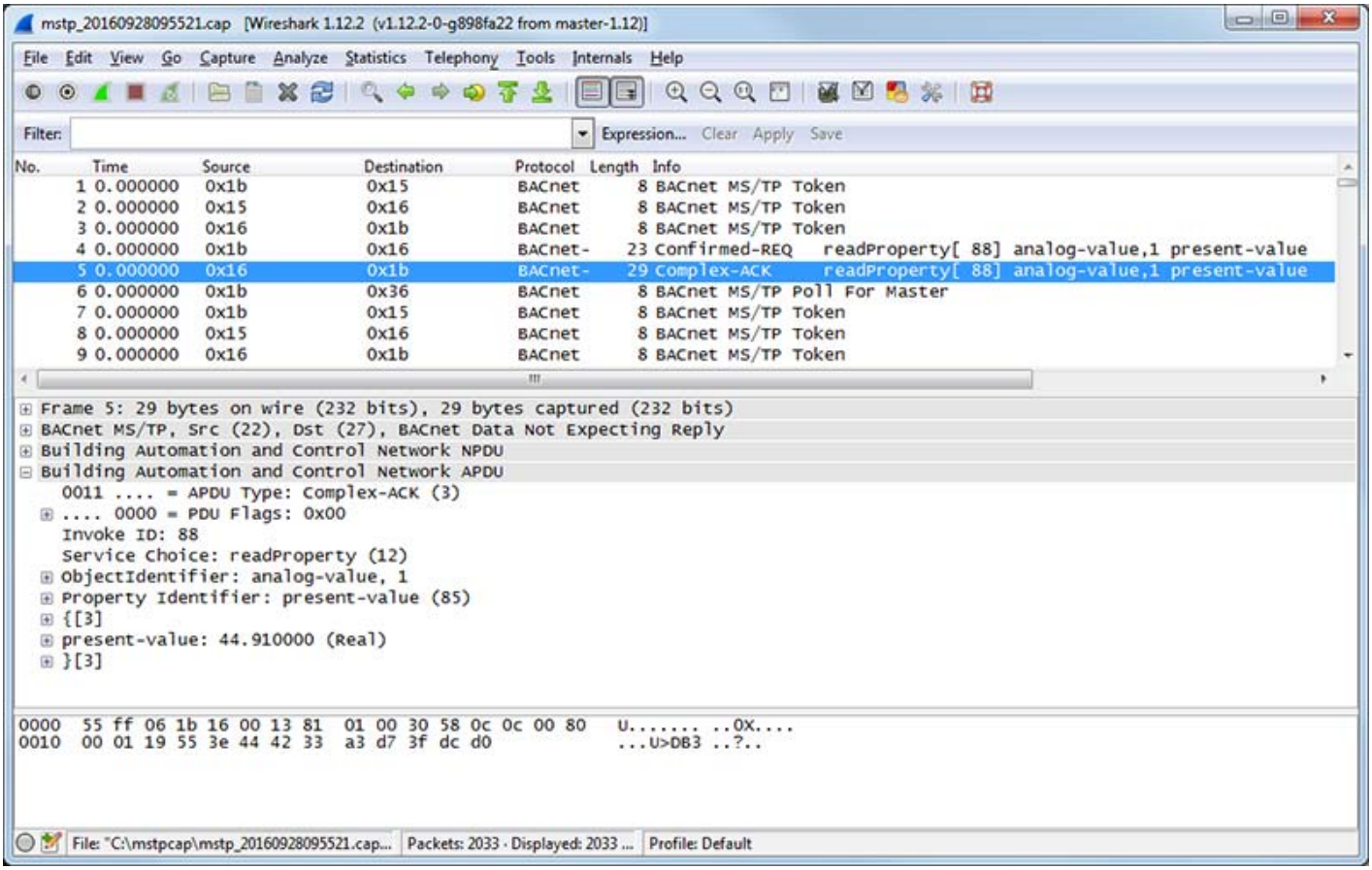
Using Wireshark is fairly easy. Get a copy at www.wireshark.org and install it. Once installed, running it is straight forward. As of version 2.2.0 of Wireshark, the startup screen looks like the following. Double click on Local Area Connection to start capturing network traffic on your PC's Ethernet port. If you have multiple network connections, they will all be listed. Be sure to select the one that represents your Ethernet connection, typically "Local Area Connection".



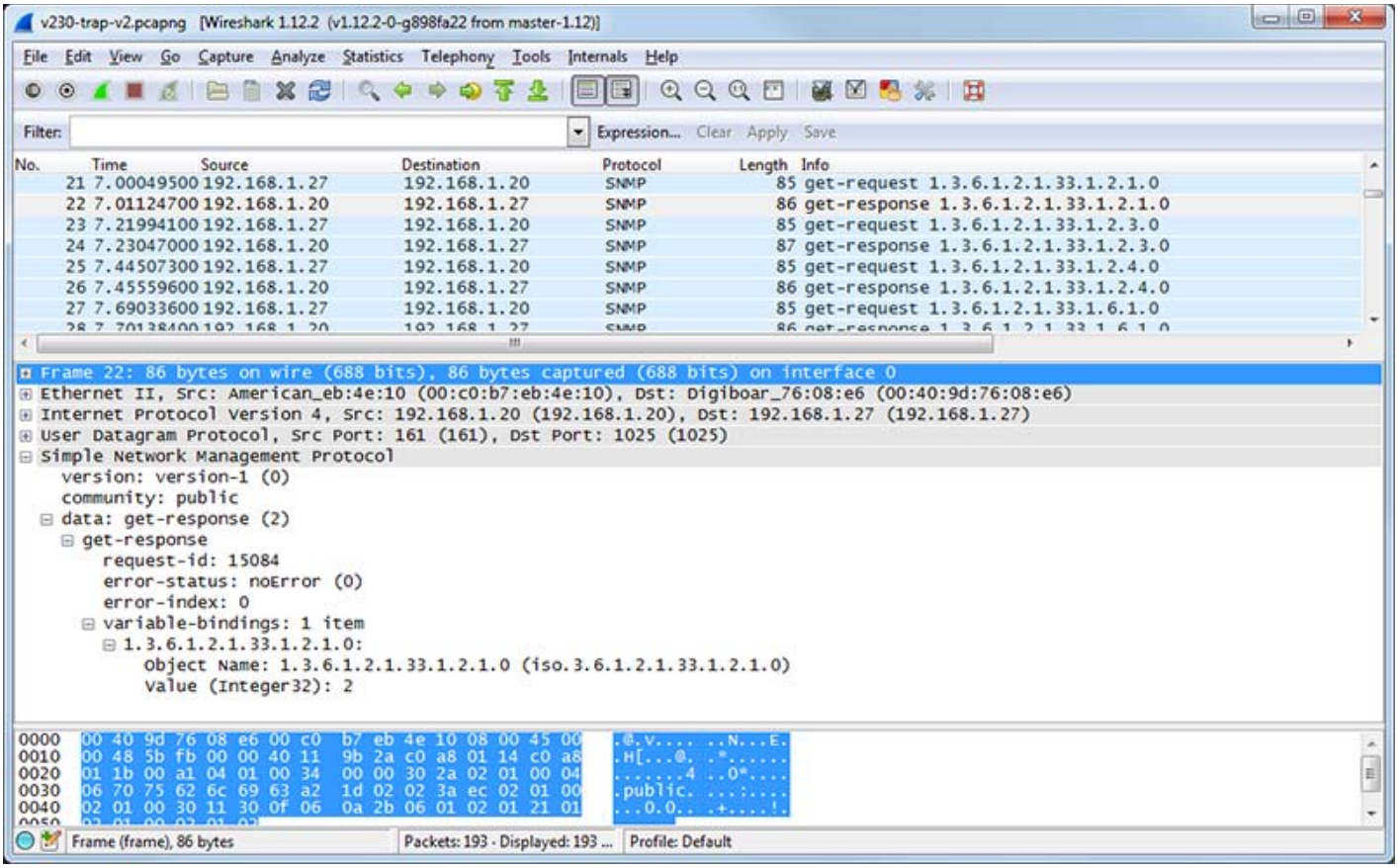
The screen will look something like the example below once Wireshark starts collecting data. Click the red icon in the toolbar to stop capturing traffic. Control Solutions technical support will often ask for a copy of the Wireshark data when a network issue seems evident. You can save a copy of all of the network traffic captured under the File menu, and you will generally save it to a .pcap or .pcapng file. A Wireshark log with .pcap extension can be posted directly as an attachment in support tickets while .pcapng needs to be zipped first.

The screen shot below shows Wireshark capturing BACnet IP traffic between a client and our Babel Buster Pro. If you click on a packet, the details of that packet will be displayed in the lower part of the screen. You can expand the tree view to see further detail.

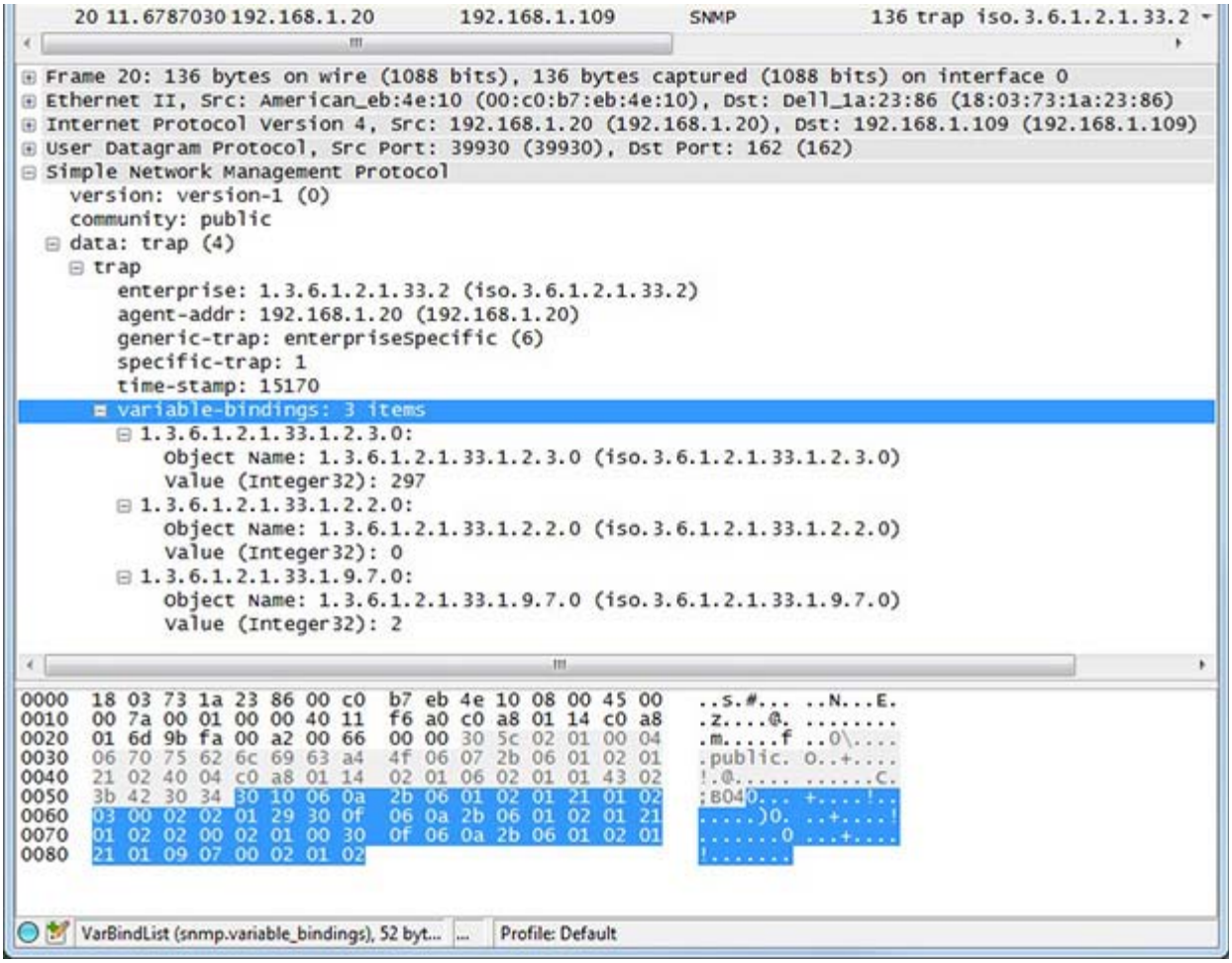
A lot of times you will see a lot of network traffic that is not of interest to you. You can filter network traffic to only display traffic to/from the device you are interested in. Do this by entering "ip.addr==192.168.1.23" in the Filter window as illustrated below. (Substitute your own device's IP address.)



Capturing a series of SNMP traffic will look like the example below. In this example, a series of Get requests is being performed by the SNMP client. Click on any one packet and expand the tree structure in the middle window to see full detail of the request or response.



If you are working on getting a trap receive rule to work, you will be interested in looking at traps in Wireshark. A trap from an RFC 1628 UPS is illustrated below, with the trap message expanded in the tree view, as well as the varbinds expanded to show OID and value.


20 11.6787030 192.168.1.20 192.168.1.109 SNMP 136 trap iso.3.6.1.2.1.33.2
Frame 20: 136 bytes on wire (1088 bits), 136 bytes captured (1088 bits) on interface 0
Ethernet II, Src: American_eb:4e:10 (00:c0:b7:eb:4e:10), Dst: Dell_1a:23:86 (18:03:73:1a:23:86)
Internet Protocol Version 4, Src: 192.168.1.20 (192.168.1.20), Dst: 192.168.1.109 (192.168.1.109)
User Datagram Protocol, Src Port: 39930 (39930), Dst Port: 162 (162)
Simple Network Management Protocol
version: version-1 (0)
community: public
data: trap (4)
trap
enterprise: 1.3.6.1.2.1.33.2 (iso.3.6.1.2.1.33.2)
agent-addr: 192.168.1.20 (192.168.1.20)
generic-trap: enterpriseSpecific (6)
specific-trap: 1
time-stamp: 15170
variable-bindings: 3 items
1.3.6.1.2.1.33.1.2.3.0:
Object Name: 1.3.6.1.2.1.33.1.2.3.0 (iso.3.6.1.2.1.33.1.2.3.0)
Value (Integer32): 297
1.3.6.1.2.1.33.1.2.2.0:
Object Name: 1.3.6.1.2.1.33.1.2.2.0 (iso.3.6.1.2.1.33.1.2.2.0)
Value (Integer32): 0
1.3.6.1.2.1.33.1.9.7.0:
Object Name: 1.3.6.1.2.1.33.1.9.7.0 (iso.3.6.1.2.1.33.1.9.7.0)
Value (Integer32): 2
0000 18 03 73 1a 23 86 00 c0 b7 eb 4e 10 08 00 45 00 ..s.#... ..N...E.
0010 00 7a 00 01 00 00 40 11 f6 a0 c0 a8 01 14 c0 a8 .z....@. ....
0020 01 6d 9b fa 00 a2 00 66 00 00 30 5c 02 01 00 04 .m.....f ..0\....
0030 06 70 75 62 6c 69 63 a4 4f 06 07 2b 06 01 02 01 .public. O..+....
0040 21 02 40 04 c0 a8 01 14 02 01 06 02 01 01 43 02 !.@..... ......C.
0050 3b 42 30 34 30 10 06 0a 2b 06 01 02 01 21 01 02 ;B040... +....!..
0060 03 00 02 02 01 29 30 0f 06 0a 2b 06 01 02 01 21 .....)0. ..+....!
0070 01 02 02 00 02 01 00 30 0f 06 0a 2b 06 01 02 01 ....0 ...+....
0080 21 01 09 07 00 02 01 02 !.......
VarBindList (snmp.variable_bindings), 52 byt...
Profile: Default