

BACnet

Filter	What it isolates
<code>bacnet bacapp</code>	All BACnet application-layer traffic
<code>bvlc bacnet bacapp</code>	Same, plus BBMD broadcast-management traffic
<code>udp.port == 47808</code>	All standard-port BACnet/IP traffic
<code>bacapp.type > 4</code>	Errors, Rejects and Aborts only — fastest way to find a problem
<code>bacapp.confirmed_service == 15</code>	WriteProperty requests specifically
<code>bacapp.confirmed_service == 12</code>	ReadProperty requests specifically
<code>bacapp.objectType == 0</code>	Traffic touching any Analog Input object
<code>bacapp.instance_number == 1</code>	Traffic touching instance 1 (combine with objectType)

BACnet object type codes (bacapp.objectType)

0 Analog Input 1 Analog Output 2 Analog Value 3 Binary Input 4 Binary Output 5 Binary Value 6 Calendar 7 Command 8 Device 9 Event Enrollment 10 File 11 Group 12 Loop 13 Multi-state Input 14 Multi-state Output 17 Schedule 19 Multi-state Value 20 Trend Log

Modbus

Filter	What it isolates
<code>mbtcp</code>	All Modbus/TCP traffic (once recognised)
<code>tcp.port == 502</code>	Standard-port Modbus TCP traffic
<code>mbtcp.trans_id</code>	Transaction ID — pair each response to its request
<code>modbus.exception_code</code>	Exception responses (original function code + 0x80)
<code>tcp.analysis.retransmission</code>	TCP retransmissions — a network signal, not a Modbus one

MODBUS NEEDS “DECODE AS”

There is no heuristic dissector for Modbus — it is bound to ports (502/TCP, 502/UDP, 802/TLS), not content. On a non-standard port: right-click a TCP packet → Decode As... → Modbus. An empty mbtcp filter usually means a Decode As problem, not a capture problem.

Coloring rules worth building (first match wins, order matters)

Rule	Filter
Red — BACnet errors	<code>bacapp.type > 4</code>
Orange — Modbus exceptions	<code>modbus.exception_code</code>
Green — discovery	<code>bacapp.unconfirmed_service == 8 bacapp.unconfirmed_service == 0</code>
Blue — writes	<code>bacapp.confirmed_service == 15</code>