

MPAS-Seaice developer guide

Adrian K. Turner

October 4, 2016

1 Preliminaries

Here we describe preliminary tasks required before MPAS can be installed.

1. Gain access to the MPAS git repository
2. Create a git fork of the MPAS/MPAS-Dev repository on Github
3. Install the libraries required for MPAS: netcdf, pnetcdf and PIO
 - On institutional computing use the pre-compiled library modules:

```
module use /usr/projects/climate/SHARED_CLIMATE/modulefiles/all/  
module load gcc/4.8.2 openmpi/1.6.5 netcdf/4.4.0 parallel-netcdf/1.5.0 pio/1.7.2
```
 - Otherwise, manually install and set the \$NETCDF, \$PNETCDF and \$PIO environment variables. Netcdf version 3.1.4, Pnetcdf version 1.7.0 and PIO version 1.7.2 are known to work.
4. Install fortran compilers including MPI versions.

2 Quick start guide

Here we describe how to download, build and run MPAS-Seaice using the gfortran compiler.

1. Check out the code from the git repository

```
git clone git@github.com:<git_username>/MPAS.git
```

Here <git_username> is your MPAS repository username
2. Move into the MPAS repository

```
cd MPAS
```
3. Checkout the cice/develop branch

```
git checkout cice/develop
```

4. Compile the code


```
make CORE=cice gfortran
```
5. Create a run directory


```
cd ../ ; mkdir rundir ; cd rundir
```
6. Set up the run directory (see section 3; in this case <MPAS_directory> is ../MPAS/)


```
../MPAS/)
```
7. Run the model


```
./cice_model
```

3 Setting up a standard run in a run directory

Here we describe how to set up a standard QU120km run of MPAS-Seaice in a run directory

1. Move to your run directory


```
cd <run_directory>
```
2. Create symbolic links to the grid file, standard forcing files and graph files for a QU120km domain


```
<MPAS-Seaice_data_directory>/domains/domain_QU120km/get_domain.py
```
3. Add the namelist and streams file for a standard test run


```
cp <MPAS-Seaice_data_directory>/configurations/standard_physics/* .
```
4. Copy the namelist and streams file to the names used by MPAS. This step is so a pristine version of these files remain in your run directory even if you modify the ones used by MPAS-seaice.


```
cp namelist.cice.base namelist.cice ; cp streams.cice.base streams.cice
```
5. Get a symbolic link to the MPAS-Seaice executable


```
ln -s <MPAS_directory>/cice_model .
```

Note: On IC machines at LANL <MPAS-Seaice_data_directory> is wibble

4 Development workflow

Here we describe the work flow needed to add new features to the sea ice model.

1. Create a run directory in the base directory.


```
mkdir <feature_directory>
cd <feature_directory>
```

2. Create a development and base directory within this feature directory. Code changes will be made in the development directory and compared during testing against an unchanged version in base.

```
mkdir development
mkdir base
```

3. Create the run directory that will be used for bit reproducible testing

```
cd base/
mkdir rundir
```

4. Clone and compile cice/develop in the base directory.

```
git clone git@github.com:MPAS-Dev/MPAS.git
cd MPAS/
git checkout cice/develop
make CORE=cice gfortran
```

5. Enter the development directory and create a testing run directory

```
cd ../../development
mkdir rundir
```

6. Create a new feature branch from cice/develop to develop the new feature in and compile it.

```
git clone git@github.com:<git_username>/MPAS.git
cd MPAS/
git remote add MPAS-Dev git@github.com:MPAS-Dev/MPAS.git
git fetch --all
git checkout -b cice/<feature_branch_name> MPAS-Dev/cice/develop
make CORE=cice gfortran
```

7. Make the changes to the code needed to implement the desired feature
8. Test that your changes haven't broken the code (see section 5).
9. Commit your changes to the repository (see section 6).
10. Push your changes to your fork on Github and create a pull request for these features to be merged into cice/develop (see section 7).

5 Testing MPAS-Seaice

Currently testing in MPAS-Seaice relies on the development structure as described in section 4.

1. Go to the development run directory
`cd <feature_directory>/development/rundir`
2. Create the testing namelists and streams
`<MPAS-Seaice_data_directory>/configurations/create_testing_configuration.py /
namelist.cice.base streams.cice.base`

This creates a series of namelists and streams files used by the testing scripts based on namelist.cice.base and streams.cice.base
3. Repeat the above in <feature_directory>/base/rundir
4. From <feature_directory>/development/rundir run the testing scripts
`<MPAS-Seaice_data_directory>/testing_scripts/bit_reproducible.py
<MPAS-Seaice_data_directory>/testing_scripts/bit_restartable.py
<MPAS-Seaice_data_directory>/testing_scripts/bit_parallel.py`

6 Committing changes to the repository

Once you are happy with the changes made to your code the changes must be committed to the local repository

1. Move to the MPAS main directory
`cd <MPAS_directory>`
2. See what files have changed since the last commit
`git status`
3. See what has changed within those files
`git diff`
4. Add newly created files to the commit
`git add <new_file>`
5. Add changed files to the commit
`git add <changed_file>`
6. Commit all the changes that have been queued up for commit by the last two sets of command
`git commit`

This should open up a text editor that allows a commit message to be written. Saving and closing this file triggers the commit.
7. Alternatively, all the file changes can be added to the commit in a single command. Note: this doesn't add new files.
`git commit -a`

7 Making pull requests into cice/develop

Once all the changes and new files needed to implement a new feature are implemented then a request must be made to have this feature merged into **cice/develop**. We assume all desired changes have been committed into the local repository as described in section 6.

1. Firstly, its useful to see what branch we are currently on.

```
git branch
```

The current branch is highlighted with an asterix.

2. Move to the branch we want to submit the pull request for (you're most likely already on it if you're following these instructions).

```
git checkout <feature_branch>
```

3. Push the branch to your fork on Github

```
git push origin <feature_branch>
```

4. Go to github.com and find the branch you just pushed there. Click on the submit pull request button and make sure the pull request is into **cice/develop** and not **develop** (which is the default). Add the current MPAS-Seaice integrator (currently **akturner**) as the assignee and add a **Sea ice** label.

5. Go tell the MPAS-Seaice integrator that you have submitted a pull request (currently Adrian Turner). Do not rely on them seeing the automated Github notification email.