ReadMe.Docx

Author: Brian Smith

2020

Contents

[1. Additional Documentation 2](#_Toc45175348)

# Overview

# BACpypes

# Configuring

1. Install the latest version of Python.
2. BACpypes has a file named setup.py that tells it what version of Python uses a specific source\_folder.

* Type the following from the command line to determine which version of Python is installed

Python –version

* Mine is showing 3.9.13
* I added the following to the source\_folder table:

source\_folder = {

    (2, 5): 'py25',

    (2, 6): 'py25',

    (2, 7): 'py27',

    (3, 4): 'py34',

    (3, 5): 'py34',

    (3, 6): 'py34',

    (3, 7): 'py34',

    (3, 8): 'py34',

    (3, 9): 'py34',

    }.get(version\_info, None)

1. BACpypes has a file named BACpypes.ini. This file must be modified to setup the BACnet server device.

I set up mine like this:

[BACpypes]

objectName: AWS\_BACpypes

address: 10.169.94.127/24

objectIdentifier: 599

maxApduLengthAccepted: 1024

segmentationSupported: segmentedBoth

vendorIdentifier: 10

# Start an AWS Server

1. Open a command prompt bash, powershell, or command prompt
2. Set the Python path

* These instructions only need to be done once per use of a console

**For Bash:**

export PYTHONPATH=D:/dev/bacpypes/py34

* Replace the ‘D:/dev/’ with the location of your BACpypes code
* Type the following to verify the path is set:

env | grep PYTHONPATH

**For Powershell:**

$Env:PYTHONPATH = 'D:\dev\bacpypes\py34'

* Replace the ‘D:/dev/’ with the location of your BACpypes code
* Type the following to verify the path is set:

$Env:PYTHONPATH

**For Command Prompt:**

set PYTHONPATH=D:\dev\bacpypes\py34

* Replace the ‘D:\dev\’ with the location of your BACpypes code
* Type the following to verify the path is set:

echo %PYTHONPATH%

1. Start the AWS\_Server.py by typing the following from a console in the D:\dev\bacpypes directory:

**For Bash and Command Prompt::**

python AWS/AWS\_Server.py

**For Powershell:**

python AWS\AWS\_Server.py

1. Stopping the ASW\_Server.py from the command line

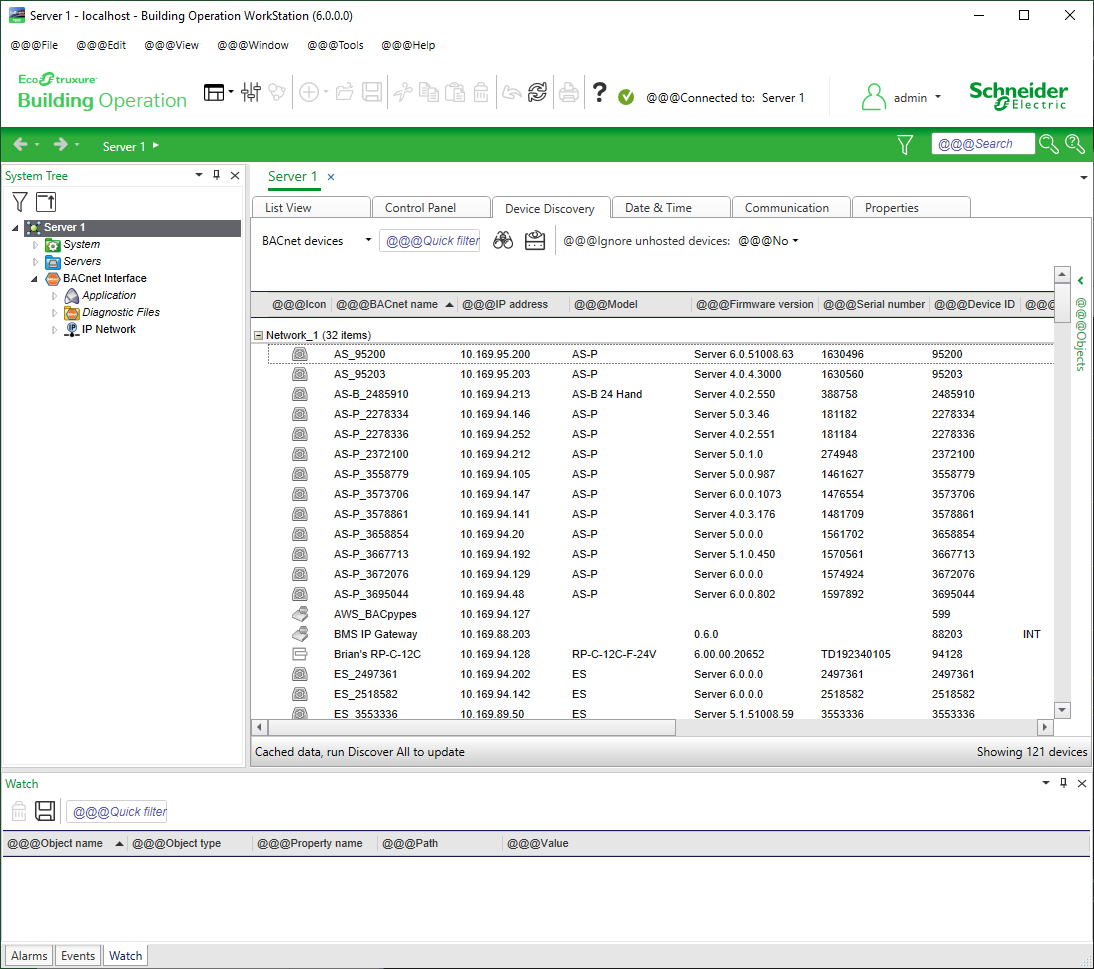
**For BASH and Command Prompt:**

Control-C

**For Powershell:**

Control-Break

1. A discovery in EBO will find the BACpypes server, named AW\_BACpypes, as shown here:



# Client Examples

Once the BACpypes AWS\_Server is started, it will print to the console it’s IP Address, device ID, and objects.

D:\dev\bacpypes>python AWS/AWS\_Server.py

IP Address = 10.169.94.127/24

Device ID = 599

Analog Input <0, 1>

Binary Output <4, 1>

Trend Log Multiple <27, 1>

Trend Log <20, 1>

Date Time Value <44, 1>

Date Time Pattern Value <43, 1>

The server can now be communicated with by using the existing BACpype samples. It is best to utilize two network cards on the PC, 1 for the server and 1 for the client. I have a BACpypes.ini for the server and a BACpypesClient.ini for the client.

The BACpypes samples expect a combination of the servers IP Address, device ID, desired object ID, etc…

The following examples worked for me.

**Read the object list:**

D:\dev\bacpypes>py samples\ReadObjectList.py --ini BACpypesClient.ini 599 10.169.94.127/24

('device', 599): AWS\_BACpypes

('analogInput', 1): Analog Input

('binaryOutput', 1): Binary Output

('trendLogMultiple', 1): Trend Log Multiple

('trendLog', 1): Trend Log

('datetimeValue', 1): Date Time Value

('datetimePatternValue', 1): Date Time Pattern Value

Note: Most of the samples will start a console and require user input. This is not the functionality I desire. I have created an AWSClient.py that will allow the user to read whatever they want and the BACpypes infrastructure will be used to get the data. This can then be simplified by using a GUI to call the scripts with an intuitive graphical view.

# AWS Client

There is a need for a general purpose BACnet client that can be updated easily and allows us to communicate with any BACnet object in a server.

The general needs outweigh the usability, so the functionality should be implemented first, followed by better usability (such as a nice GUI and better help).

Current Needs:

1. Can you send me an email with exactly what you want me to do?

The first ask is that you help me get familiar with the methods.  To that end, I have installed the package already 😊.  If we have time, it would be good to generate some type of client user interface (aka tool), or at least a framework, where a target for the message, a service (ReadProperty, SubscribeCOV, etc), and the service parameters can be specified.  N number of requests could be saved to send again later, if needed.

The purpose of the tool would be to send ad-hoc requests, as needed for troubleshooting or testing a specific test case, where writing a script doesn’t make sense.

1. Can you send me a few examples of commands you want me to send?

A WriteProperty request sent to a device:

Address = ? (presumable some IP address, but the target may be on a remote network – IP or MSTP)

Object ID = (Network Port, 64040)

Property = Command (417)

Value = generate-csr-file (8)

OR

A WriteProperty request sent to a device:

Address = ?

Object ID = (Network Port, 64040)

Property = sc-primary-hub-uri (4194325)

Value = “wss://bacnet.org:1234”

OR

Run script MyGroovyScript.py

* 1. I will get a BACpypes A side and B side working to make sure I can do it
  2. I will then get the A side to communicate to your BTF

1. Do you also need an all purpose command that you can just put whatever you want into?
   1. If so, this can start off minimal and get more and more useful – I agree.  Let’s start small and add along the way.  The ability to send Read/Write requests is the most important thing for now.  I don’t even need to get the answer because I can look to the wire for that.
2. Do you need a GUI front end? A GUI is ideal, with some features as described in the first answer.  I would prefer an application (rather than a web page).  I have some thoughts about how it could be laid out but let’s start simple.
   1. I could start on a webpage or something to give us an interface we can use to call into the python scripts and pass it the data it needs
   2. That would be useful if QA wanted to take over the project in the future
   3. Python is known by most engineers and lots of testers
   4. I can document things thoroughly and have step by step exampless

# Troubleshooting

1. The AWS\_Server is not showing up in the EBO Workstation discovery
   1. Try disabling the firewall
2. TBD