**UVM Energy Hackathon**

Thursday, August 22, 2024

9:00 PM – 3:00 PM EDT

Burlington, VT

**AGENDA**

| TIME | TOPIC | Facilitator |
| --- | --- | --- |
| **Day 1 08/22/2024** | **Thursday** |  |
| **08:30 – 09:00 pm** | Session Overview | All |
| **09:00 – 11:30 pm** | Translating Electric Power Distribution System Models (Leveraging GridAPPS-D) | Monish Mukherjee, PNNL |
| **12:00 – 12:30 pm** | LUNCH BREAK | -- |
| **12:30 – 2:30 pm** | Developing Distribution Grid Response Models (Leveraging TESP) | Jessica Kerby, PNNL |
| **2:30 – 4:30 pm** | Enabling Co-Simulation with Distribution Grid Response Models (Leveraging HELICS) | Trevor Hardy, PNNL |
| **Day 1 08/23/2024** | **Friday** |  |
| **2:30 – 4:30 pm** | Overview of Large-scale studies and PNNL capabilities. | T. Hardy, M. Mukherjee, J. Kerby PNNL |

# Installation Instructions

## GRIDAPPS-D CIMHUB Installation

### OpenDSS and OPENDSSCMD

1. Download the regular version of OpenDSS from SOURCEFORGE:
   * Download Link: [*https://sourceforge.net/projects/electricdss/*](https://sourceforge.net/projects/electricdss/)
   * Install the binary (*OpenDSSInstaller1000\_2.exe*)
   * This version works well for Windows users and provides GUI support.
2. Download the command-line version of OpenDSS from SOURCEFORGE:
   * Download: [*https://sourceforge.net/projects/electricdss/files/OpenDSSCmd/*](https://sourceforge.net/projects/electricdss/files/OpenDSSCmd/)
   * Install the OS specific binary (Windows: *opendsscmd-1.7.7-windows-installer.exe*). The opendsscmd binary distribution is available for all OS.
   * Opendsscmd also supports some key features like CIM export and co-simulation interface (with HELICS and FNCS).

### CIMHUB

1. Java SDK
   * Java SDK 11 or Higher:
     + Java SDK binaries are available for different OS. Here is link for downloading Java SDK 11: <https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>
   * Apache Maven 3.8.1:
     + The binary can be downloaded from this link: <https://maven.apache.org/docs/3.8.1/release-notes.html>
     + Extract the binary (for example: *apache-maven-3.9.9-bin.zip*) in a chosen folder (for example: *C:\cimhub\_req\)* and add two environmental variables (**ME\_HOME**: *C:\cimhub\_req\apache-maven-3.8.1\bin* and **MAVEN\_HOME**: *C:\cimhub\_req\apache-maven-3.8.1*) for maven to be discoverable.
2. Java IDE (optional)
   * Download a Java IDE to facilitate auto configuration of the required dependencies. If you do not have a preferred IDE, try IntelliJ IDEA (<https://www.jetbrains.com/idea/download/?section=windows>). It has a community version is free to use and is compatible across different OS.
3. CIMHUB
   * Download the maple\_leaf

## TESP (-ish) + HELICS Installation

1. (optional but highly recommended) - Create a virtual environment (conda, venv, etc) and enter it.
   1. (MacOS with conda) Make an x86 virtual environment: CONDA\_SUBDIR=osx-64 conda create -n <<environment name>> -c conda-forge python=3.12
2. (macOS only): Launch an x86 Terminal: arch -x86\_64 zsh
3. Install generic dependencies: pip install matplotlib pandas
4. Install TESP API – Inside the virtual environment, define an installation directory of your choice
   1. Clone in the TESP API repo: git clone <https://github.com/pnnl/tesp.git>
   2. Change directory to TESP repo
   3. Switch to develop branch: git checkout develop
   4. Navigate to ..../tesp/src/tesp\_support
   5. Install TESP: pip install –e .
5. Install HELICS and Python language binding: pip install helics
6. Install GridLAB-D with HELICS support
   1. Windows
      1. [Download installer](https://github.com/gridlab-d/gridlab-d/releases/download/v5.2/GridLAB-D-5.2.0-Windows.exe) from [Github repository](https://github.com/gridlab-d/gridlab-d)
      2. Find the helics.dll library installed with HELICS; pip show helics should show you the path where HELICS was installed by pip
      3. Find the location where the GridLAB-D binary was installed; where gridlabd should work.
      4. Copy the helics.dll from where pip installed it and copy it to the same folder as the GridLAB-D binary, renaming it to libhelics.dll
   2. MacOS
      1. Find path to the HELICS dynamic library: pip show helics will help get you pointed in the right direction; you’re looking for “libhelics.dylib” which should be in the “helics” folder.You’ll be using this to set the GLD\_HELICS\_DIR and HELICS\_DIR build variables when building GridLAB-D.
      2. Find path to where binaries are installed in your environment: which helics\_broker is that path. This will be the CMAKE\_INSTALL\_PREFIX build variable when building GridLAB-D.
      3. Clone in the GridLAB-D repository:
         1. Move back to the root of your installation direction (where you cloned in tesp\_support).
         2. git clone <https://github.com/gridlab-d/gridlab-d.git>
         3. cd gridlab-d
         4. git checkout develop
         5. git submodule update --init
      4. Install build dependencies
         1. Install homebrew: /bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)”
         2. brew install cmake
         3. brew install ccmake
      5. Configure build
         1. mkdir cmake-build
         2. cd cmake-build
         3. cmake ..
         4. ccmake .
            1. t – turn on all the variables
            2. CMAKE\_INSTALL\_PREFIX = <<path to environment install location, not ending with “bin”>>
            3. GLD\_HELICS\_DIR = <<path to where the HELICS library has been installed>>
            4. GLD\_USE\_HELICS = ON
            5. c – configure
            6. An error will be produced but it will populate the CCMAKE interface with the missing environment variable, HELICS\_DIR.
            7. HELICS\_DIR = <<path to where the HELICS library has been installed>>/cmake/HELICS
            8. c – configure
            9. g - generate
      6. Build GridLAB-D
         1. make –j<<number of cores>>
         2. make install
         3. Check install with gridlabd --version
7. Switch to the root of the install direction of the Clone in the HELICS-Example repository with the test case: git clone <https://github.com/GMLC-TDC/HELICS-Examples.git>
8. Checkout the “uvm” branch: git checkout uvm
9. Move to the directory with the example: ..../unmaintained/python/uvm
10. Test your installation
    1. TESP
       1. Go to your installation root, enter the “tesp” directory and set up your environment variables by running
          1. Windows: run tesp.bat
          2. macOS and Linux: source tesp.env
       2. Put the “South\_D1\_Alburgh\_mod\_tesp\_v2” distributed with this into the tesp/capabilities/uvm folder.
       3. Open feeder\_config.json5 and define
          1. "in\_file\_glm": "South\_D1\_Alburgh\_mod\_tesp\_v2.glm"
          2. "out\_file\_glm": "South\_D1\_Alburgh\_mod\_tesp\_v2\_populated.glm"
       4. Run the script: python gld\_residential\_feeder.py
       5. Confirm the file South\_D1\_Alburgh\_mod\_tesp\_v2\_populated.glm is created.
    2. HELICS
       1. From the root of your installation directory, go to HELICS-Examples/unmaintained/python/uvm.
       2. Run the 5 EV example:   
          helics run --path=ev\_charge\_runner\_5ev.json. The console should show the example running to completion without error and two graphs should display

