Using Wireshark for Trouble Shooting

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 ManageEngine 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.

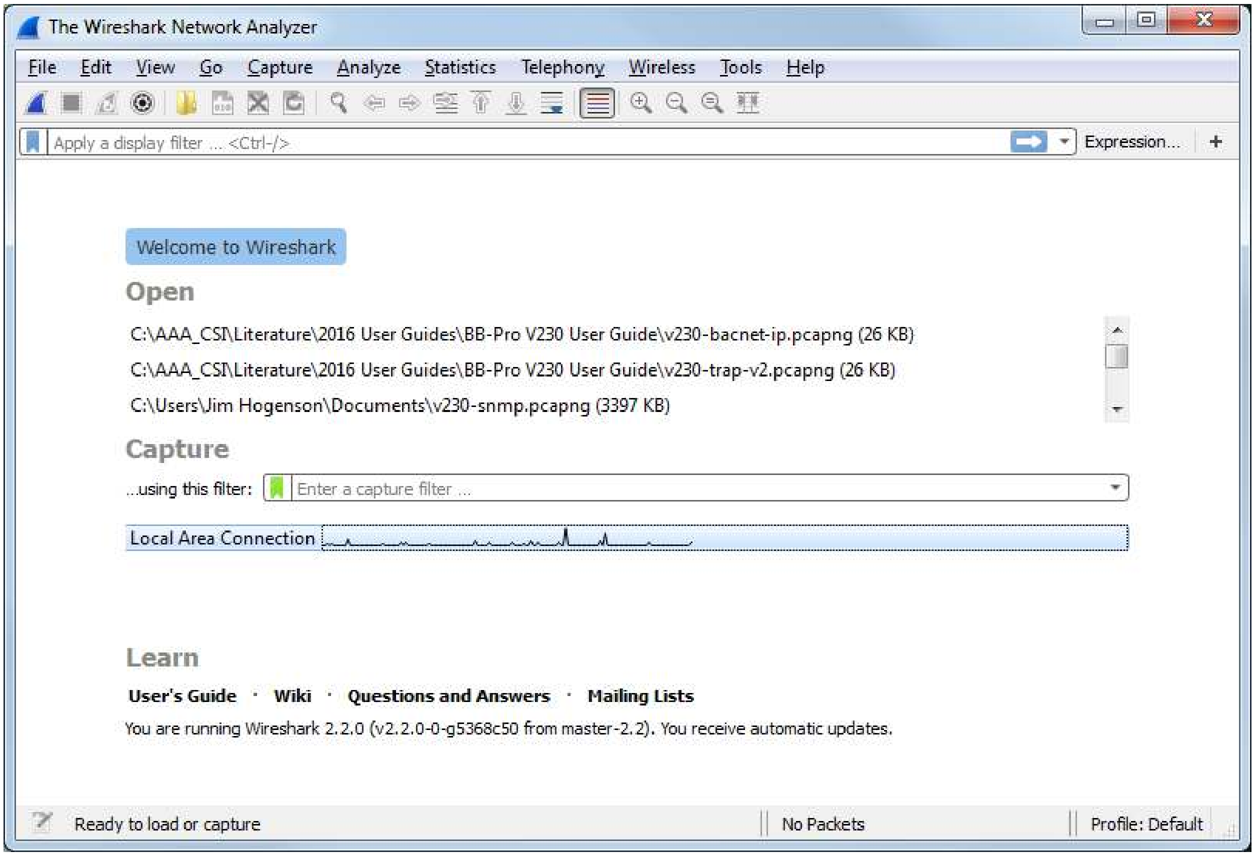
Since manufacturers of hubs decided nobody should have a use for them anymore, they are generally out of production. A list of devices that have been tested can be found here: https://wiki.wireshark.org/HubReference. (Pay attention to comments. This is a list of tested devices, not strictly working devices, meaning some are tested and reported to not work with Wireshark.) Some of the devices listed can still be found on Amazon or eBay. Finding a 10BaseT hub for sale may require a little searching, 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).

Examples 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".



C:\AAA_CSI\Literature\2016 User Guides\BB-Pro V230 User Guide\v230-bacnet-ip.pcapng (26 KB)
C:\AAA_CSI\Literature\2016 User Guides\BB-Pro V230 User Guide\v230-trap-v2.pcapng (26 KB)
C:\Users\Jim Hogenson\Documents\v230-snmp.pcapng (3397 KB)

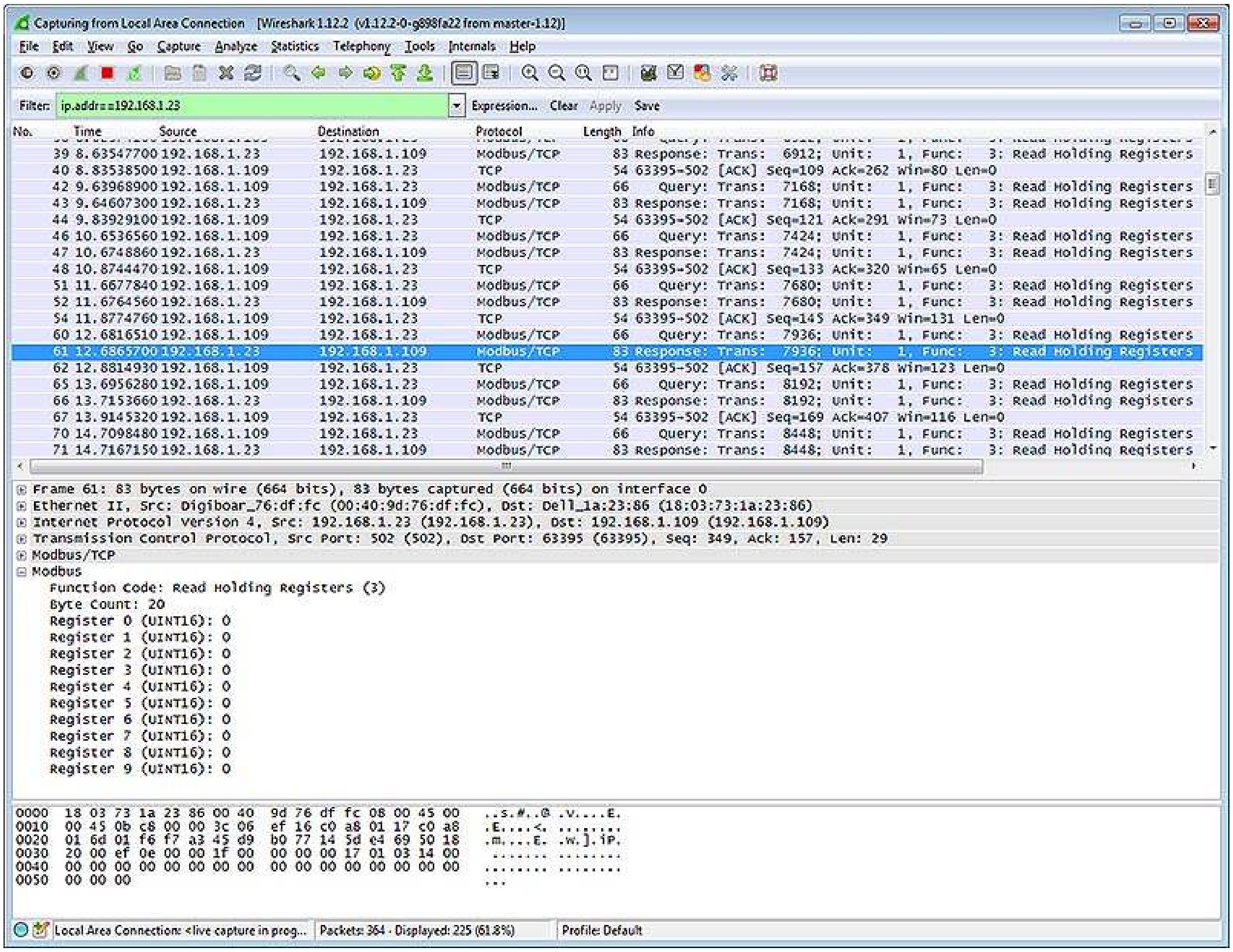
You are running Wireshark 2.2.0 (v2.2.0-0-g5368c50 from master-2.2). You receive automatic updates.

Ready to load or capture | No Packets | Profile: Default

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 Modbus TCP traffic between a client and our Babel Buster Pro V210. 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. In the case of Modbus, we will see function code, register count or data count, etc.

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 from Local Area Connection [Wireshark 1.12.2 (v1.12.2-0-g898fa22 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: ip.addr==192.168.1.23

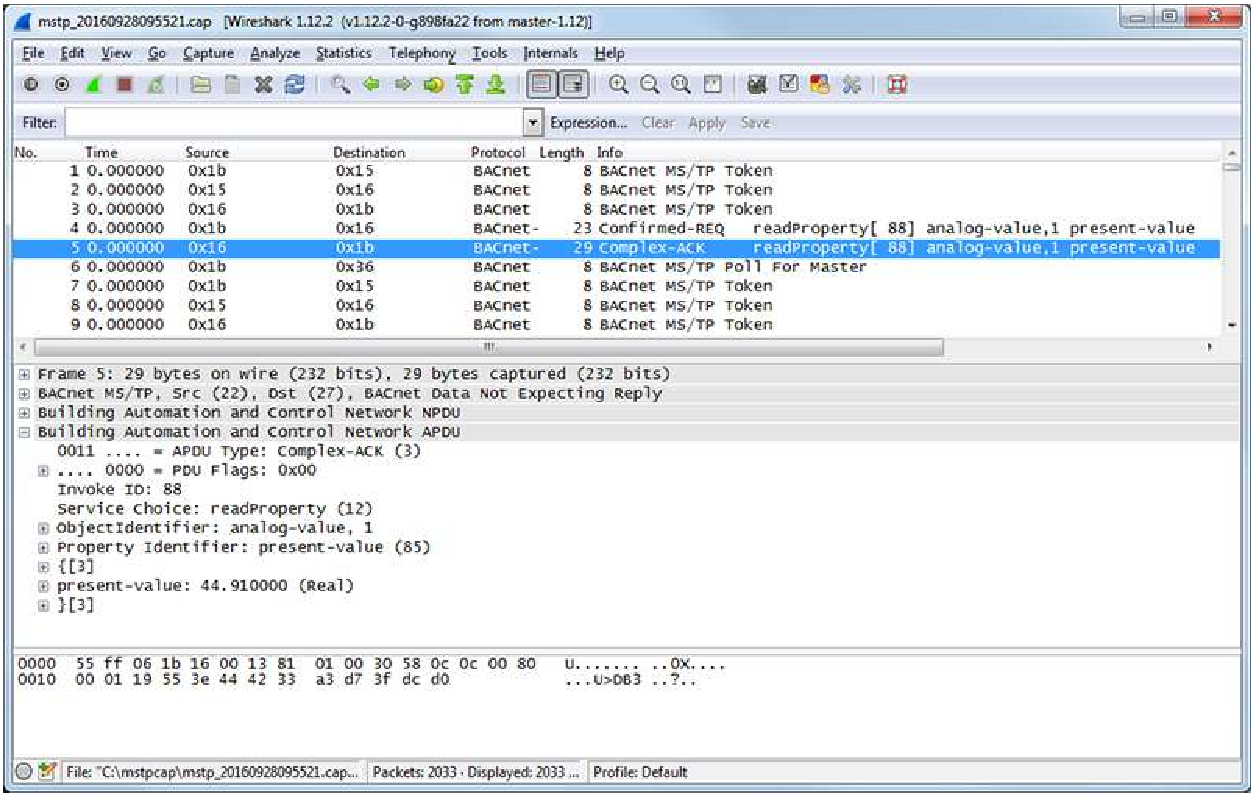
```
No.  Time        Source         Destination    Protocol    Length Info
  39 8.63547700 192.168.1.23   192.168.1.109  Modbus/TCP      83 Response: Trans:  6912; Unit:   1, Func:   3: Read Holding Registers
  40 8.83538500 192.168.1.109  192.168.1.23   TCP             54 63395-502 [ACK] Seq=109 Ack=262 Win=80 Len=0
  42 9.63968900 192.168.1.109  192.168.1.23   Modbus/TCP      66    Query: Trans:  7168; Unit:   1, Func:   3: Read Holding Registers
  43 9.64607300 192.168.1.23   192.168.1.109  Modbus/TCP      83 Response: Trans:  7168; Unit:   1, Func:   3: Read Holding Registers
  44 9.83929100 192.168.1.109  192.168.1.23   TCP             54 63395-502 [ACK] Seq=121 Ack=291 Win=73 Len=0
  46 10.6536560 192.168.1.109  192.168.1.23   Modbus/TCP      66    Query: Trans:  7424; Unit:   1, Func:   3: Read Holding Registers
  47 10.6748860 192.168.1.23   192.168.1.109  Modbus/TCP      83 Response: Trans:  7424; Unit:   1, Func:   3: Read Holding Registers
  48 10.8744470 192.168.1.109  192.168.1.23   TCP             54 63395-502 [ACK] Seq=133 Ack=320 Win=65 Len=0
  51 11.6677840 192.168.1.109  192.168.1.23   Modbus/TCP      66    Query: Trans:  7680; Unit:   1, Func:   3: Read Holding Registers
  52 11.6764560 192.168.1.23   192.168.1.109  Modbus/TCP      83 Response: Trans:  7680; Unit:   1, Func:   3: Read Holding Registers
  54 11.8774760 192.168.1.109  192.168.1.23   TCP             54 63395-502 [ACK] Seq=145 Ack=349 Win=131 Len=0
  60 12.6816510 192.168.1.109  192.168.1.23   Modbus/TCP      66    Query: Trans:  7936; Unit:   1, Func:   3: Read Holding Registers
  61 12.6865700 192.168.1.23   192.168.1.109  Modbus/TCP      83 Response: Trans:  7936; Unit:   1, Func:   3: Read Holding Registers
  62 12.8814930 192.168.1.109  192.168.1.23   TCP             54 63395-502 [ACK] Seq=157 Ack=378 Win=123 Len=0
  65 13.6956280 192.168.1.109  192.168.1.23   Modbus/TCP      66    Query: Trans:  8192; Unit:   1, Func:   3: Read Holding Registers
  66 13.7153660 192.168.1.23   192.168.1.109  Modbus/TCP      83 Response: Trans:  8192; Unit:   1, Func:   3: Read Holding Registers
  67 13.9145320 192.168.1.109  192.168.1.23   TCP             54 63395-502 [ACK] Seq=169 Ack=407 Win=116 Len=0
  70 14.7098480 192.168.1.109  192.168.1.23   Modbus/TCP      66    Query: Trans:  8448; Unit:   1, Func:   3: Read Holding Registers
  71 14.7167150 192.168.1.23   192.168.1.109  Modbus/TCP      83 Response: Trans:  8448; Unit:   1, Func:   3: Read Holding Registers
```

```
⊞ Frame 61: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
⊞ Ethernet II, Src: Digiboar_76:df:fc (00:40:9d:76:df:fc), Dst: Dell_1a:23:86 (18:03:73:1a:23:86)
⊞ Internet Protocol Version 4, Src: 192.168.1.23 (192.168.1.23), Dst: 192.168.1.109 (192.168.1.109)
⊞ Transmission Control Protocol, Src Port: 502 (502), Dst Port: 63395 (63395), Seq: 349, Ack: 157, Len: 29
⊞ Modbus/TCP
⊟ Modbus
    Function Code: Read Holding Registers (3)
    Byte Count: 20
    Register 0 (UINT16): 0
    Register 1 (UINT16): 0
    Register 2 (UINT16): 0
    Register 3 (UINT16): 0
    Register 4 (UINT16): 0
    Register 5 (UINT16): 0
    Register 6 (UINT16): 0
    Register 7 (UINT16): 0
    Register 8 (UINT16): 0
    Register 9 (UINT16): 0
```

```
0000  18 03 73 1a 23 86 00 40  9d 76 df fc 08 00 45 00   ..s.#..@ .v....E.
0010  00 45 0b c8 00 00 3c 06  ef 16 c0 a8 01 17 c0 a8   .E....<. ........
0020  01 6d 01 f6 f7 a3 45 d9  b0 77 14 5d e4 69 50 18   .m....E. .w.].iP.
0030  20 00 ef 0e 00 00 1f 00  00 00 00 17 01 03 14 00    ....... ........
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 00                                           ...
```

Local Area Connection: <live capture in prog... | Packets: 364 · Displayed: 225 (61.8%) | Profile: Default

The screen shot below shows Wireshark capturing BACnet IP traffic between a client and our Babel Buster Pro V230. 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.



mstp_20160928095521.cap [Wireshark 1.12.2 (v1.12.2-0-g898fa22 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

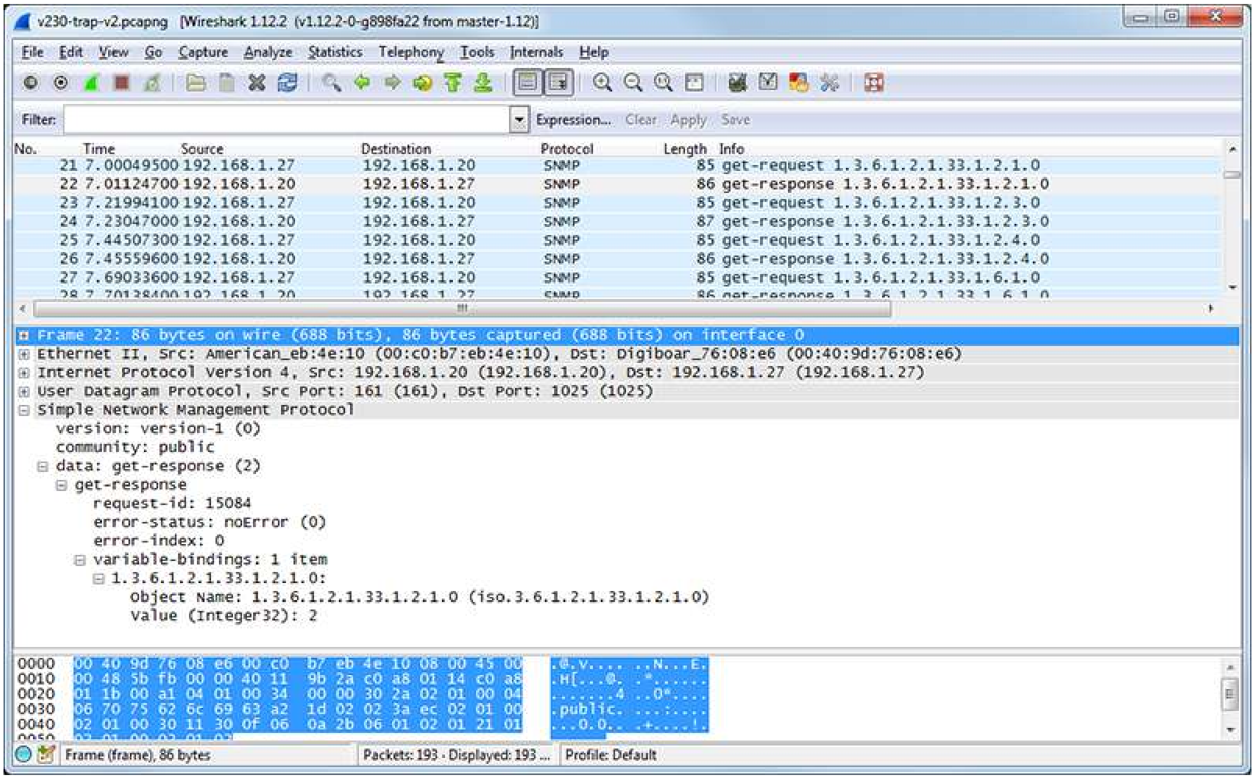
```
No.     Time      Source        Destination    Protocol Length Info
      1 0.000000  0x1b          0x15           BACnet      8 BACnet MS/TP Token
      2 0.000000  0x15          0x16           BACnet      8 BACnet MS/TP Token
      3 0.000000  0x16          0x1b           BACnet      8 BACnet MS/TP Token
      4 0.000000  0x1b          0x16           BACnet-    23 Confirmed-REQ   readProperty[ 88] analog-value,1 present-value
      5 0.000000  0x16          0x1b           BACnet-    29 Complex-ACK     readProperty[ 88] analog-value,1 present-value
      6 0.000000  0x1b          0x36           BACnet      8 BACnet MS/TP Poll For Master
      7 0.000000  0x1b          0x15           BACnet      8 BACnet MS/TP Token
      8 0.000000  0x15          0x16           BACnet      8 BACnet MS/TP Token
      9 0.000000  0x16          0x1b           BACnet      8 BACnet MS/TP Token
```

```
⊞ Frame 5: 29 bytes on wire (232 bits), 29 bytes captured (232 bits)
⊞ BACnet MS/TP, Src (22), Dst (27), BACnet Data Not Expecting Reply
⊞ Building Automation and Control Network NPDU
⊟ Building Automation and Control Network APDU
    0011 .... = APDU Type: Complex-ACK (3)
  ⊞ .... 0000 = PDU Flags: 0x00
    Invoke ID: 88
    Service Choice: readProperty (12)
  ⊞ ObjectIdentifier: analog-value, 1
  ⊞ Property Identifier: present-value (85)
  ⊞ {[3]
  ⊞ present-value: 44.910000 (Real)
  ⊞ }[3]
```

```
0000   55 ff 06 1b 16 00 13 81  01 00 30 58 0c 0c 00 80   U....... ..0X....
0010   00 01 19 55 3e 44 42 33  a3 d7 3f dc d0            ...U>DB3 ..?..
```

File: "C:\mstpcap\mstp_20160928095521.cap... | Packets: 2033 · Displayed: 2033 ... | Profile: Default

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.

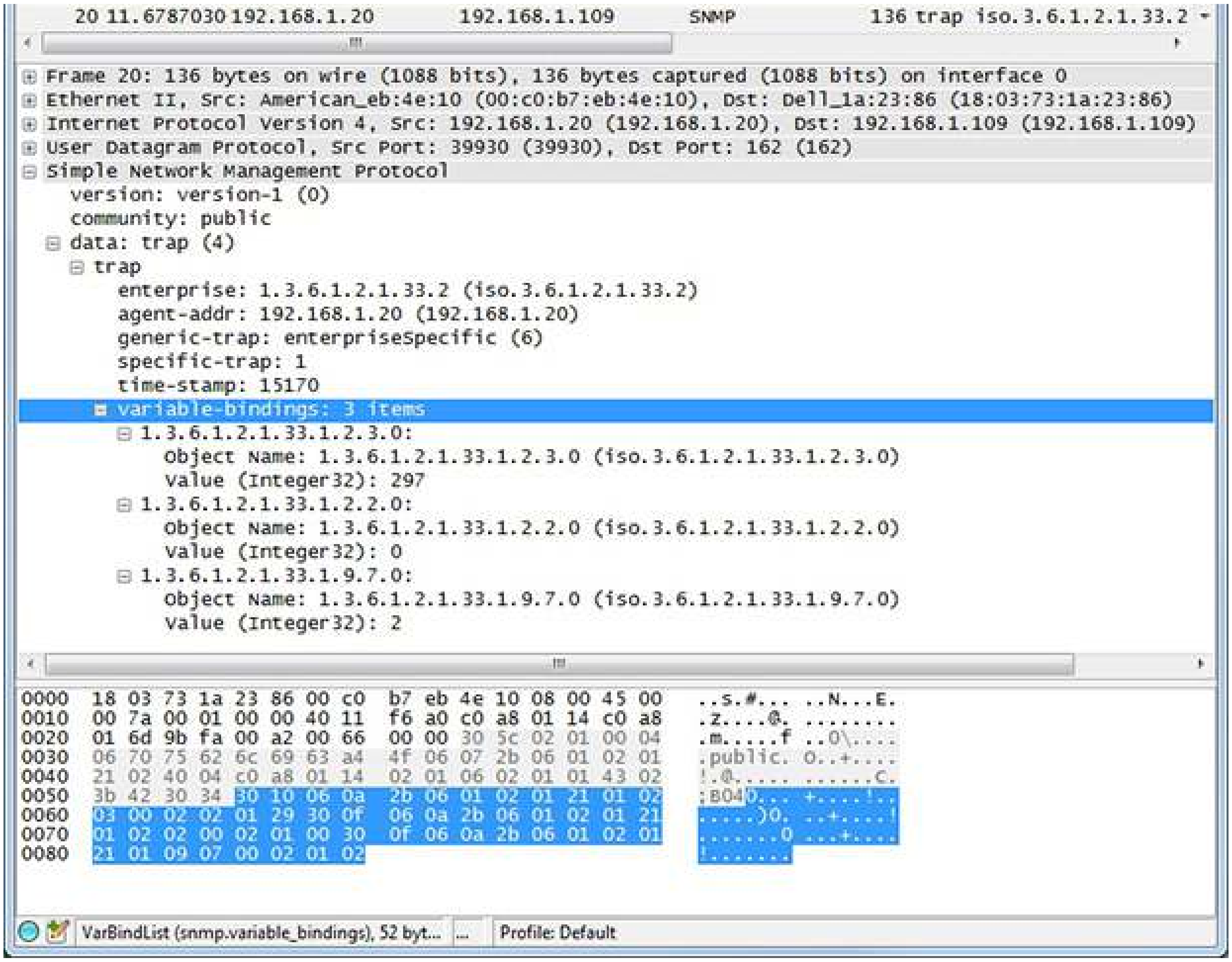

v230-trap-v2.pcapng [Wireshark 1.12.2 (v1.12.2-0-g898fa22 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time       Source         Destination    Protocol  Length Info
   21 7.00049500 192.168.1.27   192.168.1.20   SNMP          85 get-request 1.3.6.1.2.1.33.1.2.1.0
   22 7.01124700 192.168.1.20   192.168.1.27   SNMP          86 get-response 1.3.6.1.2.1.33.1.2.1.0
   23 7.21994100 192.168.1.27   192.168.1.20   SNMP          85 get-request 1.3.6.1.2.1.33.1.2.3.0
   24 7.23047000 192.168.1.20   192.168.1.27   SNMP          87 get-response 1.3.6.1.2.1.33.1.2.3.0
   25 7.44507300 192.168.1.27   192.168.1.20   SNMP          85 get-request 1.3.6.1.2.1.33.1.2.4.0
   26 7.45559600 192.168.1.20   192.168.1.27   SNMP          86 get-response 1.3.6.1.2.1.33.1.2.4.0
   27 7.69033600 192.168.1.27   192.168.1.20   SNMP          85 get-request 1.3.6.1.2.1.33.1.6.1.0
   28 7.70138400 192.168.1.20   192.168.1.27   SNMP          86 get-response 1.3.6.1.2.1.33.1.6.1.0
```

```
⊞ Frame 22: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
⊞ Ethernet II, Src: American_eb:4e:10 (00:c0:b7:eb:4e:10), Dst: Digiboar_76:08:e6 (00:40:9d:76:08:e6)
⊞ Internet Protocol Version 4, Src: 192.168.1.20 (192.168.1.20), Dst: 192.168.1.27 (192.168.1.27)
⊞ User Datagram Protocol, Src Port: 161 (161), Dst Port: 1025 (1025)
⊟ Simple Network Management Protocol
    version: version-1 (0)
    community: public
  ⊟ data: get-response (2)
    ⊟ get-response
        request-id: 15084
        error-status: noError (0)
        error-index: 0
      ⊟ variable-bindings: 1 item
        ⊟ 1.3.6.1.2.1.33.1.2.1.0:
            Object Name: 1.3.6.1.2.1.33.1.2.1.0 (iso.3.6.1.2.1.33.1.2.1.0)
            Value (Integer32): 2
```

```
0000  00 40 9d 76 08 e6 00 c0  b7 eb 4e 10 08 00 45 00   .@.v.... ..N...E.
0010  00 48 5b fb 00 00 40 11  9b 2a c0 a8 01 14 c0 a8   .H[...@. .*......
0020  01 1b 00 a1 04 01 00 34  00 00 30 2a 02 01 00 04   .......4 ..0*....
0030  06 70 75 62 6c 69 63 a2  1d 02 02 3a ec 02 01 00   .public. ...:....
0040  02 01 00 30 11 30 0f 06  0a 2b 06 01 02 01 21 01   ...0.0.. .+....!.
0050  02 01 00 02 01 02
```

If you are configuring a device to send traps, you may want to look at traps with Wireshark. If you are working on getting a trap receive rule to work in a Babel Buster Pro, you will be interested in looking at traps in Wireshark there, too. 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
```

Article ID: 36
Created On: Tue, Dec 6, 2016 at 9:28 PM
Last Updated On: Thu, Feb 9, 2017 at 9:19 AM