Bacpypes Simulator

Author: Brian Smith

2020

# Overview

The BACpypes simulator comes with all of the needed BACpypes files, which have been modified.

This should be updated to also utilize git patches so that a full version of BACpypes can be downloaded and have the patch applied to it.

It is good to have the full BACpypes for simplicity and so we can recreate the exact simulator executable.

It is good to have the patch files created so that a good diff can be performed to see what we modified and also to upgrade BACpypes as needed.

# Pre-Requisites

1. Install the latest version of Python
   1. I have 3.11.8 as of this writing, which will work with the current Simulator
   2. It must be added to the system PATH
   3. Mine is in C:\Users\sesa565821\AppData\Local\Programs\Python\Python311
2. Install pyinstaller using pip
   1. I have pyinstaller 6.4
   2. python -m pip install pyinstaller
   3. pyinstaller creates standalone executables from Python scripts. These are not cross-platform but our Fit tests are run on Windows.
   4. It must be added to the system PATH
   5. Mine is in C:\Users\sesa565821\AppData\Local\Programs\Python\Python311\Scripts
3. Install the python ConfigParser
   1. python -m pip install ConfigParser
4. The BACpypesSimulator comes with a modified version of BACpypes
   1. If the latest BACpypes is needed then it can be obtained from GitHub at <https://github.com/JoelBender/bacpypes>
   2. It can be obtained from the command line with
   3. git clone git@github.com:JoelBender/bacpypes.git clean\_bacpypes
      1. I call it clean\_bacpypes since we should use patches to update it locally
5. Install 7-Zip
   1. This should be available from the Schneider Electric Software Updater
   2. It must be added to the system PATH
      1. Mine is here : "C:\Program Files\7-Zip\7zFM.exe"

# BACpypes

# Configuring

This information is useful if a full version of BACpypes is downloaded from git, since it will need to be updated to use the latest version of Python.

1. 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 3.11.8
* 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',

(3, 11): 'py34',

    }.get(version\_info, None)

# Logging

1. Import the needed module for logging

from bacpypes.debugging import ModuleLogger

1. Import the ability to turn on the logging via the console

from bacpypes.consolelogging import ConfigArgumentParser

1. Enable the debugging

\_debug = 1

\_log = ModuleLogger(globals())

1. Add the template code that lists the names of the debugging log handlers

args = ConfigArgumentParser(description=\_\_doc\_\_).parse\_args()

1. Launch the script from the command line with the debugging option

python BACpypesSim.py --debug \_\_main\_\_

# Starting Simulator

TODO: Update this section with the latest testing to help make it easier to understand and more accurate.

**From the script**

Type:

export PYTHONPATH=D:/dev/bacnet\_tools/bacpypes\_simulator/bacpypes (for bash)

or

set PYTHONPATH=D:/dev/ bacnet\_tools/bacpypes\_simulator/bacpypes (for Windows command prompt)

then type

python BACpypesSim.py --ini=BACpypes.ini --localAddress=10.169.94.127:47809 --bbmdAddress=10.169.94.127:47808 --objects=object\_configurations\basic\_objects\_cfg.py

**From the executable script**

Type:

build.bat

go to the newly created simulator directory and type:

./BACpypesSim.exe --ini=../../BACpypesSim.ini --localAddress=10.169.94.125:47809 --bbmdAddress=10.169.94.125:47808 --objects=../../object\_configurations\basic\_objects\_cfg.py

**Similar to Fit test**

export PYTHONPATH=D:/dev/bacnet\_tools/bacpypes\_simulator/bacpypes

python BACpypesSim.py --ini=BACpypes.ini --localAddress=10.169.94.127:47809 --bbmdAddress=10.169.94.127:47808 --objects=object\_configurations\empty\_config.py

**Loading all Objects into the BACpypes Server:**

export PYTHONPATH=D:/dev/bacnet\_tools/bacpypes\_simulator/bacpypes

python BACpypesSim.py --ini=BACpypes.ini --localAddress=10.169.94.127:47809 --bbmdAddress=10.169.94.127:47808 --objects=object\_configurations\basic\_objects\_cfg.py

**Working with EBO:**

1. Start EBO server and Workstation
2. Create a BACnet Interface with an IP Network
3. Create a BBMD
   1. This is mandatory if the BACpypes is on the same IP address but uses a different port
4. Start the BACpypes simulator as shown in one of the steps above
5. A discovery will find the BACpypes server as long as it was started with the --bbmdAddress entry.
6. Right click on the BACpypes server and do an upload all. This will obtain all of the objects.

**Working with Fit Test**

1. Go to the EBO directory
2. Open wireshark on the ‘Adapter for loopback traffic capture’
   1. Follow the instruction in the Wireshark section to setup Wireshark correctly
3. Type the following to run the DownloadUpload test

./nspbuild.exe acceptancetest nspclients --suite=single:BACnetDone/BACnetDownloadUploadTest.html

1. When the test finishes, go to the ebo\clientcomponents\Fit\test\_results\SuiteNsp\SuiteDone\BACnetDone directory
2. Open the BACnetDownloadUploadTest.html file to see the result of the test
3. Save the Wireshark to see the packets going to/from the BACpypesSim
4. Open the Windows Task Manager and delete the BACpypesSim.exe (it doesn’t always get removed)

# pyInstaller

1. I placed the bacpypes folder in the same location as the BACpypesSimulator so that pyInstaller can easily find it and create the executable without issues.
2. I ran

pyinstaller BACpypesSim.spec

* I originally ran ‘pyinstaller BACpypesSim.py’ and then modified the newly created BACpypesSim.spec so that it has “bacpypes” in the hiddenimports section.

1. I then ran build.bat, which will zip the BACpypesSim.exe file and the ‘\_internal’ subfolder that is in ‘dist->BACpypesSim’, into a file called simulator.zip
2. I then unzipped the simulator.zip file in the C:\dev\ebo\artifacts\tools\bacpypessimulator directory
   1. This is so that the fit tests will use the new simulator

# BBMD (BACnet Broadcast Management Device)

The BBMD foreign device handling is shown in the BACnet standard in Annex J.5.

EBO will interact with the BACpypes server with the use of a BBMD. The BACpypes server resides on the same IP Address as the EBO server. Because of this, they must both use different ports. The BBMD will allow the EBO workstation to communicate to the BACpypesSim with a who-is.

The BACpypesSim interacts with the EBO BBMD using a foreign device. A foreign device is a BACnet device that resides on a different subnet than the BACnet network the device seeks to join.

BBMD operation:

* The foreign device must register itself with a BBMD. It does this by sending a BVLL Register-Foreign-Device message to the BBMD and receives a BVLC-Result message of X’0000’ .
* The BBMD has an FDT (Foreign device table). When a foreign device registers with a BBMD, it is added the the BBMD’s FDT.
* Has NetworkPort information for the BACnet\_IP\_Mode that signifies the device is operating as a foreign device.

This is configured the following way:

1. The EBO server IP Network has an IP address:port of 10.0.0.77:51917
2. The BACpypes server has an IP address:port of 10.0.0.77:51918
3. The BACpypesSim.py will create a foreign device application that will work with EBO

# Artifactory

From Patrik Hartlen:

The EBO packages are described in:

buildsystem/configuration/packages.xml (ebo repo)

you can see this line

<binary name="BACpypesSimulator" destination="artifacts/tools/BACpypesSimulator" version="1.3.3-se1" category="tools" platforms="windows" symlink="true"/>

So you need to deploy a new package to Artifactory (if its the same version use 1.3.3-se2 other vice 1.x.y-se1

<https://artifactory.eur.gad.schneider-electric.com/ui/repos/tree/General/Artifacts-Release/tools/BACpypesSimulator>

In order to get the BACpypes zip file, created by the build.bat file, to Artifactory:

From Patrik Hartlen:

zip the files to BACpypesSimulator-2.0.0-se1.zip

and then after logging in to Artifactory upload the zip file to:

<https://artifactory.eur.gad.schneider-electric.com/ui/repos/tree/General/Artifacts-Release/tools/BACpypesSimulator>

Note the version path:

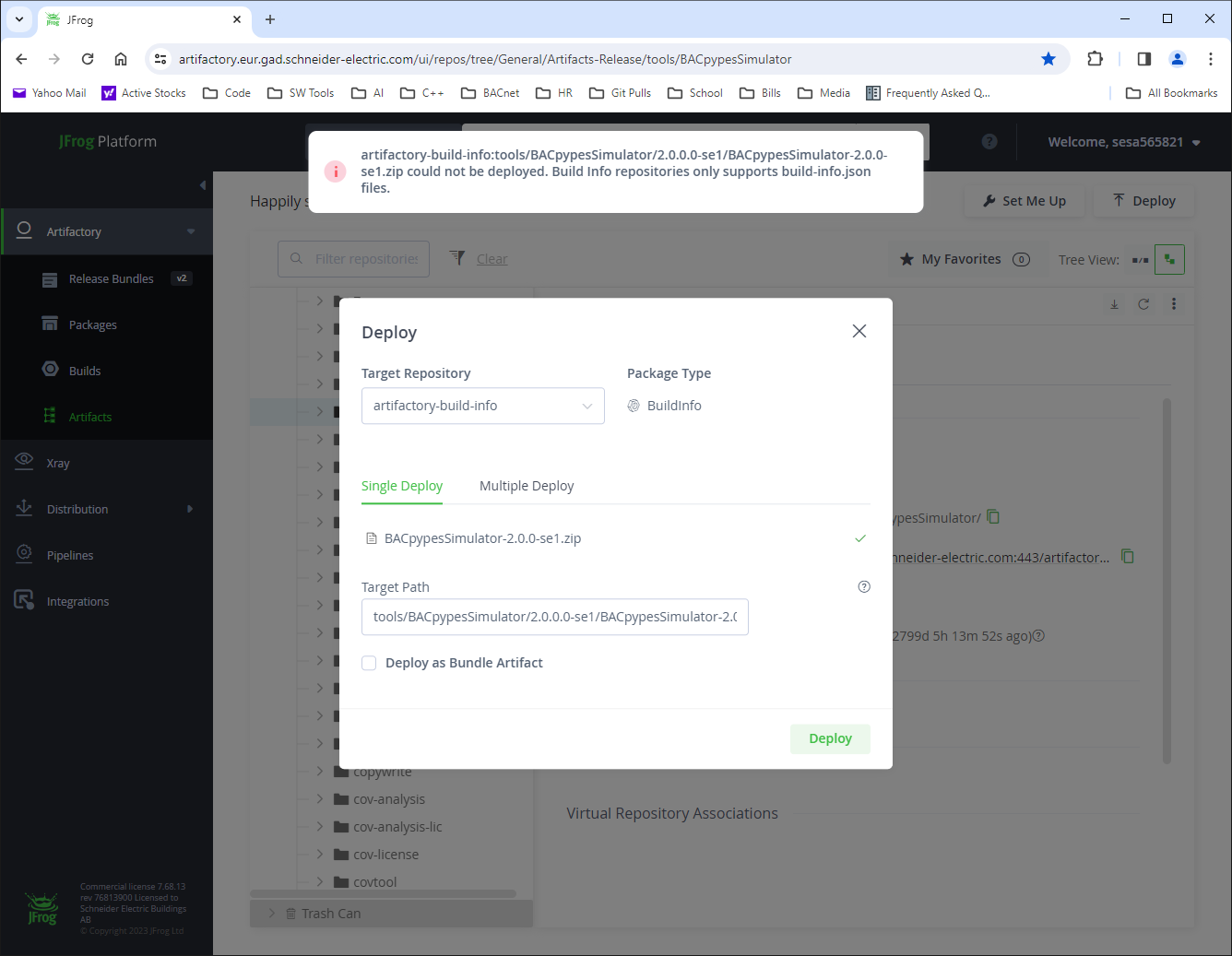
Artifacts-Release/tools/BACpypesSimulator/**2.0.0-se1**/BACpypesSimulator-2.0.0-se1.zip

Then change the version in packages.xml to match the new uploaded version:

❯ grep "BACpypesSimulator" buildsystem/configuration/packages.xml

    <binary name="BACpypesSimulator" destination="artifacts/tools/BACpypesSimulator" version="1.3.3-se1" category="tools" platforms="windows" symlink="true"/>

I initially ran into an issue and got this error:



The solution was to upload to the ‘Artifacts-Release’ repository.

Note: If you manually modify the artifacts/tools/BACpypesSimulator directory, then you will have to delete the whole directory and run ‘./nspbuild.exe config nspclients’ in order to get it back.

* The artifacts/tools/BACpypesSimulator directory is a link to the artifactory download cache on your local PC.
* My cache is located at C:\dev\nspbuild\_cache\
* I had to delete the following directory:
* C:\dev\nspbuild\_cache\BACpypesSimulator-1.3.3-se1

# Debugging VSCode

In order to debug the python scripts using VSCode, I have done the following:

1. I created a .env file and placed it in the BACpypes\_Simulator folder
   1. This file contains the following line

PYTHONPATH=D:/dev/bacpypes/py34

* 1. Make sure to replace the D:/dev with the location your bacpypes code is in

1. I created a launch.json file in the VSCode debugger tab
   1. This is the launch.json file I use:

{

    "version": "0.2.0",

    "configurations": [

        {

            "name": "BACpypesSim: Current File",

            "type": "debugpy",

            "request": "launch",

            "program": "${file}",

            "args":[

                "--ini","BACpypesServer.ini",

                "--deviceIdentifier", "4194302",

                "--localAddress", "10.0.0.77:51918",

                "--bbmdAddress", "10.0.0.77:51917",

                "--objects", "object\_configurations/basic\_objects\_cfg\_small.py"

            ],

            "console": "integratedTerminal",

            "justMyCode": true

        }

    ]

}

1. Modify the launch.json file to use the desired device, ip address, commands, etc…
2. When debugging, make sure the file to be debugged is the active file in the editor

# Debugging Workstation

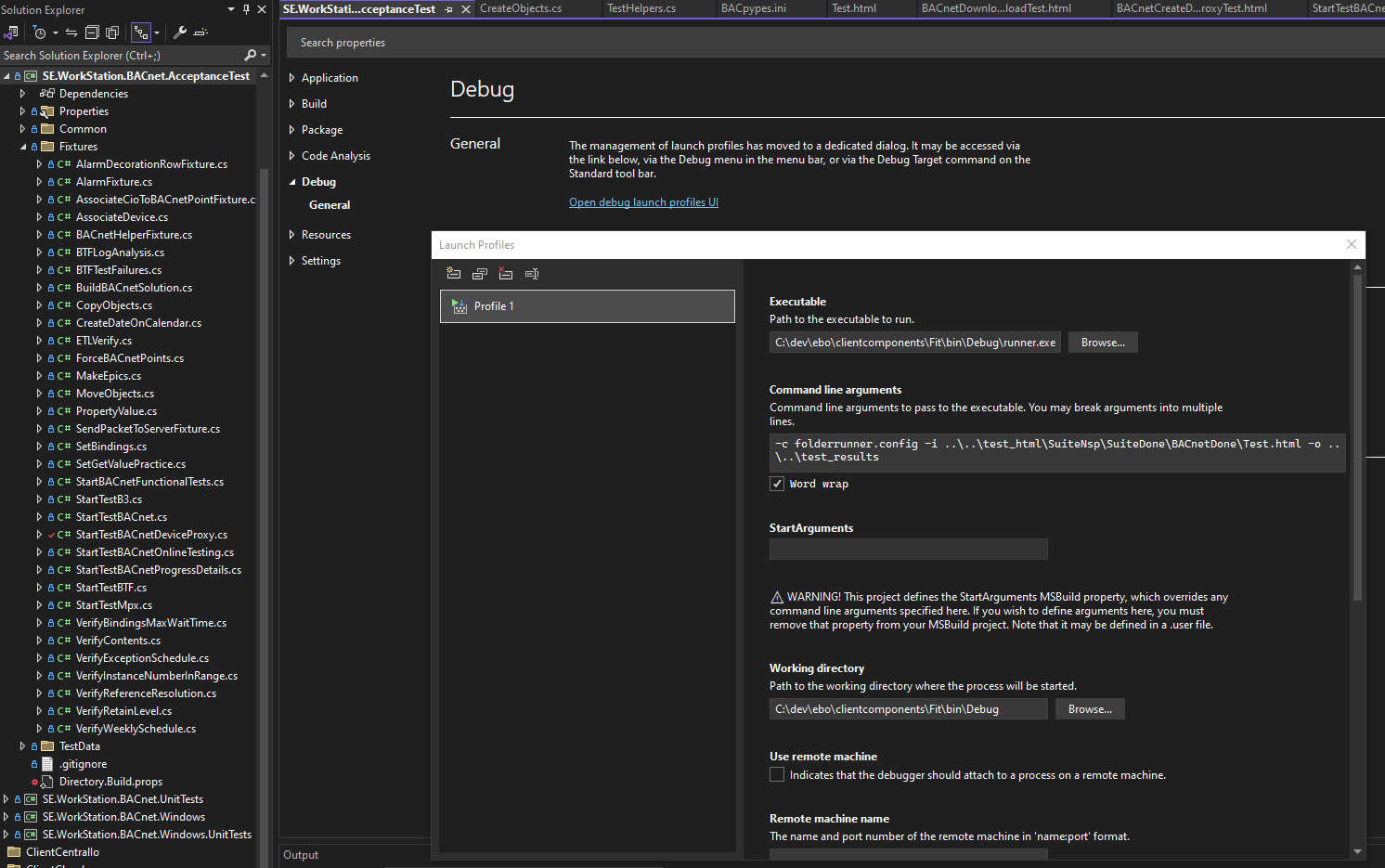
**With Fit Test Breakpoint**

* Put a break in the acceptance test. This must be put anywhere after the startTest table

<table><tr><td>Debug message</td><td>Your message here</td></tr></table>

* Build and run the Workstation from Visual studio
* Run the acceptance test
  + When it stops waiting for you to acknowledge the debug message, then from Visual studio, attach to the “runner” process.
  + Acknowledge the debug message and the test will continue to run

**With Visual Studio Debugger**



For the executable: select C:\dev\ebo\clientcomponents\Fit\bin\Debug\Runner.exe

For the Working directory: C:\dev\ebo\clientcomponents\Fit\bin\Debug

Add the following "Command line arguments" for the html you are testing.

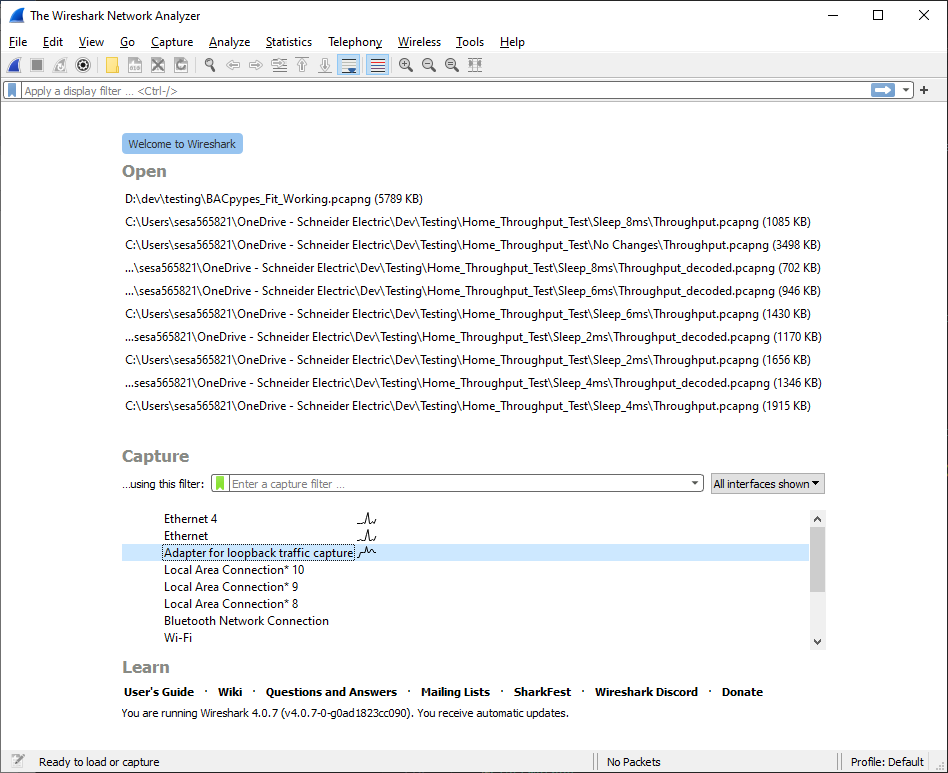
-c folderrunner.config -i ..\..\test\_html\SuiteNsp\SuiteDone\BACnetDone\BACnetCreateDeleteProxyTest.html -o ..\..\test\_results

# Wireshark

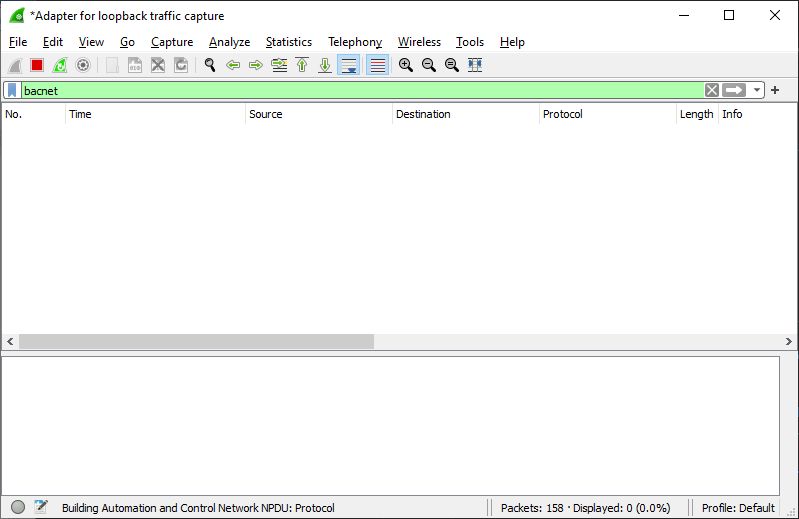
The Fit test will create random ports for the EBO server and the BACpypesSim server. Wireshark normally associates port 47808 with a BACnet BVLC.

In order to view and decode the messages, the following should be done in Wireshark:

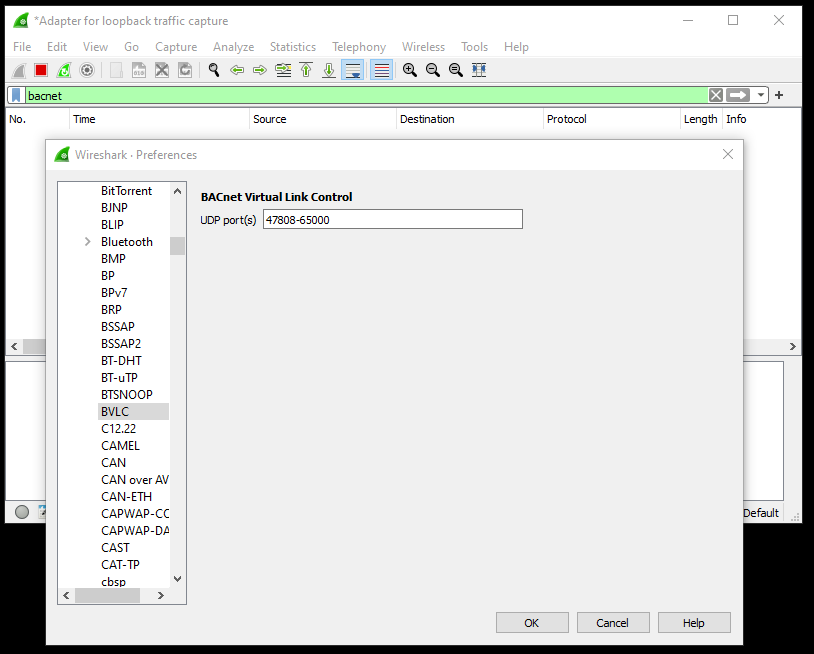
1. Capture on the ‘Adapter for loopback traffic capture’



1. Set the filter to ‘bacnet’



1. From ‘Edit’->’Preferences’, update the UDP like this (it will allow Wireshark to decode the entire range used by the Fit Test randomization as BACnet):



# Filtering Examples

This is how you would search for all ‘resolution’ properties:

bacnet && (bacapp.property\_identifier == 106)

* The resolution ID is 106. It is listed in the specification in the following section:

**BACnetPropertyIdentifier** ::=

# Fit Test

------

Pre-requisite (must build wines and nspclients fully (do not use the --notests flag):

./nspbuild.exe config,clean,build wines,nspclients

------

One of the fit tests used is:

clientcomponents\Fit\test\_html\SuiteNsp\SuiteDone\BACnetDone\BACnetCreateDeleteProxyTest.html

------

It is started with:

./nspbuild.exe acceptancetest nspclients --suite=single:BACnetDone/BACnetCreateDeleteProxyTest.html --server=localhost --httpport=80

* The ‘--server=localhost --httpport=80’ can be removed if the wines server needs to be spawned automatically
* A breakpoint can be put in the BACnet server code to debug while a fit test is running

Other fit tests are:

./nspbuild.exe acceptancetest nspclients --suite=single:BACnetDone

* This will run all the BACnet tests

./nspbuild.exe acceptancetest nspclients --suite=single:BACnetDone/BACnetDownloadUploadTest.html

* This test relies on the following:

C:\dev\ebo\clientcomponents\ClientBACnet\SE.WorkStation.BACnet.AcceptanceTest\TestData\BACnet\Imports\download\_test\_objects.xml

./nspbuild.exe acceptancetest nspclients --suite=single:BACnetDone/BACnetWorkstationTest.html

------

The result of the test is located here:

clientcomponents\Fit\test\_results\SuiteNsp\SuiteDone\BACnetDone

------

The EBO Workstation starts the simulator with the following C# code:

clientcomponents\ClientBACnet\SE.WorkStation.BACnet.AcceptanceTest\Fixtures\StartTestBACnetDeviceProxy.cs

* Uses the ConnectSimulatorToNetworkWithConfiguration() function

string nspbuildRoot = Environment.GetEnvironmentVariable("NSPBUILD\_ROOT");

string simulatorExe = nspbuildRoot + @"\artifacts\tools\BACpypesSimulator\BACpypesSim.exe";

string simulatorArgs = string.Format("--ini={0} --localAddress=0.0.0.0:{1} --bbmdAddress={2}:{3} -- objects={4}", simulatorIni, simulatorPort, bbmdAddr, bbmdPort, configPath);

Note: An example of the above is

localAddress: 0.0.0.0 => is 10.0.0.77 on my laptop

simulatorPort = 62630

bbmdAddr = 10.0.0.77

bbmdPort = 62629

------

The BACpypes simulator that the Fit test connects to is in:

artifacts\tools\bacpypessimulator\BACpypesSim.exe

* This executable can be temporarily renamed as a zip file to look at it’s contents
* It contains all of the needed scripts to run the python simulator as an executable
* If changes are done to the BACpypesSime.exe, it can be updated here
* The update of this file will be done automatically during config as shown in the Artifactory section

------

The BACnet objects to load and the BACpypes ini files are here:

clientcomponents\ClientBACnet\SE.WorkStation.BACnet.AcceptanceTest\TestData\BACnet

* The basic\_objects\_cfg.py and basic\_objects\_cfg\_2.py contain the objects to load for testing
* BACpypes.ini contains the DeviceName, DeviceID, and some enable/disables options

Wireshark decoding:

* See the Wireshark section

# BACpypes Modifications

1. BACpypes.ini
   1. This must have the desired IP Address and port, Network number, Device ID, Device Name, Maximum APDU Accepted, and Segmentation Supported
2. SimulatorDevice
   1. This is the actual BACnet device that will hold all of the desired BACnet objects
   2. This inherits from LocalDeviceObject
3. SimulatorApplication
   1. This must be started to utilize the instantiated the SimulatorDevice
4. Object Configuration
   1. Contains config files to setup the initial value of an object’s properties.

# Updating an Object to use an Existing Property

This example uses the tags property, since I had to add it to all the objects.

1. The tags are defined in the BACnet spec as BACnetARRAY[N] of BACnetNameValue
2. The BACpypes has an existing type defined for BACnetNameValue called NameValue, located in basetypes.py
3. I had to modify the NotificationClassObject so that it now has an OptionalProperty of tags that is an Arrayof(NameValue)
4. I had to modify the property\_defaults.py file to have a default
   1. This step is not needed if the default is already there
   2. I added

'tags':                             { 'default': ArrayOf(NameValue)() },

# Multi State & Exception Schedule

1. I had to modify ReadProperty of the Property class in object.py
   1. It was not decoding the ‘stateText’ correctly
   2. I had to put a workaround in until I can figure out how to get the ArrayOf(CharacterString) to correctly get the length to the first element
2. Wireshark shows the following for the older code:

[0] returns the size

[1] returns the first state text value

* 1. The new code is treating [0] as the first index. That is not correct. It must return the size.

# Exceptions

The previous code was written in an earlier version of Python. The Try, Except must now use ‘as’ and not a comma.

# property\_defaults

I had to modify this file so that it contains all the properties that were added in the newer specifications. This file will set the defaults to None if they are unknown, which causes the read and writes to fail.

# basetypes

I added new base types that are used with BACnetSC.

# WeekNDay

I fixed an issue where the conversion was not correct. It is in the CE function in object)utils.py

This mostly appeared with Upload of a Calendar object. It could be seen in Wireshark since it was decoded wrong.

# ForceOverrideTest

1. I fixed a bug where the createObject function in application.py was re-using the default property values instead of creating a new one for each object
2. You could originally see this error by simply creating 2 AnalogValue objects and changing the PresentValue of 1. Then look at the Command tab and the value is changed on both.

# Resolution Property

1. The AnalogInput, AnalogOutput, IntegerValue, Positive Integer Value, and AnalogValue all had the resolution as a Parameter
   1. I changed it to a Variable since it is read only in the spec
   2. Keeping it as a Parameter causes it to always be removed on create and download, since EBO sees that it is an optional parameter and there is no setter. Adding a setter would fail as well since a well behaved device will have it as read only.

Ex… of the old situation

* + - An Analog Input object in a device has a resolution of 0.5
    - EBO goes into database mode and deletes the object
    - EBO leaves database mode and create a new AnalogInput object with the same ID
    - EBO downloads the object
    - It will remove the value from the device permanently with no way to put it back
  + I tested with database upgrade and import/export.

# ReadRange

1. The ReadRangeRequest is defined in bacpypes\apdu.py and is registered with register\_confirmed\_request\_type(ReadRangeRequest)
   1. This is similar to the ReadProperty
2. I added the doReadRangeRequest in bacpypes\service\object.py
   1. This is similar to do\_ReadPropertyRequest

# Backup

The BACpypes simulator now has basic Backup and Restore capabilities.

1. Add the ReinitializeDevice service
   1. There is an existing service called ReinitializeDeviceRequest and it is registered
   2. It inherits from ConfirmedRequestSequence, just like ReadPropertyRequest
   3. I added this in the backuprestore.py file
   4. I register this in application.py, just like the other overridden services
2. Add some server side logic for a backup
   1. Thread
   2. State machine
   3. Sample configuration file creation and restoration
3. I instantiated a BackupRestore class in the simulator application so that service requests can obtain the state information of the backup
   1. class SimulatorApplication(LocalDeviceObject):

# Object Configurations

Here is an example configuration that can be used as a template when updating others. There is some tricky formatting that is needed to make the assignment acceptable to the internal code. Ex… The eventTimeStamps is an ArrayOf(TimeStamp,3) and needs to be created as shown below.

#Table of objects - formated as a python list

[

    (AnalogInputObject,  [(1, 'BACnet Analog Input (Generic)',

        dict(

            presentValue = 0.0,

            statusFlags = [0,0,0,0],

            eventState = 'normal',

            outOfService = False,

            units = 'noUnits',

            reliabilityEvaluationInhibit = False,

            updateInterval = 1000,

            covIncrement = 1.0,

            resolution = 0.1,

            minPresValue = -100.0,

            maxPresValue = 100.0,

            interfaceValue = OptionalReal(real = 1.0),

            highLimit = 80.0,

            lowLimit = -80.0,

            faultHighLimit = 90.0,

            faultLowLimit = -90.0,

            timeDelay = 1,

            notificationClass = 1,

            deadband = 0.2,

            limitEnable = [1,1],

            eventEnable = [1,1,1],

            ackedTransitions = [1,1,1],

            notifyType = 'alarm',

            eventMessageTexts = ArrayOf(CharacterString)

            (

                [

                    'OffNormal',

                    'Fault',

                    'Normal'

                ]

            ),

            eventMessageTextsConfig = ArrayOf(CharacterString)(['','','']),

            timeDelayNormal = 1,

            eventDetectionEnable = True,

            eventAlgorithmInhibit = False,

            eventAlgorithmInhibitRef = ObjectPropertyReference(

                objectIdentifier = ("binaryOutput", 1),

                propertyIdentifier = PropertyIdentifier('activeText'),

                propertyArrayIndex = 0

            ),

            eventTimeStamps = ArrayOf(TimeStamp)([TimeStamp(dateTime=DateTime(date=Date(), time=Time())) for i in range(3)]),

        ))]

    ),

]

# Testing

1. Create Object

The BACpypes simulator is capable of handling the CreateObject service, documented in the spec in section 15.3.

* CreateObject with an ObjectType of analog-input with an object-name will attempt to create the object
  + It will fail if an object with this name already exists
  + An error will be sent if the name is duplicated of ‘First failed element number (0)’
  + From the spec:

**First Failed Element Number**

This parameter, of type Unsigned, shall convey the numerical position, starting at 1, of the offending 'Initial Value' in the 'List of Initial Values' parameter received in the request. If the request is considered invalid for reasons other than the 'List of Initial Values' parameter, the 'First Failed Element Number' shall be equal to zero.

* CreateObject with an ObjectType of analog-input without an object-name section will attempt to create the object
  + The BACpypes code will reach CreateObject in the createdeleteobject.py script
  + It will the call make\_object\_id in order to make a unique name

# Python Useful Functions

1. issubclass
   1. This is used to check if a class is derived from another class
   2. In the decoding of BACnet, this is used a lot

            datatype = obj.get\_datatype(apdu.propertyIdentifier)

            if \_debug: ReadWritePropertyServices.\_debug("    - datatype: %r", datatype)

            # get the value

            value = obj.ReadProperty(apdu.propertyIdentifier, apdu.propertyArrayIndex)

            if \_debug: ReadWritePropertyServices.\_debug("    - value: %r", value)

            if value is None:

                raise PropertyError(apdu.propertyIdentifier)

            # change atomic values into something encodeable

            if issubclass(datatype, Atomic) or (issubclass(datatype, (Array, List)) and isinstance(value, list)):

                value = datatype(value)

            elif issubclass(datatype, Array) and (apdu.propertyArrayIndex is not None):

                if apdu.propertyArrayIndex == 0:

                    if ( (apdu.propertyIdentifier == 'stateText') or

                         (apdu.propertyIdentifier == 'shedLevelDescriptions')):

                        value = CharacterString(value)

                    elif (apdu.propertyIdentifier == 'exceptionSchedule'):

                        value = datatype.subtype(value)

Ex…

obj = bacpypes.object.ScheduleObject

apdu.propertyIdentifier = ‘exceptionSchedule’

datatype = bacpypes.constructeddata.ArrayOf(SpecialEvent)

datatype.subtype = bacpypes.basetypes.SpecialEvent

issubclass(datatype, Array) = true

// The ArrayOf() class has a member called subtype, which defaults to None

# Troubleshooting

1. The Artifact directory will not update
   1. ./nspbuild.exe config nspclients
   2. If you have edited files that are symlinked to the artifactory download cache, then running config will not help since the cached files are now containing your edits
   3. If this occurs, then delete the complete folder to start over.
2. There are already instances of bacpypesSim.exe still running
   1. Open the Windows Task Manager and delete all occurrences of bacpypesSim.exe.
3. Upload is failing for ArrayOf properties
   1. Update the fix I described in section 11 for the Multi State

# TODO

1. There is a new repository for BACpypes3
   1. If we have a lot to work on, we may want to move to this. It has newer python, IPv6, some SC
   2. <https://bacpypes3.readthedocs.io/en/latest/>

git clone https://github.com/JoelBender/BACpypes3.git