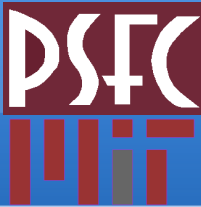




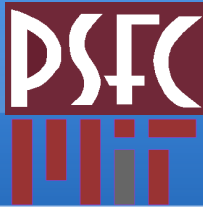
Implementation of an IPS component: a tutorial



J. C. Wright
CSWIM workshop @ ORNL 2007



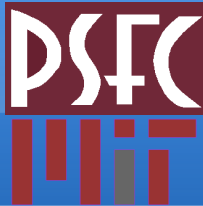
Steps needed to create a working component



1. Port your code to the IPS platform.
2. Call Doug and write pre and post processors for the Plasma State
3. Call Lee and write the python component wrapper
4. Call Lee and write the configuration file for the run you want to do and test.



Porting TORIC to VIZ



1. Port and test TORIC code to the desired Integrated Plasma State (IPS) platform. Document and store the test in SVN.
2. We learned that the XPLASMA interface TORIC had used for equilibrium importing no longer worked with the newest version. Doug McCune made a workaround available, `geqxpl` – has been incorporated into IPS/XPLASMA build.
3. Parallel and Serial test cases were confirmed.



A real CONF file



```
#based on berry's change temp prototype (components/drives/jwright/rfsim_dbb.conf)
#jcw - used to test toric with replay from tsc c-mod run
#10 aug 2007
```

```
IPS_ROOT = /p/swim/jwright/ips/           # Root of IPS component and binary tree
SIM_NAME = TORIC_CM0D044a_jcw           # Name of current simulation
#*
SIM_ROOT = $IPS_ROOT/$SIM_NAME         # Where to put results from this simulation
```

```
#*work directory and naming of files
# Where to put plasma state files as the simulation evolves
PLASMA_STATE_WORK_DIR = $SIM_ROOT/work/plasma_state
RUN_ID = $SIM_NAME
#*
```

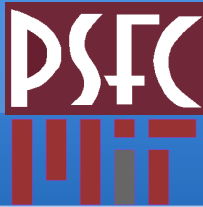
```
OUTPUT_PREFIX =
```

```
CURRENT_STATE = ${RUN_ID}_ps.cdf
PRIOR_STATE = ${RUN_ID}_ps.cdf # ${RUN_ID}_psp.cdf
CURRENT_EQDSK = ${RUN_ID}_ps.geq
```

```
#* What files constitute the plasma state
PLASMA_STATE_FILES = $CURRENT_STATE $PRIOR_STATE $CURRENT_EQDSK
```



A real CONF file



```
PLATFORM = # Simulation Platform (implemented?)
BATCH_SYSTEM = # Which Batch system to use (implemented?)
MPIRUN = mpirun # How are MPI jobs launched (interactively)
SIMULATION_MODE = SINGLE_STEP | RESTART # Simulation mode (is it used yet?)
INITIALIZATION_MODE = # Initialization Mode (implemented?)
MACHINE_CONFIG_FILE = # Machine configuration file (implemented?)
```

#step service in .py driver determines what is actually done

[PORTS]

```
NAMES = DRIVER EPA RF_IC
```

#Driver has custom section to control how components are used.

```
[[DRIVER]] # REQUIRED Port section, see step section of driver
```

```
IMPLEMENTATION = TSC_REPLAY_TORIC_DRIVER # TSC_REPLAY_DRIVER
```

```
[[INIT]] # REQUIRED Port section (check currently disabled)
```

```
IMPLEMENTATION = # Extra special setup stuff
```

```
[[EPA]]
```

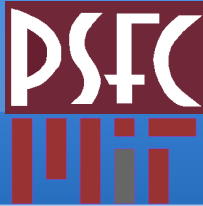
```
IMPLEMENTATION = TSC_REPLAY #loads TSC runs into state files
```

```
[[RF_IC]]
```

```
IMPLEMENTATION = TORIC #wave code component
```



A real CONF file



```
# Individual configuration sections
# Component specification (entries similar for all components)

# NAME entry MUST match the name of the python component class
[TORIC]
CLASS = rf
SUB_CLASS = ic
NAME = toric
NPROC = 4
BIN_PATH = $IPS_ROOT/components/$CLASS/$NAME/src
INPUT_DIR = $IPS_ROOT/components/$CLASS/$NAME/src/CMODIC_A
INPUT_FILES = machine.inp
OUTPUT_FILES = torica.inp machine.inp toric.nc profstat.dat toric.sol \
               equigs.data equidt.data toric.out toric_cfg.nc
SCRIPT = $IPS_ROOT/bin/rf_ic_toric.py
```



A real CONF file



```
[TSC_REPLAY]
```

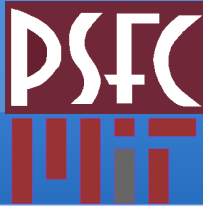
```
CLASS = epa
SUB_CLASS =
NAME = tsc_replay
NPROC = 1
BIN_PATH = $IPS_ROOT/bin
INPUT_DIR = $IPS_ROOT/components/epa/tsc_replay
INPUT_FILES =
OUTPUT_FILES = $PLASMA_STATE_FILES
SCRIPT = $BIN_PATH/epa_tsc_replay.py
REPLAY_DIR=/p/swim/lpku/IPS/my_ips2/trunk/ips/CMOD044a/simulation_results/histor
REPLAY_RUNID = CMOD044a #make sure this points to longpo's files
```

```
[TSC_REPLAY_TORIC_DRIVER]
```

```
CLASS = drivers
SUB_CLASS = berry
NAME = tscReplayAorsaDriver
NPROC = 1
BIN_PATH = $IPS_ROOT/bin
INPUT_DIR = $IPS_ROOT/components/$CLASS/$SUB_CLASS
INPUT_FILES =
OUTPUT_FILES =
SCRIPT = $BIN_PATH/tsc_replay_aorsa_driver.py
```



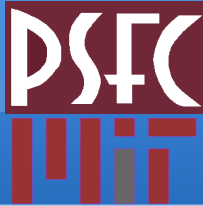
A real CONF file



```
# Time loop specification (two modes for now) EXPLICIT | REGULAR
# For MODE = REGULAR, the framework uses the variables START, FINISH, and NSTEP
# For MODE = EXPLICIT, the frame work uses the variable VALUES
#           (space separated list of time values)
[TIME_LOOP]
MODE = EXPLICIT
START = 0.750
FINISH = 2.000
NSTEP = 2
VALUES = 0.750 2.000
```



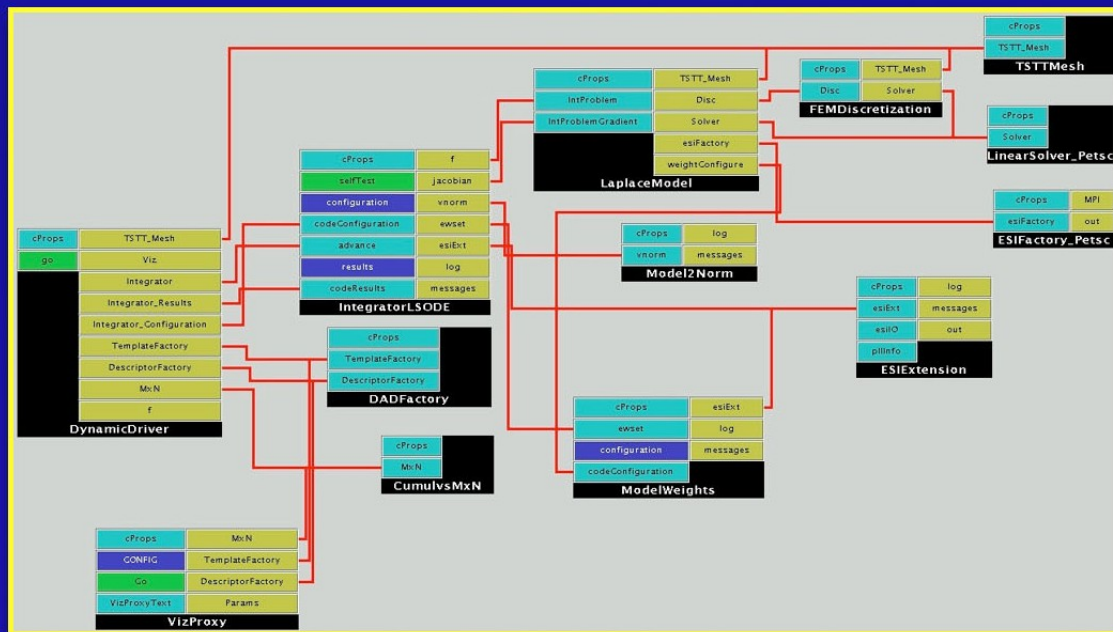
Issue Cards



- TSC or TSC_REPLAY does not create a prior state file. I had to redefine the prior state to be the same as the current state in my .conf file and had to edit epa_tsc_replay.py to define the priorPlasmaState the same as the currentPlasmaState. I don't know what the proper solution to this is, but what I did is just a work around.
- epa_tsc_replay.py had a different naming convention for time stamps than the simulation results in lpku's directory from TSC. His results had timestamps of the form %05.3f, but the python script for the component was trying to read for directories named like %03f. I had to modify the script so the directories were found correctly. This is an inconsistency and I don't know where or in which direction it should be fixed.
- Lack of a machine file. Because we don't have a working machine file, I modified my component script and wrapper preprocessor to read an abbreviate namelist called machine.inp from INPUT_DIR to specify things like nphi, omega, resolution for the run, etc. Probably better to use the MACHINE_CONFIG_FILE variable. And things like resolution don't belong there. I need to think of a way to handle non plasma state determined inputs not related to the machine but that may vary - like resolution.
- The equilibrium quality from TSC wasn't great and caused toric some problems initially, perhaps I can change settings in the equilibrium importer to improve this.



IDEAS



- Wiring diagram for LSODE solution of a PDE

- Image from CCTTSS Scidac using CCAFFIENE
- More published components standards, localize magic numbers and expressions in a 'used' object
- Examples area for conf files with heavy comments