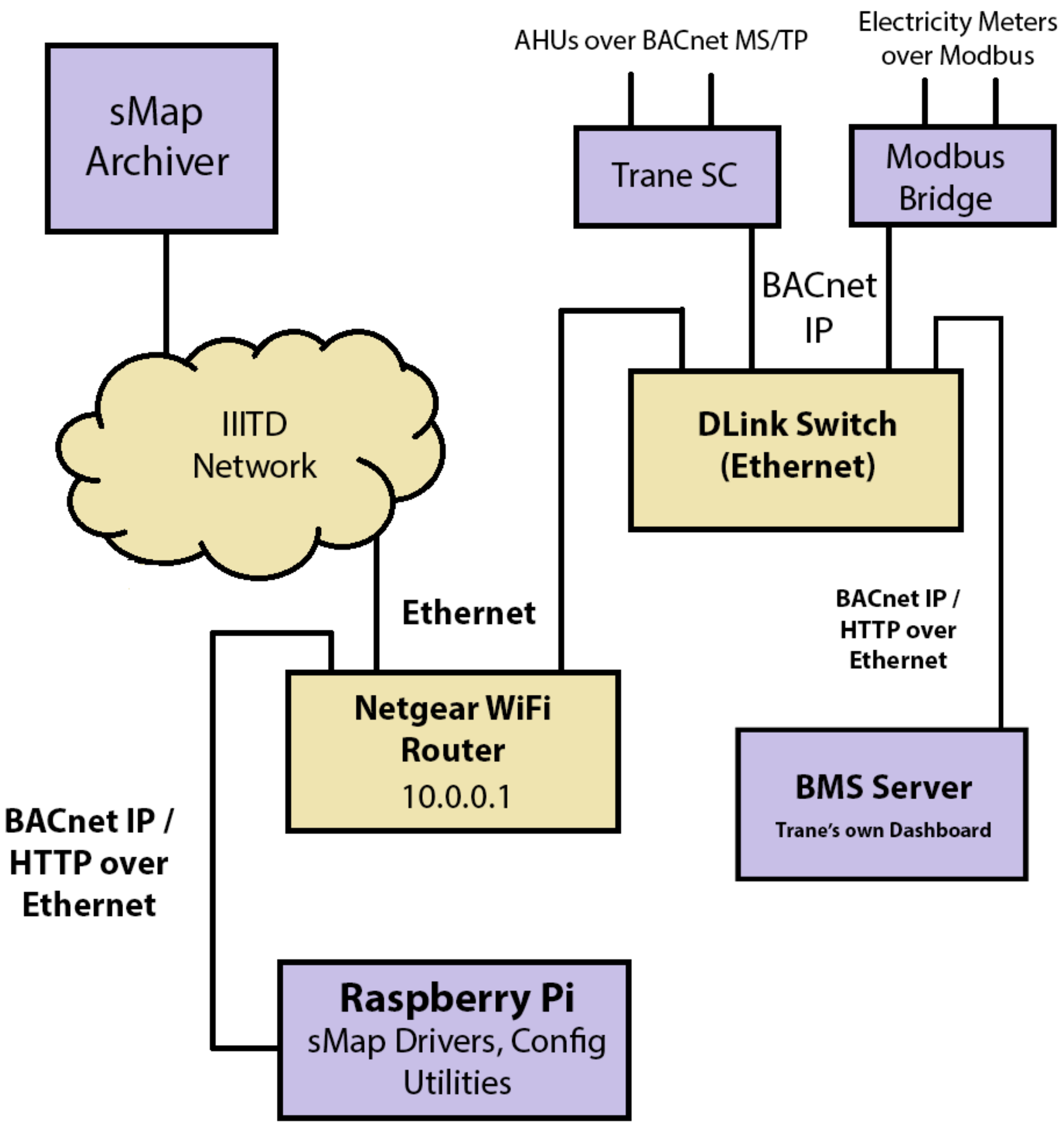
Smap BACnet Driver Documentation

Romil Bhardwaj
2011092

Table of Contents

High Level Architecture



Device	Current IP
Raspberry Pi	10.0.0.5
Trane SC	10.0.0.11
JENEsys Modbus Bridge	10.0.0.12
BMS Server	10.0.0.10

The Drivers

The Smap BACnet Package contains two drivers, one for generic BACnet devices (iiitd_bms.py) and one for BACnet meters specifically (iiitd_bms_meters.py). Their core structure is the same, except for the metadata tags and stream paths. The BACnet meter driver is kept in accordance with the standard tags we have followed for the other meters in the campus. This was not possible for other generic BACnet points (AHUs etc.) since the previously defined metadata schema was specific to electricity meters.

BACnet & BACPypes

BACnet is a communications protocol for building automation and control networks. In order to read any BACnet point on a BACnet IP network, we require four parameters:

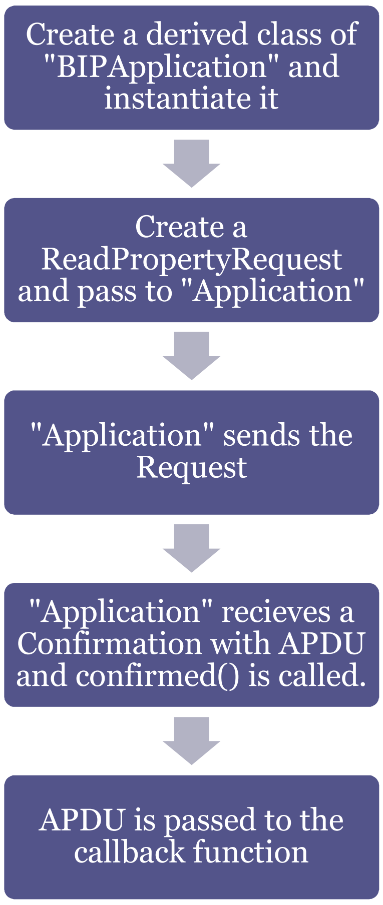
- BACnet Device IP Address
- BACnet Point Instance ID
  - Two devices with different object type can have the same instance id.
- BACnet Point Object Type
  - An enumeration of BACnet object types, analogValue, analogInput (read only), analogOutput (read and write), binaryValue etc.
- BACnet Property Type
  - Usually "presentValue" is used. objectName can be used to read the object's configured name.

These four uniquely identify any point on a BACnet network. Note that these points have to be configured in a BACnet IP device, such as the Trane SC.

Not many libraries for BACnet exist. BACnet4J is the library used in Mango M2M, and is considered to be very well written. BACpypes is an extensive python library for BACnet communication but it has a very fruitless documentation. Much about it has to be learnt by examining the source code and its tiny community at http://sourceforge.net/mailarchive/forum.php?forum_name=bacpypes-developers.

I will try to cover the basics of BACpypes: how to read a BACnet point value. In order to read a property in BACpypes, we first create an "application" (derived from the BIPApplication class) which will act as a BACnet device on the network and then place that request. A flowchart is given below to explain the process. APDU stands for Application Protocol Data Unit, the basic unit of data in BACnet.

Although this is not used for reading properties, but when a request is received by BACpypes (eg. IAm request), it is passed to the indication() function where it is accordingly processed.


Create a derived class of "BIPApplication" and instantiate it
Create a ReadPropertyRequest and pass to "Application"
"Application" sends the Request
"Application" recieves a Confirmation with APDU and confirmed() is called.
APDU is passed to the callback function

BACpypes ReadProperty Program Flow

Config Files

Each driver has a corresponding conf file (iiitd_bms.conf and iiitd_bms_meters.conf), which contains information about the driver to be used, the points to be read, the metadata tags and the delivery location key (archiver key). The generic driver conf file has the following fields:

Fields	**Remarks**
inst_id	BACnet Instance ID
ip	BACnet Device IP
obj_type	BACnet Object Type
prop_type	BACnet Property Type (Generally presentValue)
value_type	Physical Parameter (eg. Power)
value_unit	Physical Units (eg. Watts)

rate	Polling Rate to read data
point_parent	Point’s Logical Parent (For Path, eg. 1F_AHU_A)
point_name	Point’s Name (eg. RelativeHumidity)
point_floor	Floor
point_building	Building
point_source	BACnet Data Source (eg. SC1, ModbusBridge)

The Electricity meter driver omits the point_parent and point_name fields (They are directly derived from MeterID and value_type) and adds the following fields: point_type, point_loadtype, point_subloadtype, point_supplytype and meterid. These are configured according to the metadata tags already in use for other meters in the facilities building.

For each installation, be sure to check the bacnetdeviceip in the config file. It has to be the IP address of the Raspberry Pi. Keep the ports different for both the confs. (Default is 47808)

Smap Specific Information

The Smap driver runs in two phases: Setup and Start. On initialization, the setup function is called, which reads the config file (opts variable) and creates the points in its memory. Then a queue of lists is created, each list indicating the list of points to be read at that second every minute. Then for each point in the config file, it is checked if its timeseries path already exists, if it does not, it is created. The appropriate metadata is associated with that stream in JSON format and then the initialization is complete.

The Start function is called when the setup function terminates. The periodicSequentialCall() function is used to repeatedly call the read() function every second. The read function reads the value from BACnet, wraps it in the correct stream and sends it to the archiver. Then it pops it from its current list in the time queue and appends it to the list corresponding to its next read time.

The Path of a stream for the Generic BMS driver is:

BACnet :: /BMS/point_source/point_building/point_parent/point_name

Eg: /BMS/SC1/Academic Building/1F_AHU_B/RAT

The Path of a stream for the Electricity Meter BMS driver is:

BACnet :: /BMS/point_source/point_building/Meter+MeterID/value_type

Eg /BMS/SC1/Facilities Building/Meter7/Energy

Config Utilities

Each driver has an associated web-based config utility written in django (bmspanel for generic driver and bmsmeterpanel for electricity meter driver). They have the following features:

- View, Add, Edit and Delete Bacnet points from the drivers' respective conf
- Pre-scanned BACnet WhoIs Scan Results for reference
- Debugging Utilities – Directly read BACnet points from the WebUI
- Authentication system (default login: admin, password: bms@iiitd)

Be sure to run BOTH the web services on different ports. On first install, if using a non standard (not /home/pi) folder on a Raspberry Pi, edit the DB Name key in /bmspanel/settings.py and the conf filepath in /webconfig/views.py for both the utilities. To run the server, goto the root directory of each utility use the command (change x to 0 or 1, just keep it different for both the utilities):

Python manage.py runserver 0.0.0.0:800x –noreload –nothreading

If running for the first time, run this before anything else:

Python manage.py syncdb

SmapBacnetUtils.py

Both utilities come with a SmapBacnetUtils.py helper class in the /webconfig folder. It contains a template class for BACnetpoints and the configparser functions for editing the config files. For each installation, be sure to check the bacnetip in SmapBacnetUtils.py in /webconfig folder.

Change Admin Password

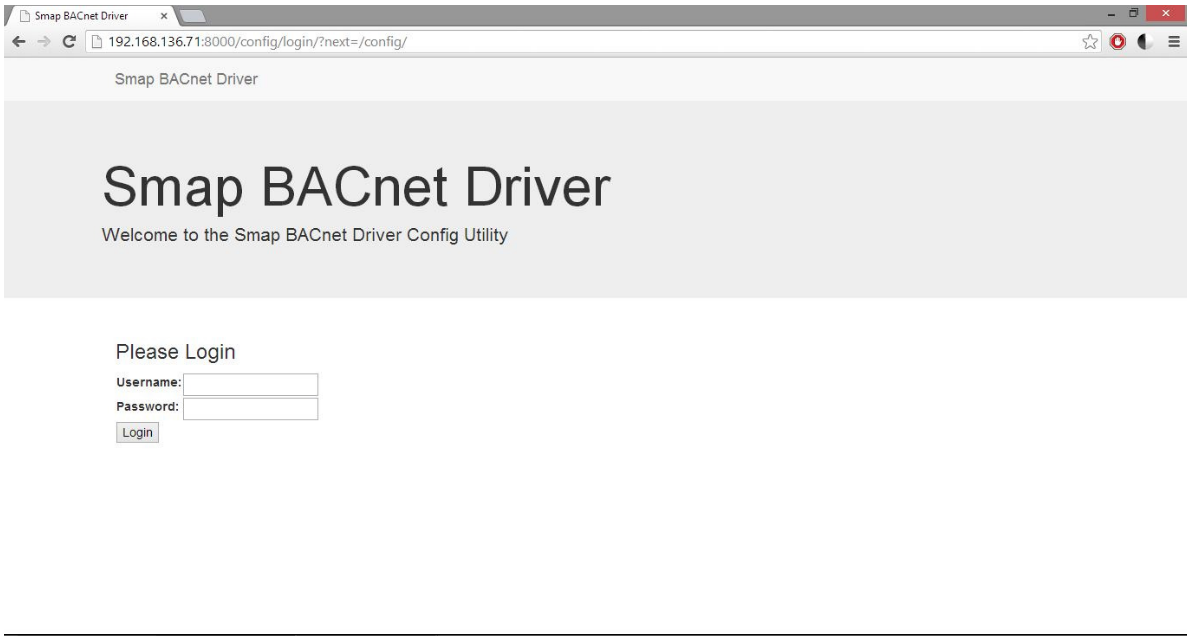
To change the admin password, simply goto http://127.0.0.1:8000/admin/password_change.

If a hard reset needs to be done, delete the "bms" file in the web utility root folder and then run "python manage.py syncdb".

Updating the BACnet Scan Results

The web utilities have pre-scanned BACnet WhoIs scan results stored for reference. To update the scan results with a new page, simply replace the static_pointslist.htm file in \webconfig\templates\webconfig directory in the utility root folder.

Screenshots



Smap BACnet Driver

Welcome to the Smap BACnet Driver Config Utility

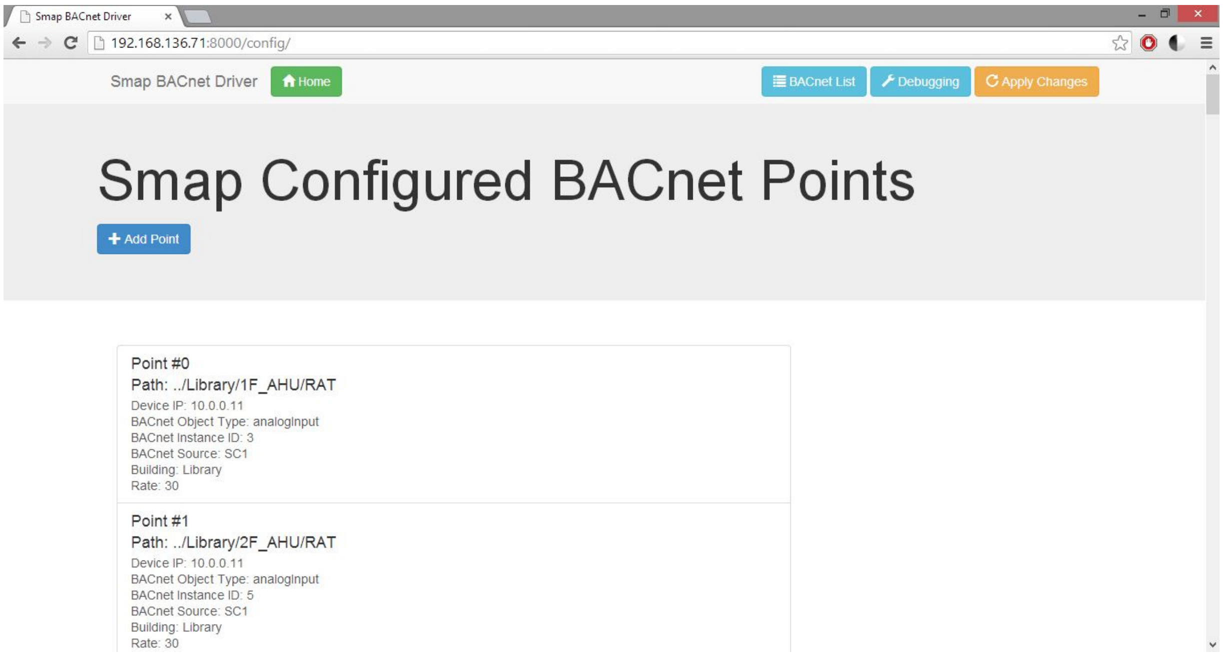
Please Login

Username:

Password:

Login

Login Page

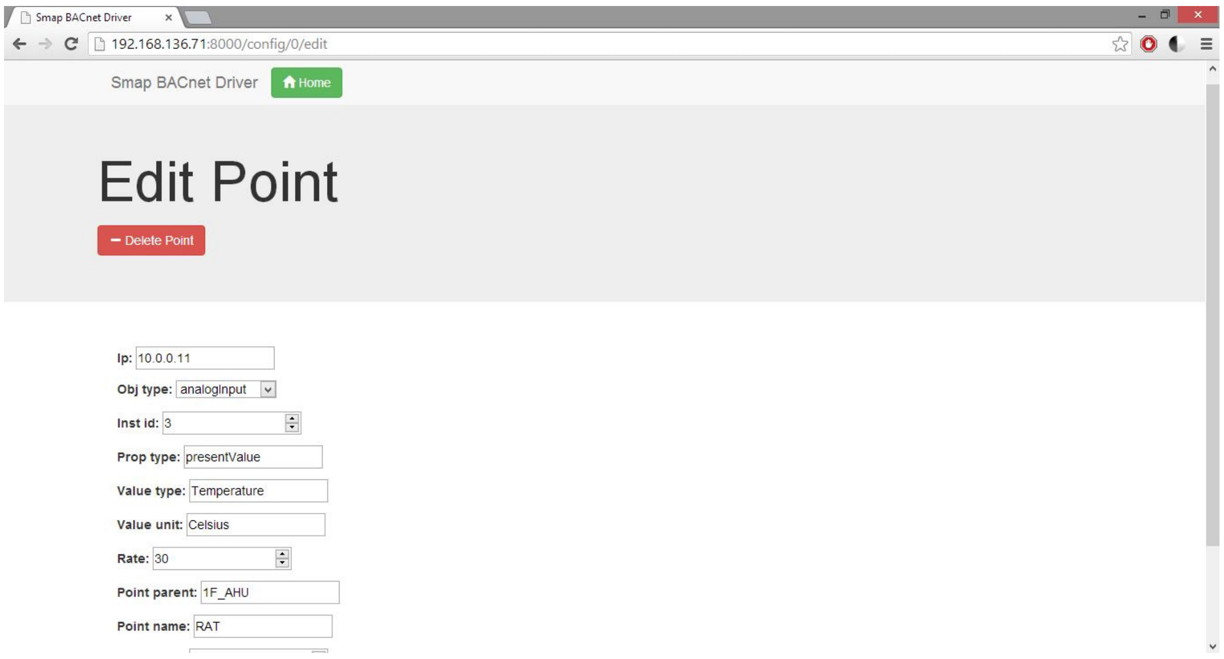


Smap Configured BACnet Points

Point #0
Path: ../Library/1F_AHU/RAT
Device IP: 10.0.0.11
BACnet Object Type: analogInput
BACnet Instance ID: 3
BACnet Source: SC1
Building: Library
Rate: 30

Point #1
Path: ../Library/2F_AHU/RAT
Device IP: 10.0.0.11
BACnet Object Type: analogInput
BACnet Instance ID: 5
BACnet Source: SC1
Building: Library
Rate: 30

Home


Smap BACnet Driver

192.168.136.71:8000/config/0/edit

Smap BACnet Driver Home

Edit Point

Delete Point

Ip: 10.0.0.11

Obj type: analogInput

Inst id: 3

Prop type: presentValue

Value type: Temperature

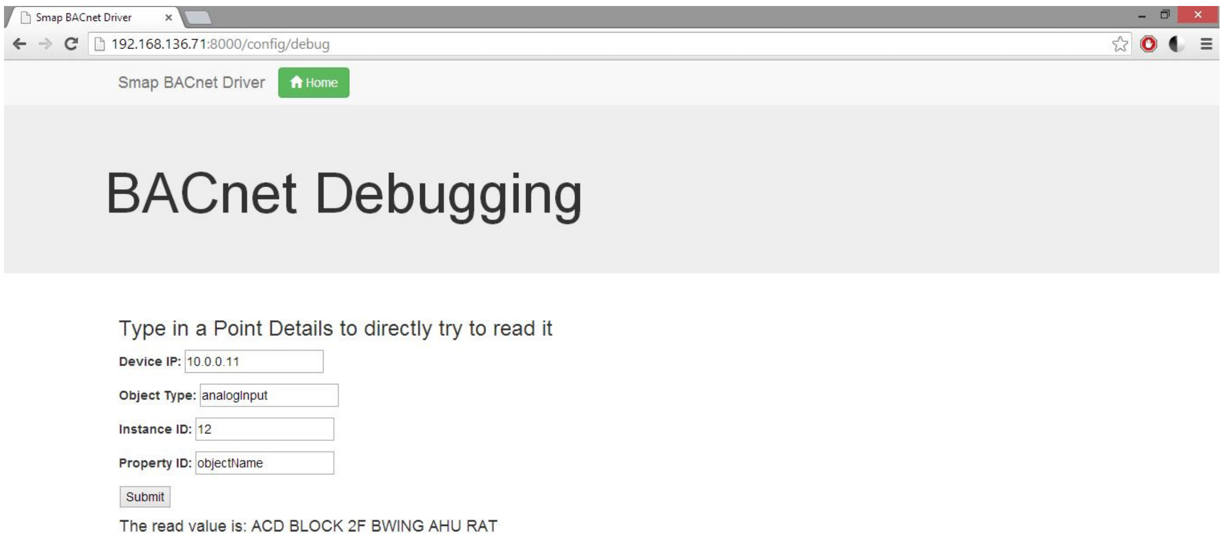
Value unit: Celsius

Rate: 30

Point parent: 1F_AHU

Point name: RAT

Edit Point Page


Smap BACnet Driver

192.168.136.71:8000/config/debug

Smap BACnet Driver Home

BACnet Debugging

Type in a Point Details to directly try to read it

Device IP: 10.0.0.11

Object Type: analogInput

Instance ID: 12

Property ID: objectName

Submit

The read value is: ACD BLOCK 2F BWING AHU RAT

Debugging Page

Installation

1) Setup a Raspberry Pi with the latest distribution of Raspbian.
2) Setup the network as shown in the High Level Architecture.
3) Put bmspanel, bmsmeterpanel, smap folders and installbmsdriver.sh, runsmap.sh files in /home/pi/
4) Run chmod +x installbmsdriver.sh; ./installbmsdriver.sh

   It essentially installs the required packages and dependencies and adds runsmap.sh to rc.local to run at boot. You may be asked to enter the login and passwords for the web utility.

5) Configure the IP addresses in /smap/iiitd_bms.conf, /smap/iiitd_bms_meters.conf, /bmsmeterpanel/webconfig/SmapBacnetUtils.py and /bmspanel/webconfig/SmapBacnetUtils.py.

   If need be, also edit the paths in /bmsmeterpanel/webconfig/views.py, /bmspanel/webconfig/views.py, /bmsmeterpanel/bmspanel/settings.py and /bmspanel/ bmspanel /settings.py.

6) Reboot and use the config utilities at http://127.0.0.1:8000 and http://127.0.0.1:8001 to configure the drivers.
7) Reboot again.

Current Deployment Notes

As on 19th December 2013, the web utilities can be accessed at:

Generic Driver: http://192.168.136.71:8000/

Meter Driver: http://192.168.136.71:8001/

Both have the same login credentials (admin/bms@iiitd)

The raspberry pi can be ssh'd into at pi@192.168.136.71:1234. Password is "raspberry".

The netgear router can be accessed at http://192.168.136.71:8080. The login credentials are (admin/bms@iiitd).

Adding new Metadata

To add new metadata, add a new key/value pair in the iiitd_bms.conf and make sure to fill the new values for the existing points. Some changes need to be done in the following driver and web utility files:

- ./bmspanel/webconfig/forms.py
- ./bmspanel/webconfig/models.py
- ./bmspanel/webconfig/views.py
- ./bmspanel/webconfig/SmapBacnetUtils.py
- ./smap/iiitd_bms.py

In all these files, the location where new code is to be added is marked with a comment "Add any new Metadata here". Simply add the code there as required, following the same code pattern as there already would be for other metadata tags. Finally restart the system and changes should be done. The procedure is same for electricity meter driver.

Problems Faced

- BACnet packets cannot be transmitted across different subnets without a special router known as BBMD (BACnet/IP Broadcast Management Device). The BMS BACnet at IIITD had been setup on an isolated network in the Facilities Building using only a simple switch and manually configured IPs. This was required to be on the IIIT Delhi network to collect data on the archiver, however directly plugging in the SC to the IIIT Network made debugging difficult since the SC was configured on the sensor VLAN, while other debugging machines maybe on different subnets and will not receive any BACnet packets.

  Thus, the local LAN setup was left undisturbed and a router was installed which linked the switch and its devices to the IIIT Delhi network. However, since this created a new subnet, a Raspberry Pi was installed to locally collect data and send it directly to the archiver on the IIITD network.

- Unlike the electricity meters which are read over Modbus, there is no clear hierarchy defined in the BMS. The BACnet devices just have a collection of data points, with no notion of hierarchy in them (Eg: Instead of an AHU having children points RAT, RH etc., BMS has just AHU 1 RAT, AHU 1 RH points). The hierarchy in smap was created using the supplied metadata tags, point_source, point_building, point_parent and point_name. For the electricity meter driver, it was created using point_source, point_building, meterid and value_type.

- The update frequency for the points in the IIIT Delhi BMS seems to be on a COV (Change of Value) basis as of now. BACpypes does not support COV subscriptions, thus the polling is done at regular intervals to collect data.

- The driver I initially wrote was generic in nature, and adding meter specific metadata to its config file required significant restructuring. Thus to accommodate meter specific metadata, a new driver was created as an extension of the generic driver.

- ConfigParser library in Python, while writing the config file for the WebUI, changes the keys (in key = value) to lowercase by default, which is not read by smap. Specifying the ConfigParser.optionxform=str option fixes this.

- BACpypes is multithreaded and extensively uses signals in python. However, signals can only be made by the main thread, whereas Django spawns multiple threads for its web server. Thus the debugging in the web UI cannot work unless Django is run in a single threaded mode. This is done using –noreload and –nothreading parameters while running the development server.

- Even though WhoIs scans may reveal multiple BACnet sub networks in the BACnet IP network, they are inaccessible via BACpypes. This is because NPDU (Network Layer Protocol Data Unit) packets have not been implemented in BACpypes yet.

- The WhoIs requests on the IIITD BMS network in BACpypes are not successfully completed. As seen using a packet sniffer such as WireShark, WhoIs request packets are sent and even IAm request packets are broadcasted on the network, but they are not collected by BACpypes. A possible explanation given on the BACpypes mailing list is that BACpypes only collects unicast IAm requests. The Trane hardware sends only broadcast IAm requests, thus no devices show up on doing a WhoIs from BACpypes on the IIITD BMS network. This may work on other hardware from other vendors.

- The driver is insensitive to blank spaces in the metadata in the config file. It is mandatory that each point has some value associated with every metadata tag, even if it is not defined just enter “None”. Direct editing of the config file using a text editor should be avoided. It is recommended to use the bundled web config utility.

- BACpypes has an almost non-existent documentation. The learning curve is steep with few examples and limited community support.