Documentation EP workflow

popaalin8100

1 Resources

Some of the resources I found useful when building the Python version of the WF

1. **https://confluence.iter.org/display/IMP/Integrated+Modelling+Home+Page** -> for keeping track of new version of IMAS/PyAL/FC2K (very important!!)

   (a) **https://jira.iter.org/projects/IMAS?selectedItem=com.atlassian.jira.jira-projects-plugin:release-pagestatus=released** -> IMAS dictionary changes

   (b) **https://confluence.iter.org/display/IMP/Access+Layer** -> HDF5 or MDS+ backend for Python

2. **https://user.iter.org/?uid=YSQENWaction=get_document** -> Backend functions documentation for retrieving/manipulating/storing data (Not only Python but also Fortran, C++ and Java)

3. **https://docs.psnc.pl/display/WFMS/FC2K+Python+wrapper+redesign** -> FC2K actor wrapper design (useful for calling an actor after being wrapped by python)

4. **https://confluence.iter.org/display/IMP/iWrap+Python+Actor** -> how to build a python actor

5. **https://confluence.iter.org/display/IMP/4.1+FC2K+Basics** -> small FC2K tutorial for kepler, but the same can be used for Python (just select the python generation)

6. **https://confluence.iter.org/display/IMP/3.2+Fortran+examples** -> 4 examples of Fortran code with IDSs

7. **https://confluence.iter.org/display/IMP/iWrap+-+Fortran+API** -> Fortran API (can be used with FC2K to generate an actor that can be used in python wf)

8. **https://confluence.iter.org/pages/viewpage.action?pageId=289069024** -> working example of the EP WF.

9. **Use the first link to keep track of the working versions of each dependency (most of them do not have backward compatibility!!)**

2 Example

In order to be able to connect the numerical tools with IMAS and to be able to perform time-dependent analysis on any scenario, Energetic Particle Stability Workflow was created. This is the first time-dependent workflow which uses IMAS infrastructure to perform Energetic particle analysis. It is written in Python and makes use also of a simple interface which makes parameter configuration easy for both the connection to the IMAS Database (for saving/retrieving data) and for the numerical codes themselves through a series of XML files. A general layout of the components that the workflow uses can be seen in Fig.1.

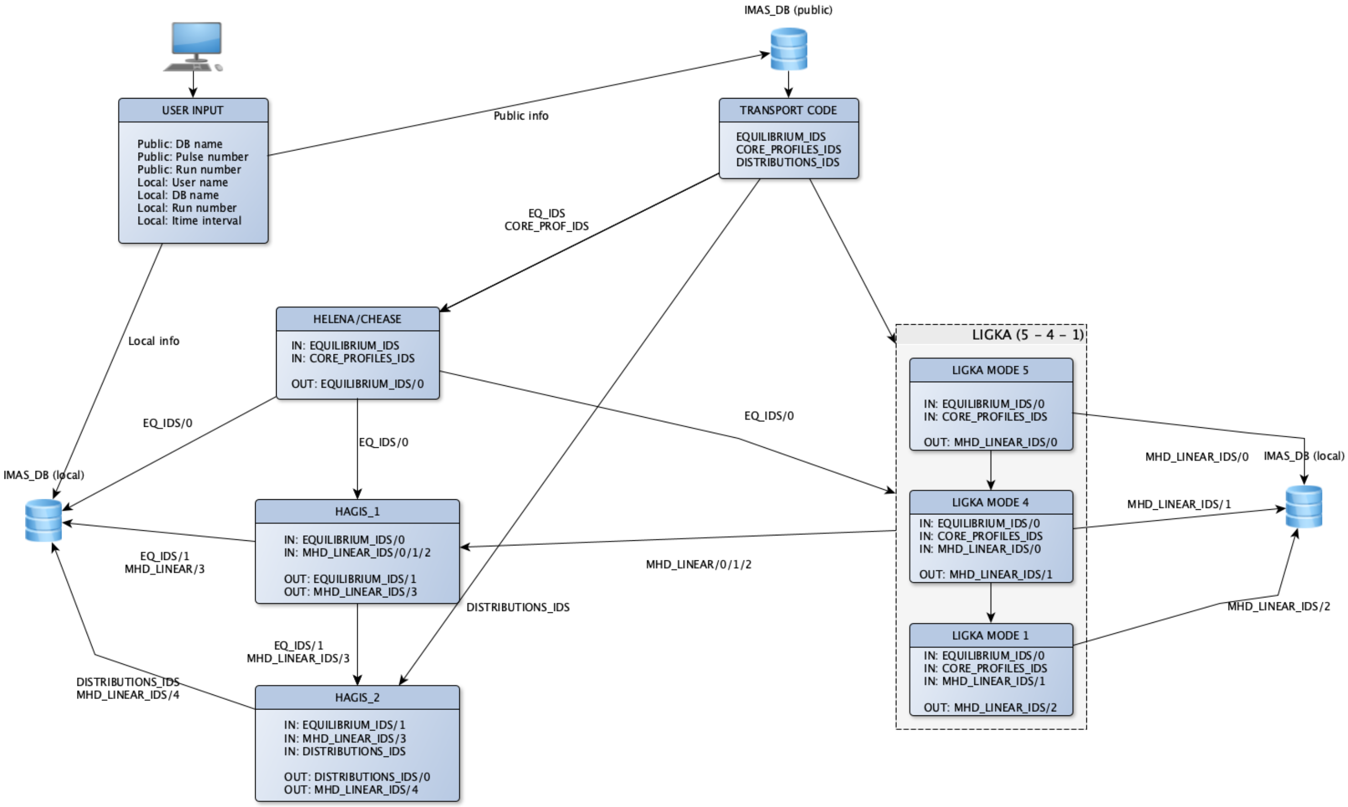
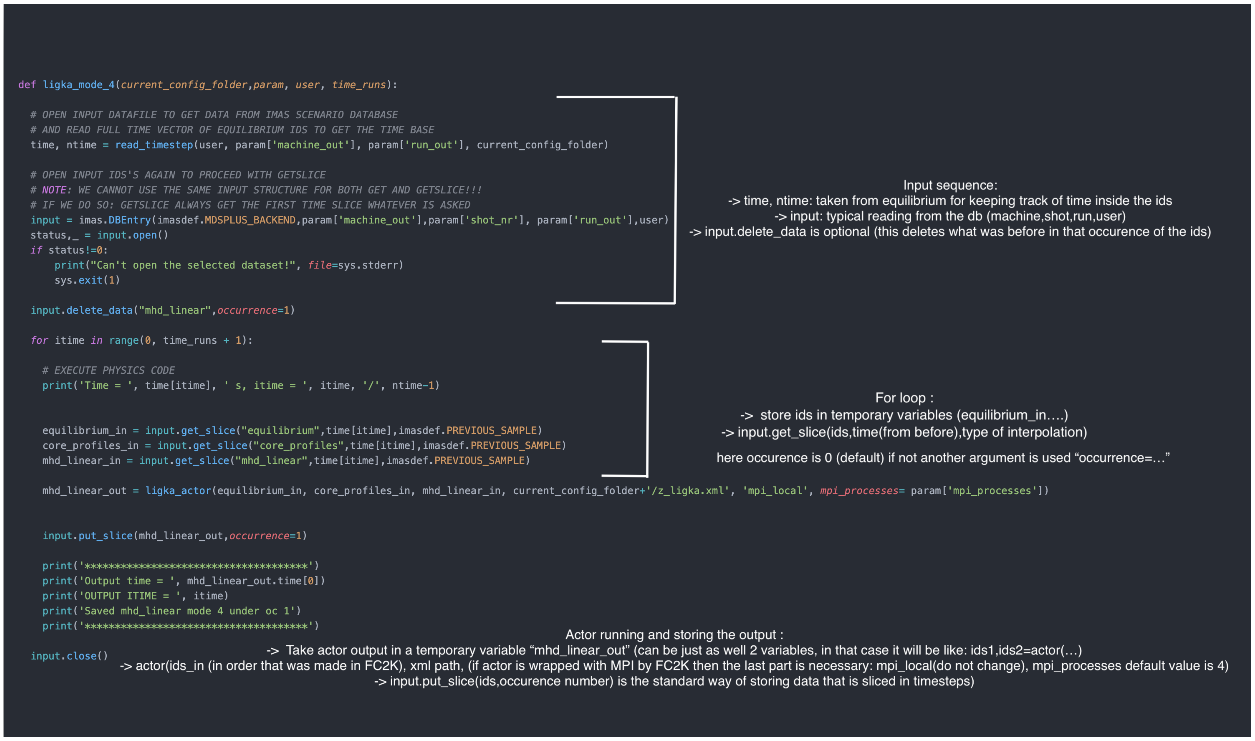


Figure 1: Energetic Particle Stability Workflow general layout of the components.

Now the example that we will use is a MPI actor (mode 4 of LIGKA): Before the actor can be used one needs to import it in python as follows: **from ligka.wrapper import ligka_actor**



```
def ligka_mode_4(current_config_folder,param, user, time_runs):

  # OPEN INPUT DATAFILE TO GET DATA FROM IMAS SCENARIO DATABASE
  # AND READ FULL TIME VECTOR OF EQUILIBRIUM IDS TO GET THE TIME BASE
  time, ntime = read_timestep(user, param['machine_out'], param['run_out'], current_config_folder)

  # OPEN INPUT IDS'S AGAIN TO PROCEED WITH GETSLICE
  # NOTE: WE CANNOT USE THE SAME INPUT STRUCTURE FOR BOTH GET AND GETSLICE!!!
  # IF WE DO SO: GETSLICE ALWAYS GET THE FIRST TIME SLICE WHATEVER IS ASKED
  input = imas.DBEntry(imasdef.MDSPLUS_BACKEND,param['machine_out'],param['shot_nr'], param['run_out'],user)
  status,_ = input.open()
  if status!=0:
      print("Can't open the selected dataset!", file=sys.stderr)
      sys.exit(1)

  input.delete_data("mhd_linear",occurrence=1)

  for itime in range(0, time_runs + 1):

    # EXECUTE PHYSICS CODE
    print('Time = ', time[itime], ' s, itime = ', itime, '/', ntime-1)


    equilibrium_in = input.get_slice("equilibrium",time[itime],imasdef.PREVIOUS_SAMPLE)
    core_profiles_in = input.get_slice("core_profiles",time[itime],imasdef.PREVIOUS_SAMPLE)
    mhd_linear_in = input.get_slice("mhd_linear",time[itime],imasdef.PREVIOUS_SAMPLE)

    mhd_linear_out = ligka_actor(equilibrium_in, core_profiles_in, mhd_linear_in, current_config_folder+'/z_ligka.xml', 'mpi_local', mpi_processes= param['mpi_processes'])


    input.put_slice(mhd_linear_out,occurrence=1)

    print('************************************')
    print('Output time = ', mhd_linear_out.time[0])
    print('OUTPUT ITIME = ', itime)
    print('Saved mhd_linear mode 4 under oc 1')
    print('************************************')

  input.close()
```

Input sequence:
-> time, ntime: taken from equilibrium for keeping track of time inside the ids
-> input: typical reading from the db (machine,shot,run,user)
-> input.delete_data is optional (this deletes what was before in that occurence of the ids)

For loop :
-> store ids in temporary variables (equilibrium_in….)
-> input.get_slice(ids,time(from before),type of interpolation)

here occurence is 0 (default) if not another argument is used "occurrence=…"

Actor running and storing the output :
-> Take actor output in a temporary variable "mhd_linear_out" (can be just as well 2 variables, in that case it will be like: ids1,ids2=actor(…)
-> actor(ids_in (in order that was made in FC2K), xml path, (if actor is wrapped with MPI by FC2K then the last part is necessary: mpi_local(do not change), mpi_processes default value is 4)
-> input.put_slice(ids,occurence number) is the standard way of storing data that is sliced in timesteps)

Figure 2: Example of a typical actor inside a WF.

An example of a working FC2K is the ligka actor: load modules from EP WF by following the tutorial in the confluence page. Then clone ligka and in root of the dir **fc2k** command. Then open the file named **ligka_WF-PY.xml** and check out the parameters/compare them with the ones in the documentation.