

Contents

1	POS_GEOS5PlugMod – a MAPL Ocean Plug for use with Guest_Ocean	2
1.1	SetServices	3
1.2	Initialize	4
1.3	Run	5
1.4	Finalize	6

1 Module POS_GEOS5PlugMod – a MAPL Ocean Plug for use with Guest_Ocean

A MAPL/ESMF Gridded Component that acts as a wrapper for George Mason University's Poseidon hybrid coordinate ocean model.

Poseidon is an object-oriented code that uses the GEMS parallelization libraries for communications, input and output, and the Zeus clock/calendar system. This plug allows the GEMS decomposition, layout and clocks to function in sync with the ESMF versions, then uses the ESMF gridded component superstructure layers for communication by implementing the MAPL framework.

The version of Poseidon used in this implementation recognizes that penetrating radiation and the resultant radiative heating are treated by a separate (sibling) component.

The actual model codes are contained in two sub-directories `./neptune` and `./eq.state`.

1.1 SetServices – Sets ESMF services for for POS_Plug wrapper

INTERFACE:

```
subroutine SetServices ( GC, RC )
```

ARGUMENTS:

```
type(ESMF_GridComp), intent(INOUT) :: GC ! gridded component
integer, optional                :: RC ! return code
```

DESCRIPTION:

The SetServices for the Poseidon.Plug needs to register its Initialize, Run and Finalize. Poseidon has no children.

IMPORT STATE:

Short Name	Units	Dims	Vert Loc	Long Name
TAUX	N m^{-2}			Agrid_eastward_stress_on_skin
TAUY	N m^{-2}			Agrid_northward_stress_on_skin
USTAR3	$\text{m}^3 \text{s}^{-3}$			frictionvelocitycubed
PS	Pa			SurfaceAtmosphericPressure
SWHEAT	W m^{-2}			solar_heating_rate
QFLX	$\text{kg m}^{-2} \text{s}^{-1}$			freshwater_flux_from_skin_to_ocean
HFLX	W m^{-2}			total_heat_flux_into_ocean
SFLX	$\text{kg m}^{-2} \text{s}^{-1}$			salt_flux_into_ocean
TR	1			tracer_mixing_ratios
TRFLUX	X			surface_fluxes_of_tracers_into_ocean

EXPORT STATE:

Short Name	Units	Dims	Vert Loc	Long Name
US	m s^{-1}			top_layer_Agrid_eastward_velocity
VS	m s^{-1}			top_layer_Agrid_northward_velocity
TS	K			top_layer_temperature
SS	psu			top_layer_salinity
DH	dyn m			layer_mass
MASK	none			3D_land-seamask

1.2 Initialize – Initialize method for the for POS_Plug wrapper

INTERFACE:

```
subroutine Initialize ( GC, IMPORT, EXPORT, CLOCK, RC )
```

ARGUMENTS:

```

type(ESMF_GridComp),      intent(INOUT) :: GC      ! Gridded component
type(ESMF_State),        intent(INOUT) :: IMPORT ! Import state
type(ESMF_State),        intent(INOUT) :: EXPORT ! Export state
type(ESMF_Clock),        intent(INOUT) :: CLOCK ! The clock
integer, optional,       intent( OUT) :: RC      ! Error code:

```

DESCRIPTION:

The Initialize method for Poseidon.Plug performs the following tasks.

1. MAPL_Generic initialization. This must come first.
2. Instantiates the POS_MAPL_internal_state
3. Initialize GMAO_gems communications. This needs to match the layout, virtual machine, and MPI Communicator from the ESMF_Grid assigned to this gridded component.
4. Initializes the Poseidon grid in the GMAO_gems communicator. The Poseidon grid information can come from a variety of sources - the restart or a special grid file. The ESMF Configuration file is used to pass this information.
5. Initializes the Poseidon state from its private restart data. At the present, only the ZDF data format is permitted for restart initialization. The name of the restart file is obtained from the Configuration file.
6. Synchronizes the Poseidon clock with the ESMF clock, sets alarms, etc. This initialization assumes that the current time on the ESMF clock represents the initial instant from which the run will be made.
7. Sets up diagnostics and private Poseidon-style history. Poseidon diagnostics are extensive and go well beyond what is available in the state. At the present time, they can only be computed and saved using the Poseidon private history mechanism. A mechanism to export any poseidon diagnostic so that MAPL History can report it is under development.
8. Validates the Poseidon state and parameters as suitable for running.
9. Sets export fields based on the restart conditions. Things like sea surface temperature need to be up-to-date.

Because Poseidon was designed as an object-based model, it has always required a supervisory driver program to perform similar initialization functions. By making the POS.Plug wrapper, the Initialize routine is now required to perform these same initializations. There are numerous internal routines which were lifted from earlier supervisory codes to perform the real work in Initialize.

1.3 Run – Run method for POS_Plug wrapper

INTERFACE:

```
subroutine Run ( gc, import, export, clock, rc )
```

ARGUMENTS:

```
type(ESMF_GridComp), intent(INOUT) :: gc      ! Gridded component
type(ESMF_State),    intent(INOUT) :: import ! Import state
type(ESMF_State),    intent(INOUT) :: export ! Export state
type(ESMF_Clock),    intent(INOUT) :: clock  ! The supervisor clock
integer, optional,   intent( OUT) :: rc      ! Error code:
```

DESCRIPTION:

This routine takes the following steps:

1. Extract the private Poseidon composite state from the gridded component, and associate poseidon style pointers with the internals.
2. Extract the import and export pointers, then copy forcing from the import state to the Poseidon forcing object. Do units conversion and vector rotations here, as well as compute some composite quantities (buoyancy flux) from the defined imports.
3. Set up the private Zeus clock to enable Poseidon to run until the end of the controlling time step. This is simply done by making a “stop_alarm” which will ring at the current (ESMF) time plus one (ESMF) time step.
4. Update the ocean state with the penetrating radiative heating
5. Step Poseidon through its timesteps until “stop_alarm” rings.
6. During this stepping, accumulate private Poseidon history and write it out, if required. Use of Poseidon private history is retained for backward compatibility, but use of MAPL history is now preferred. (The two are NOT mutually exclusive, and both work).
7. After running for the required time, update the export state quantities
8. Exit

1.4 Finalize – Finalize method for POS_Plug wrapper

INTERFACE:

```
subroutine Finalize ( gc, import, export, clock, rc )
```

ARGUMENTS:

```
type(ESMF_GridComp), intent(INOUT) :: gc      ! Gridded component
type(ESMF_State),    intent(INOUT) :: import ! Import state
type(ESMF_State),    intent(INOUT) :: export ! Export state
type(ESMF_Clock),    intent(INOUT) :: clock  ! The supervisor clock
integer, optional,   intent( OUT) :: rc      ! Error code:
```

DESCRIPTION:

Finalize performs the following tasks:

1. Extract the private Poseidon composite state from the component
2. Determine the requested name and type of the private Poseidon checkpoint file
3. Save the checkpoint data. At this time, only the private ZDF format is allowed.
4. Deallocate any space used for history, clocks and the poseidon state.
5. Call MAPL_GenericFinalize for full clean-up