MDSPlus with the FLAP package at W7-X

S. Zoletnik

24.03.2021

How to install

To use MDS+ on the virtual machine (VM) it should be first installed. Instructions can be found here: http://mds-data-1.ipp-hgw.mpg.de/

MDSPlus is implemented in flap via two modules. The general flap_mdsplus and the flap_w7x_mdsplus. Therefore for FLAP at least three modules should be installed:

fusion_flap, flap_mdsplus, flap_w7x_mdsplus

Open explorer on the VM and select a directory where all the Python packages will be located. Right click on the directory and select “git bash”. In the window type:

```
git clone https://github.com/fusion-flap/flap.git flap_fusion
git clone https://github.com/fusion-flap/flap_mdsplus.git flap_mdsplus
git clone https://github.com/ fusion-flap/flap_w7x_mdsplus.git flap_w7x_mdsplus
```

(The local directory name of the flap package should not be set to ‘flap’ as it would be mixed up with the package name inside the directory.)

Flap is normally used with the Anaconda package manager. To install Anaconda download the install the package from anaconda.com and run it.

To create the FLAP environment start Anaconda navigator from the start menu, select “Environments” and “Import”. Type in the name of the new environment (e.g. flap) and select the conda_setup.yml file in the flap_fusion/docs directory. The relevant Python packages will be downloaded. to

Unfortunately there is a library collision with MDSPlus. It appears in a way that the MDSShr library fails to load in Python. According to Timo Schröder: “*A quick and dirty fix would be to rename or delete the 'iconv.dll' from "%ANACONDA_HOME%\Library\bin" and hope that nothing else needs it.”* (Note from http://mds-data-1.ipp-hgw.mpg.de/) I deleted this and indeed it helps. It might happen that after an anaconda environment change again the MDSShr library fails. In this case again iconv.dll should be searched for and erased or renamed as anaconda moves files when an environment is changed.

In the Anaconda “Home” page select the “flap” environment (“Applications on:”). Then start Spyder (if you use spyder for Python programs.) In spyder select “Tools→PYTHONPATH manager” and add three directories to the path: the one where all the flap packages are installed, under this the flap_fusion directory and ‘*%HOME*\MDSPlus\python, where ‘*%HOME* ‘ is the user home directory. Then restart Spyder as in most versions the path setting is changed only when restarted.

How to use

Click the start menu and type anaconda, select “Anaconda prompt”. In the window type:

```
  conda activate flap
```

This activates the flap environment. Then type “`spyder`”.

Before using MDSPlus it is advisable to configure it in the `flap_defaults.cfg` file in the working directory. (Working directory is set in the icon line of spyder.) Enter a section into this file:

```
[Module W7X_MDSPlus]
 Server = "mds-data-1"
 Protocol = None
 User = None
 Virtual name file = 'Virtual_names.cfg'
 Cache data = False
 Cache directory = 'C:\Users\zoletnik\Root\W7-X\ABES\Measurement\data'
```

On the virtual machines no user and protocol should be used for accessing the MDSplus, therefore these are set to None and the server is set to mds-data-1. The “Virtual name file” is a file located in the working directory which enables translating MDSPlus names into virtual names, even different ones for different shot ranges. It also enables constructing complex signals from two MDSPlus signals. See example below. The “Cache data” entry (if it is set True) tells flap module to store a copy of the data read in a local file in the “Cache directory”. This way in the second data read call the data it will be taken from the cache file. Also, these files can be copied to another computer and data can be made available without MDSPlus access.

The following example program reads a PCI signal and plots it:

```
import matplotlib.pyplot as plt

import flap
import flap_mdsplus
import flap_w7x_mdsplus

flap_mdsplus.register()
flap_w7x_mdsplus.register()


def test_pci(exp_id='20181018.004', time_interval=[2,3]):
    print("Reading PCI data.")
    flap.get_data('W7X_MDSPlus',exp_id=exp_id,coordinates={'Time':time_interval},
                  name='w7x::QOC.DATA:DET1CH16',
                  object_name='PCI'
                  )
    plt.close('all')
    plt.figure()
    flap.plot('PCI')

test_pci()
```

Example virtual names file:

```
[Virtual names]
# Correlation reflectometry Op 1.2a
CR-B(170901000-180701000) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_03,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_19)
CR-C(170901000-180701000) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_04,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_20)
CR-D(170901000-180701000) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_01,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_17)
CR-E(170901000-180701000) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_02,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_18)
#Correlation reflectometry Op 1.2b
CR-B(180701001-) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_18,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_17)
CR-C(180701001-) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_20,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_19)
CR-D(180701001-) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_04,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_03)
CR-E(180701001-) = complex(\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_02,\QMC::TOP.HARDWARE:ACQ132_168:CHANNELS:INPUT_01)
# PCI OP 1.2b
PCI-1-16(180701001-) = \W7X::QOC.DATA:DET1CH16
```

Note on time scales
In the W7-X MDSPlus database the unit of the signal time axis is not set, this way the general flap_mdsplus module cannot always determine the timescale correctly. According to Timo Scröder in most cases the time unit is ns, but in some cases it is s. As default the flap_w7x_mdsplus module assumes ns. If the timescale is not correct the ‘MDS time unit’ keyword can be used in the flap.get_data call to set it to ‘s’.