ME Journal

Sean Keene (Template by Tylor Slay)

last updated: April 9, 2021

Contents

List of Errors

List of Warnings

Fall 2020

March 9, 2021 – Initial Script

OBJECTIVE: Start a python script that connects me to GridAPPS. Start playing around with queries.

OUTLINE:

Today's tasks were:

- Get a python file up and running
- Connect to a model via python api
- Get the model mrid
- Do some sort of live query

PROCEDURE:

Getting the script set up was routine. PyCharm is a lot more efficient than Spyder. I used a combination of the hackathon and instructions to generate the required code.

PARAMETERS:

N/A

OBSERVATIONS:

Everything worked as expected. The visualization id was easy to get from a query. The blazegraph part was harder. See data.

I had a hard time with the Pull Request. Make a new branch first locally, make changes, commit, push to remote, then make the pull request. Update main locally, and delete branches.

Some other issues and errors with the latex compiler. Turned off syncTex (which seems unnecessary to begin with, it just allows you to go directly to a pdf line from the code??) Needed another run command too, see options.

DATA:

- Code is on GitHub.
- sim id: '357545598'
- 13 node feeder MRID: '_49AD8E07-3BF9-A4E2-CB8F-C3722F837B62'

RESULTS:

Done.

March 11, 2021 – Initial Script

OBJECTIVE: Get GridAPPS-D running from a script.

OUTLINE:

Today's tasks were:

- Figure out how to set the configuration of a model from the python script
- Connect to and run the script

PROCEDURE:

First, I needed the command to get all the feeders in the database:

```
# list all the feeders, with substations and regions - DistFeeder
    PREFIX r:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX c:  <http://iec.ch/TC57/CIM100#>
    SELECT ?feeder ?fid ?station ?sid ?subregion ?sgrid ?region ?rgnid WHERE {
     ?s r:type c:Feeder.
     ?s c:IdentifiedObject.name ?feeder.
     ?s c:IdentifiedObject.mRID ?fid.
     ?s c:Feeder.NormalEnergizingSubstation ?sub.
     ?sub c:IdentifiedObject.name ?station.
     ?sub c:IdentifiedObject.mRID ?sid.
     ?sub c:Substation.Region ?sgr.
     ?sgr c:IdentifiedObject.name ?subregion.
     ?sgr c:IdentifiedObject.mRID ?sgrid.
     ?sgr c:SubGeographicalRegion.Region ?rgn.
     ?rgn c:IdentifiedObject.name ?region.
     ?rgn c:IdentifiedObject.mRID ?rgnid.
}
ORDER by ?station ?feeder
```

Then I wrote the code in the LogTest.py file to set up the simulation:

```
run_config_13 = {
    "power_system_config": {
        "GeographicalRegion_name": "_73C512BD-7249-4F50-50DA-D93849B89C43",
        "SubGeographicalRegion_name": "_ABEB635F-729D-24BF-B8A4-E2EF268D8B9E",
        "Line_name": "_49AD8E07-3BF9-A4E2-CB8F-C3722F837B62"
    },
    "application_config": {
        "applications": []
    },
    "simulation_config": {
        "start_time": "1570041113",
        "duration": "120",
        "simulator" : "GridLAB-D",
        "timestep_frequency": "1000",
        "timestep_increment": "1000",
        "run_realtime": True,
        "simulation_name": "ieee123",
        "power_flow_solver_method": "NR",
        "model_creation_config":{
            "load_scaling_factor": "1",
            "schedule_name": "ieeezipload",
            "z_fraction": "0",
```

```
                "i_fraction": "1",
                "p_fraction": "0",
                "randomize_zipload_fractions": False,
                "use_houses": False
            }

        },
```

Then I wrote the code to start the simulation.

PARAMETERS:

See above

OBSERVATIONS:

Need `{\usepackage[outputdir=auxil]{minted}}` for minted in this document, as well as to change the auxil folder to the one in the journal folder.

RESULTS:

Done. Now able to run a 13-node simulation from the script.

Mar 18, 2021 – Initial Log API Functionality

OBJECTIVE:

Get some logging done

OUTLINE:

1. Meet with Shiva
2. Write a script that subscribes to the simulation output topic
3. Put this data somewhere

PROCEDURE:

Today's task was to take what I've learned so far and begin to apply it in unique ways. The first thing I did was write (or more accurately adapt) a callback function. This is required whenever you subscribe to a topic. In the case of the simulation output, the python thing calls this function once every three seconds. So the ideal thing to do was, after a lot of testing, use this callback to write a line of data to a dataframe. In this early step I skipped the middleman and just wrote to a csv. This will allow me to easily test and troubleshoot the eventual data going into the pandas dataframe.

I went of script and tried subscribing to the simulation log script. This is useful because it gives more data in the python terminal about the simulation and, importantly, the status of the simulation. This can be used to stop the script from running infinitely.

RESULTS:

Logs are being generated containing who-knows-what, I just used a random message for the topic. I think it's pretty much the entire simulation output. Next step is to filter it down to more specific data and put that in a pandas dataframe with useful labels and, importantly, timecodes. Then, sadly, I need to rewrite the entire script since it's basically a scratchpad file at the moment.

April 9, 2021 – Log API test: tightening up

OBJECTIVE:

Further updates on the log test, including the lookup table for headers, adding timecodes, and fixing some stuff. I forgot to update this for a couple weeks so this will contain quite a few different small updates.
OUTLINE:

1. Find a way to update the .csv header with real names rather than MRIDs
2. Find a way to add the simulation timecode as a column
3. Convert the simulation timecode to an actual datetime

PROCEDURE:

Header

The output stream provides a dictionary of dictionaries, each sub-dictionary containing values associated with a measurement id (MRID). These MRIDs are a CIM thing that are associated not just with measurements but with models, components, etc. I needed a way to take these MRIDs and turn them into names so the logs are reasonable. In order to do this, I needed to use one of the queries to get a list of MRIDs with names, and write a script in the callback function to automate converting MRIDs to names for the header (but, importantly, not the csvwriter object's header! Otherwise the writer stops working.) This is done.

Add timecodes

This is a little more dry. I am taking timecodes from the logging topic, putting them in an array each time the callback function is called, and adding that to the .csv output. I needed a snippet to get the column on the left. This works well enough, except at the end I realized this contains the CURRENT date and time, not the SIMULATION date and time. So I need to rewrite this to get the right thing.

Readable timecodes

This was a pain and took a long while until I realized pandas does it natively. I convert the GAD timecodes (which are in a unix-like format but with extra digits for ms) to dates and times using pandas. I should really just use pandas for everything on the rewrite.

RESULTS:

Mostly, except the timecodes are wrong. I just need to modify the script so that `SIMULATION_TIME` reads from the output stream rather than the logging stream, I think.