BACnet Automation Testbench

Key Learnings & Industry Reference Guide

Python 3.13 · BAC0 v2025.09.15 · bacpypes3 v0.0.104

February 2026

1. BACnet Protocol Fundamentals

1.1 What is BACnet?

BACnet (Building Automation and Control Networks) is an ANSI/ASHRAE/ISO standard communication protocol for building automation systems. It defines how devices such as thermostats, sensors, HVAC controllers, and lighting systems exchange data over a network.

1.2 Object Model — The Core Concept

Every BACnet device exposes its data as Objects. Each object has Properties. You read/write properties, not raw memory addresses.

Object Type	Type Code	Typical Use	Example
Analog Input (AI)	0	Read-only sensor value	Temperature sensor
Analog Output (AO)	1	Writable actuator	Valve position
Analog Value (AV)	2	Writable setpoint	T Set, Setpoint
Binary Input (BI)	3	Read-only digital	Occupancy sensor
Binary Value (BV)	5	Writable on/off	Heater state
Multi-State Value (MV)	19	Enumerated states	Ventilation level

1.3 Key Properties Every Engineer Must Know

Property	BACnet Name	What It Does
Present Value	present-value	The current live value of the object
Out Of Service	out-of-service	When True, disconnects hardware; PV can be injected manually
Priority Array	priority-array	16-slot table controlling which source wins for the PV
Object Identifier	object-identifier	Unique ID: type + instance number (e.g. analog-value,0)
Object Name	object-name	Human-readable label (e.g. SetPoint.Value)
Units	units	Engineering unit (e.g. 62 = Degrees Celsius)
Reliability	reliability	0 = no-fault-detected means healthy
Status Flags	status-flags	4-bit flag: in-alarm, fault, overridden, out-of-service

1.4 The Priority Array — Critical for Testing

Commandable objects (AO, AV, BV) use a 16-level priority array. A write only succeeds if its priority level is not outranked by a higher-priority existing command.

Priority	Name	Typical Use
1	Manual Life Safety	Fire/smoke override
2	Automatic Life Safety	Automated safety system
3–7	Reserved	Critical building control
8	Manual Operator	Most common for testing/commissioning
9–15	Supervisory / Scheduling	BMS automation layers
16	Default (lowest)	Factory default fallback

⚠ **NOTE:** *Always use priority 8 for test injections. Without specifying a priority, many simulators and real controllers silently reject the write.*

1.5 Out Of Service Override Pattern

This is the standard testbench pattern to inject arbitrary values into a controller without hardware:

- **Step 1:** Write out-of-service = True — disconnects the hardware input
- **Step 2:** Write present-value = <test vector> @ priority 8 — injects your value
- **Step 3:** Read back present-value — verify the write landed
- **Step 4:** Write out-of-service = False — restore normal hardware operation after test

⚠ **NOTE:** *Never forget Step 4 in a real system. Leaving Out Of Service = True on live hardware causes the controller to run on a fake value indefinitely.*

2. BACnet/IP Network Layer

2.1 Port Architecture

BACnet/IP uses UDP, not TCP. This has major implications for discovery and routing.

Port	Role	Notes
47808 (0xBAC0)	Standard BACnet/IP port	WhoIs broadcast always goes here
Dynamic (e.g. 52025)	Simulator / software device port	Changes every restart — check Yabe each time
47810	Python testbench port	Must be different from 47808 and the DUT port

2.2 Device Discovery: WhoIs / IAm

Before any read/write, BACnet requires device discovery. The initiator broadcasts WhoIs on port 47808; the target responds with IAm. BAC0 requires this registration before it can serialize packets to a device.

⚠ **NOTE:** *Software simulators running on non-standard dynamic ports (e.g. 52025) will NEVER hear a broadcast on 47808. This is why BAC0's built-in discovery always fails for local simulators. The fix is to use bacpypes3 directly and target the device's explicit IP:port.*

2.3 The Loopback Problem on Windows

When Python and the BACnet simulator run on the same machine (same IP), Windows drops self-addressed UDP packets at the network stack level. ICMP (ping) works fine, but UDP does not — this is why ping succeeds while BACnet writes fail.

Scenario	Solution
Python and simulator on same IP, different ports	Bind Python directly to the same IP but a different port (e.g. :47810 vs :52025). Windows allows same-IP different-port UDP.
Need completely separate IP	Install Microsoft KM-TEST Loopback Adapter, assign 192.168.100.200, bind Python there
Broadcast WhoIs not reaching simulator	Skip BAC0 discovery entirely; use bacpypes3 direct mode with explicit Address target

2.4 Windows Firewall and UDP

Windows Firewall treats UDP differently from ICMP. A successful ping does NOT mean BACnet UDP packets are allowed. You must explicitly open UDP ports:

```
New-NetFirewallRule -DisplayName "BACnet IN"  -Direction Inbound  -Protocol UDP
-LocalPort 47808,47810 -Action Allow
```

```
New-NetFirewallRule -DisplayName "BACnet OUT" -Direction Outbound -Protocol UDP
-LocalPort 47808,47810 -Action Allow
```

3. Python Stack: BAC0 & bacpypes3

3.1 Library Architecture

BAC0 is a high-level wrapper built on top of bacpypes3. BAC0 Lite is the async version using bacpypes3 under the hood. When BAC0's abstractions fail (e.g. discovery broken), drop down to bacpypes3 directly.

Layer	Library	When to Use
High-level	BAC0 Lite	Standard read/write when discovery works
Mid-level	BAC0 + explicit targeting	When you know the device IP:port
Low-level	bacpypes3 direct	When discovery is broken or you need raw APDU control

3.2 Windows Python 3.13 Monkey Patch

Python 3.13 on Windows raises ValueError: reuse_port not supported by socket module when bacpypes3 tries to create a UDP socket. This MUST be patched before any bacpypes3 import:

```
import asyncio.base_events
asyncio.base_events._set_reuseport = lambda sock: None
asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())
```

⚠ **NOTE:** *The patch must come before any bacpypes3 import. If bacpypes3 is imported first, it captures the broken reference and the patch has no effect.*

3.3 BAC0 Lite API — Sync vs Async (version 2025.09.15)

This is one of the most confusing aspects of BAC0 Lite. Methods are mixed sync and async:

Method	Type	How to Call
bacnet.write()	Sync	bacnet.write('...') — no await
bacnet.read()	Coroutine	await bacnet.read('...')
bacnet.who_is()	Coroutine	await bacnet.who_is(low_limit=X, high_limit=Y)
bacnet.discover()	Sync	bacnet.discover() — no await
bacnet._discover()	Coroutine	await bacnet._discover(...)

⚠ **NOTE:** *Use inspect.iscoroutinefunction(method) to detect this at runtime. The RuntimeWarning: coroutine '...' was never awaited tells you a coroutine was called without await.*

3.4 BAC0 Write String Syntax

The write string format is strict. Getting any part wrong causes a silent failure:

```
bacnet.write('<IP:PORT> <objectType> <instance> <property> <value> -
<priority>')
```

Field	Example	Notes
IP:PORT	192.168.100.183:52025	Must match current dynamic port from Yabe
objectType	analogValue	camelCase, no spaces
instance	0	Integer instance number
property	presentValue	camelCase
value	31	Numeric or boolean
- priority	- 8	Hyphen space priority. OMIT this for non-commandable properties like outOfService

3.5 bacpypes3 Direct Mode — The Reliable Alternative

When BAC0 discovery fails, use bacpypes3 directly. This bypasses device registration entirely:

```
from bacpypes3.ipv4.app import NormalApplication
from bacpypes3.local.device import DeviceObject
from bacpypes3.pdu import Address
from bacpypes3.primitivedata import Real, Boolean, ObjectIdentifier
from bacpypes3.basetypes import PropertyIdentifier


device = DeviceObject(objectIdentifier=('device', 9999),
objectName='TestBench', vendorIdentifier=999)
app = NormalApplication(device, Address('192.168.100.183:47810'))
await app.write_property(Address('192.168.100.183:52025'),
ObjectIdentifier('analog-value,0'), PropertyIdentifier('present-value'),
Real(31.0), priority=8)
```

⚠ **NOTE:** *The ObjectIdentifier must be constructed as ObjectIdentifier('analog-value,0') — not a tuple. Use the dash-separated type name, not the integer type code.*

4. Debugging Toolkit

4.1 Wireshark — The Ground Truth

When writes dispatch (Running one shot task _write in logs) but values don't change, Wireshark is the definitive tool to determine if packets are physically leaving the Python process.

Filter	Use
udp.port == 52025 || udp.port == 47810	Display filter (in the toolbar after capture starts) — use || not 'or'
Leave blank	Capture filter (on welcome screen) — always leave blank to capture everything first

What You See	Diagnosis
Write packet leaving Python, SimpleACK reply from simulator	Success — both ends communicating
Write packet leaving, NO reply from simulator	Simulator rejecting: wrong port, wrong encoding, or simulator not running
ZERO packets matching filter	Packets not leaving Python: wrong interface, firewall blocking, or binding to wrong NIC
AbortPDU: no-response	Packet left Python but simulator timed out — check encoding or port

4.2 Yabe — The BACnet Inspector

Yabe (Yet Another BACnet Explorer) is the essential GUI tool for inspecting BACnet devices. Key things to check before every test run:

- Current simulator port — hover over the device node in the left panel. It changes on every simulator restart.
- Object instance numbers — expand the Objects list. Never assume AV:0 is the setpoint; verify by checking the Object Name and Description in the Properties panel.
- Present Value and Out Of Service — use the Properties panel on the right to verify writes actually landed.
- Status Flags — 0000 means healthy. Any other value indicates an alarm, fault, or override condition.

4.3 API Introspection — Know Your Library

When documentation is absent or incorrect, use Python introspection to discover the exact behavior at runtime:

```
import inspect
# Check if a method needs await
```

```
print(inspect.iscoroutinefunction(bacnet.read))   # True = needs await

# Check if a call result needs awaiting
result = bacnet.some_method()
if inspect.isawaitable(result): result = await result

# List all available methods
[m for m in dir(bacnet) if not m.startswith('__')]
```

5. Final Working Script (Production Template)

This is the battle-tested script that successfully changes T Set to 31°C on the Room Control Simulator:

```
import asyncio, sys

# MUST be before any bacpypes3 import
if sys.platform == 'win32':
    import asyncio.base_events
    asyncio.base_events._set_reuseport = lambda sock: None
    asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())

from bacpypes3.ipv4.app import NormalApplication
from bacpypes3.local.device import DeviceObject
from bacpypes3.pdu import Address
from bacpypes3.primitivedata import Real, Boolean, ObjectIdentifier
from bacpypes3.basetypes import PropertyIdentifier

TARGET = Address('192.168.100.183:52025')  # verify port in Yabe first!
OBJ    = ObjectIdentifier('analog-value,0')

async def main():
    device = DeviceObject(objectIdentifier=('device', 9999),
                          objectName='TestBench', vendorIdentifier=999)
    app = NormalApplication(device, Address('192.168.100.183:47810'))
    await asyncio.sleep(1)
    # Step 1: Override hardware
    await app.write_property(TARGET, OBJ,
        PropertyIdentifier('out-of-service'), Boolean(True))
    # Step 2: Inject test vector
    await app.write_property(TARGET, OBJ,
        PropertyIdentifier('present-value'), Real(31.0), priority=8)
    # Step 3: Read back
    result = await app.read_property(TARGET, OBJ,
        PropertyIdentifier('present-value'))
    print(f'Verified: {result} degC')
    app.close()

asyncio.run(main())
```

6. Pre-Test Checklist for Real Industry Use

#	Check	How
1	Confirm target IP and port	Check Yabe — port changes on every simulator restart
2	Verify object instance number	Yabe Objects list → check Name and Description
3	Check Out Of Service is False before starting	Properties panel → Out Of Service
4	Check Status Flags = 0000	Properties panel → Status Flags
5	Confirm Python NIC IP	ipconfig — ensure binding IP is reachable to target
6	Windows Firewall UDP rules	New-NetFirewallRule for ports 47808 and 47810
7	Wireshark filter ready	udp.port == <target_port> || udp.port == 47810
8	Restore Out Of Service after test	Write out-of-service = False when test is complete

7. Lessons Learned from This Debugging Session

Symptom	Root Cause	Fix
ValueError: reuse_port not supported	Python 3.13 Windows socket restriction	Monkey patch _set_reuseport before importing bacpypes3
'Running one shot task _write' but no value change	BAC0 queued the task but packets not leaving (wrong binding or discovery not done)	Use bacpypes3 direct mode; bind to same IP as DUT on different port
Trouble with Iam... = []	BAC0 read requires device registration; WhoIs on 47808 never reaches simulator on 52025	Use bacpypes3 app.read_property() with explicit Address — no discovery needed
AbortPDU: no-response	Packet left Python but simulator didn't reply — wrong encoding or port	Verify port in Yabe; use write_property() helper not raw PDU construction
coroutine '...' was never awaited	Called an async BAC0 method without await	Use inspect.iscoroutinefunction() to check; add await
NormalApplication TypeError	Wrong argument type for local_address	Pass Address('IP:port') object, not a plain string
Zero Wireshark packets on loopback adapter	Packets routing through main NIC, not loopback	Capture on main Ethernet adapter; bind Python to main NIC IP
Ping works but BACnet reads timeout	Windows Firewall allows ICMP but blocks UDP	Add explicit UDP firewall rules via PowerShell New-NetFirewallRule