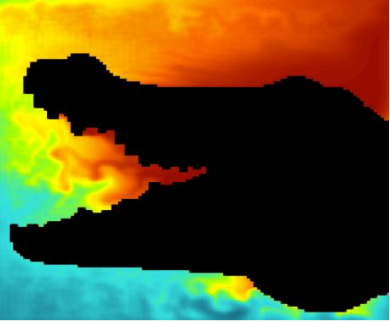
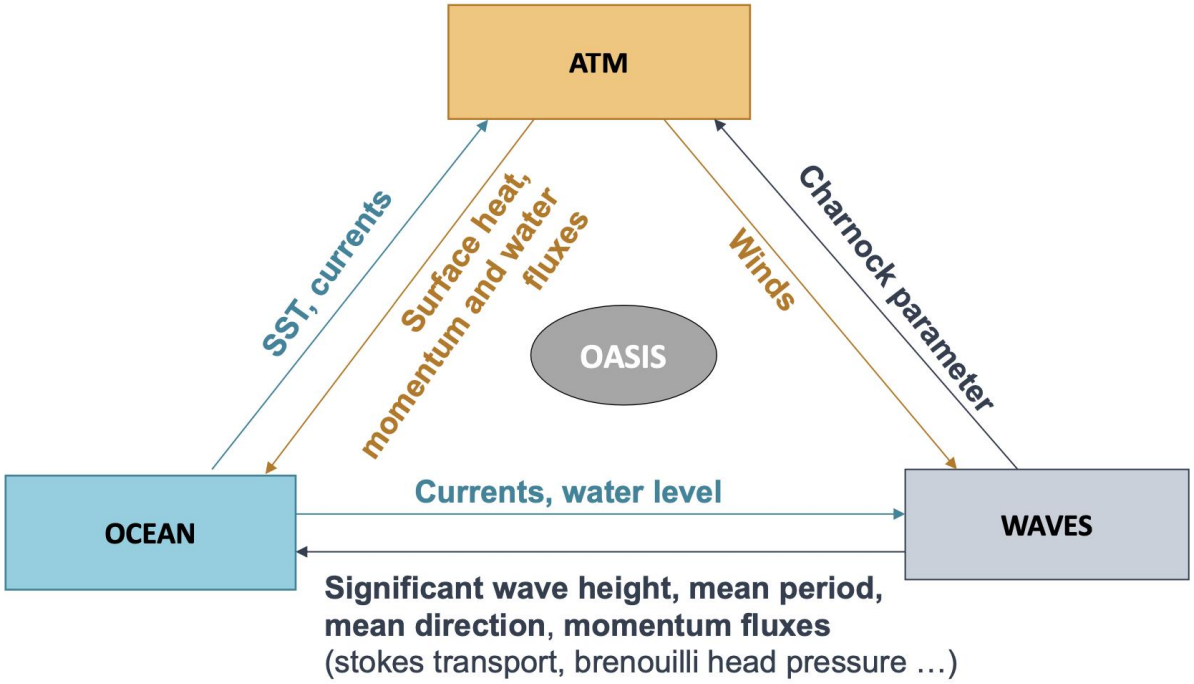


Coupling CROCO with other models using OASIS

Introduction: coupling philosophy

Coupling with OASIS



Coupling with OASIS

OASIS-MCT (Ocean-Atmosphere-Sea-Ice-Soil, Model Coupling Toolkit) is a coupler developed at CERFACS, Toulouse, France.

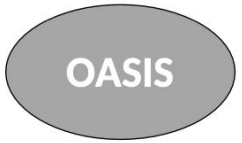
It is a **set of libraries** (not an executable file) providing functions which are called in the models themselves:

- Exchange of variables and time interpolations (PSMILE library)
- Parallel exchanges (MCT library)
- Grid interpolations (SCRIPR library)

It has the **advantage** of being:

- non-intrusive, easy implementation: only a few calls in the model time stepping, and a few additional routines
- A common interface for a variety of models (e.g. CROCO, NEMO, SURFEX, WAVEWATCHIII...)

Coupling with OASIS



Coupling strategy:

On the developer side:

- a few additional routines in each model to specify exchanges
- a few calls in the main model routines (initialization, time stepping, finalization)

On the user side:

- **compilation** with options for coupling in each code and link to the OASIS library
- coupling settings (variables to exchange, coupling frequency, grids...) are controlled thanks to an external file: **namcouple**
- **additional restart files** are created for the coupler
- models are **launched together** at the same time
- a few additional log files to check

Implementation in models

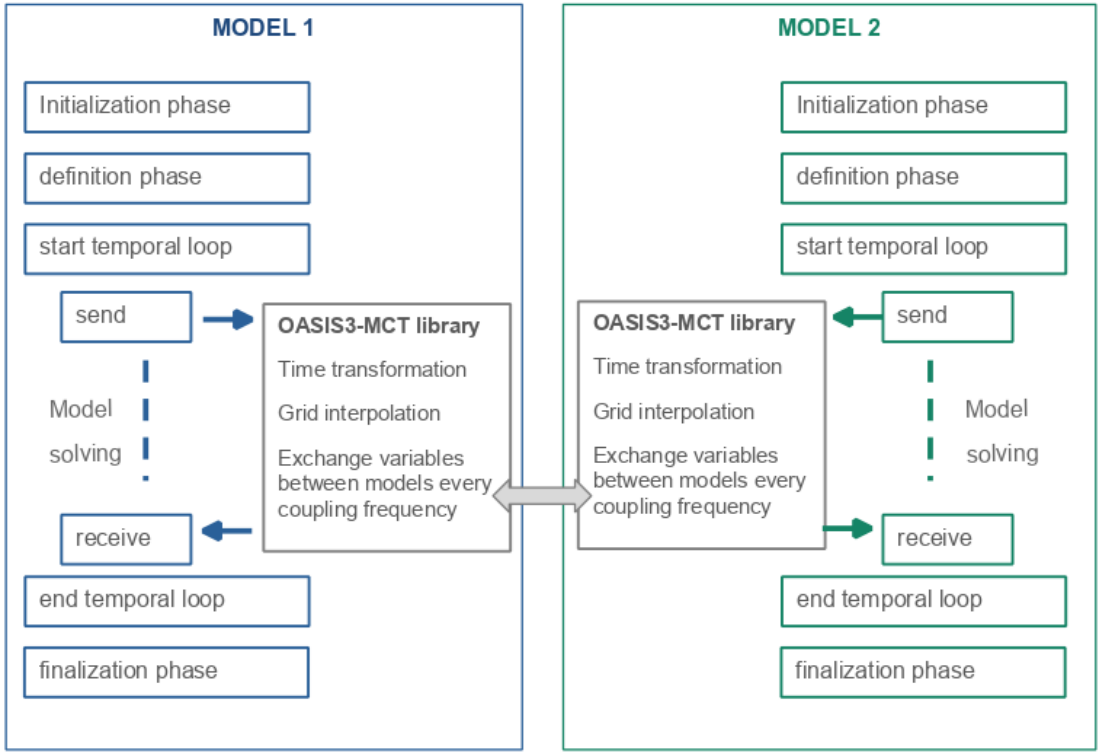
OASIS-MCT implementation calls:

Initialization
oasis_init_comp(...)
oasis_get_localcomm(..)

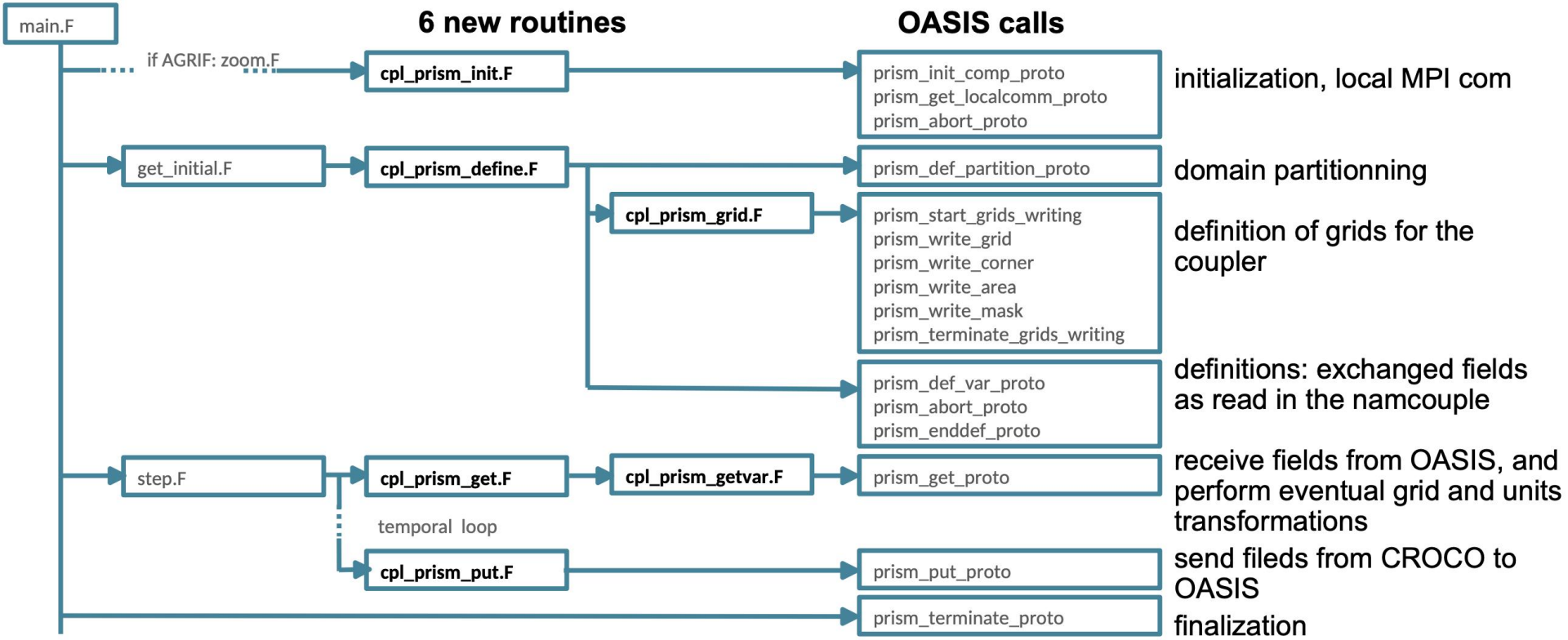
Definitions
oasis_write_grid(...)
oasis_def_partitions(...)
oasis_def_var(...)

Exchange fields(within time stepping)
oasis_put(...)
oasis_get(...)

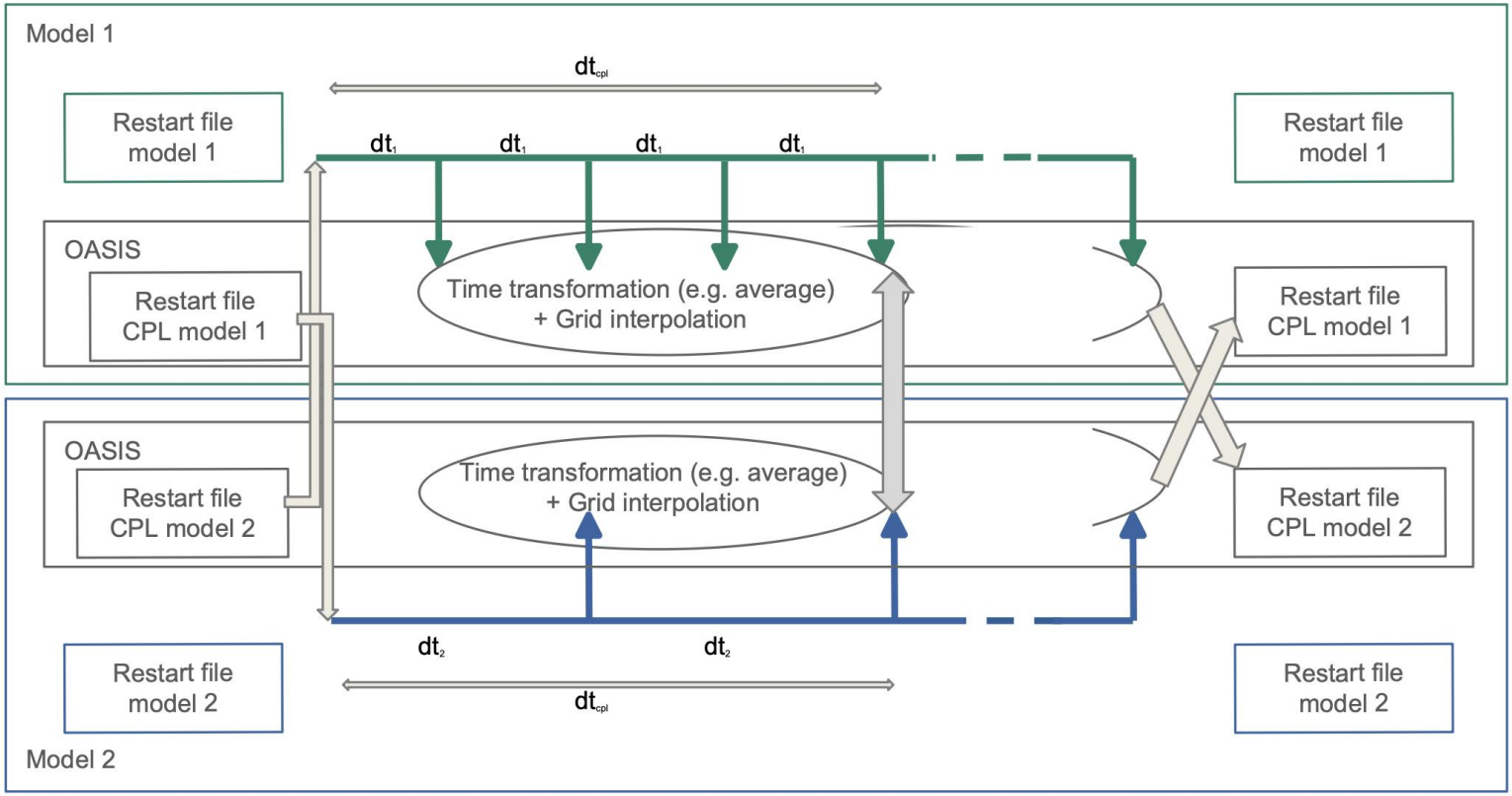
Finalization
oasis_terminate(...)



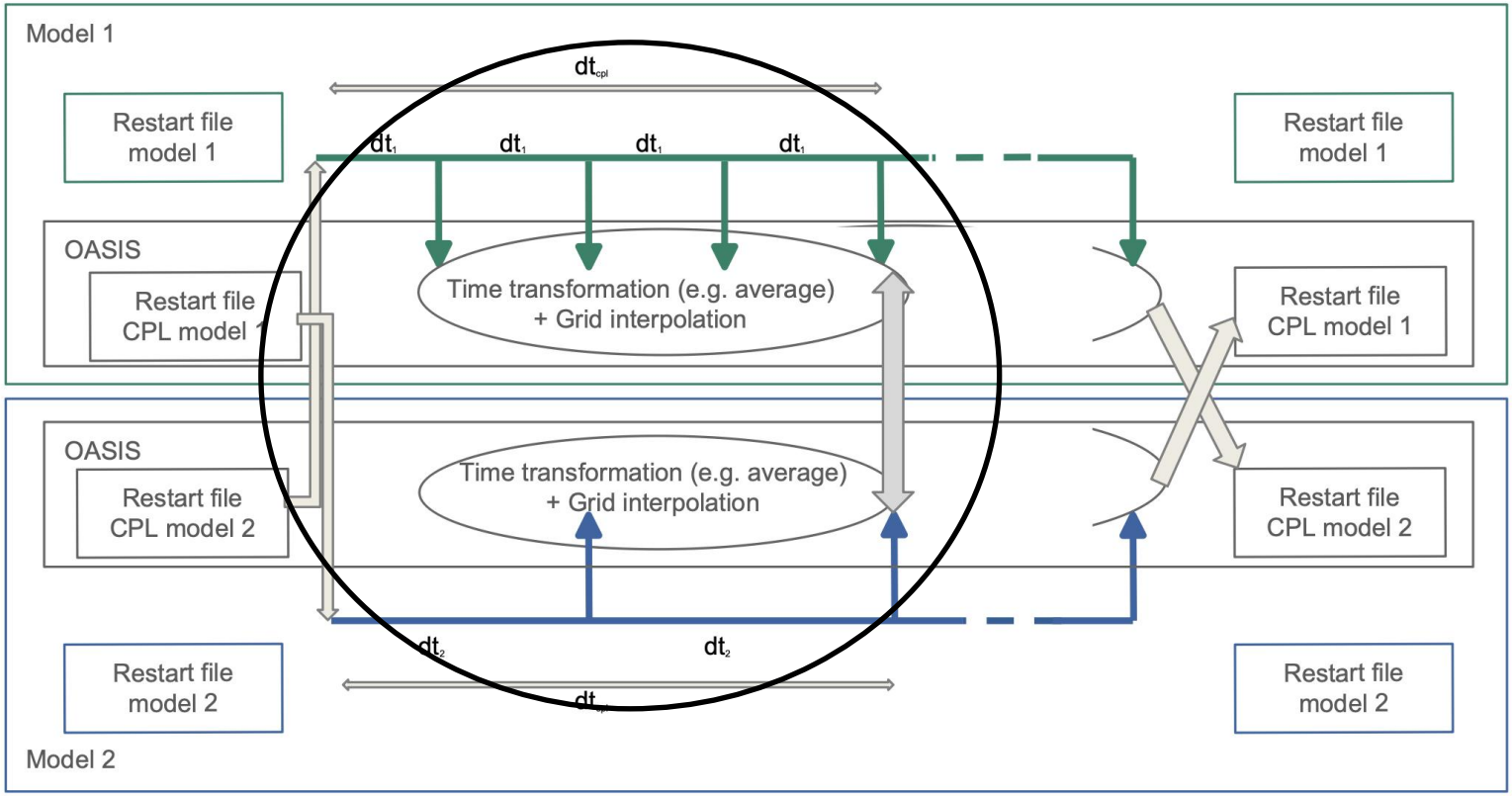
Implementation in CROCO



Coupling sequence



Coupling sequence



The namcouple

The **namcouple** is the text file through which you will specify which fields will be coupled, and how...

A first section gathers the general settings:

```
# NFIELDS: total number of fields being exchanged
 $NFIELDS
 23
#############################################################
# NBMODEL: number of models and their names (6 characters)
 $NBMODEL
 3  wrfexe  wwatch  crocox
#############################################################
# RUNTIME: total simulated time for the actual run in seconds (<I8)
 $RUNTIME
 86400
#############################################################
# NLOGPRT: debug and time statistics informations printed in log file
$NLOGPRT
 1 1
```

The namcouple

The **namcouple** is the text file through which you will specify which fields will be coupled, and how...

A second section provides the information on exchanged fields.
A typical sub-section for one exchanged field looks like:

```
CROCO_SSH WW3__SSH 1 360 1 oce.nc EXPORTED
318 248 318 248 ocnt ww3t LAG=180
R  0  R  0
SCRIPR
DISTWGT LR SCALAR LATLON 1 4
```

Line 1: OASIS name for field in sending model
OASIS name for field in sending model in target model
Unused digit
Coupling period
#of transformations (here 1 interpolation)
Restart file name
Keyword: EXPORTED = field written only at the end of the simulation
EXPOUT = field written at every coupling time step in restart file

Line 2: # of points of the sending and target grids, names of the sending and target grids, LAG=dt of sending model
Line 3: type of grid (**P**eriodical or **R**egional) and # of overlapping points for sending and target models
Line 4: keywords for transformations to perform
Line 5: Parameters for each transformation

Time and grid transformations

The OASIS3-MCT coupler can process:

=> See OASIS manual for more detailed information

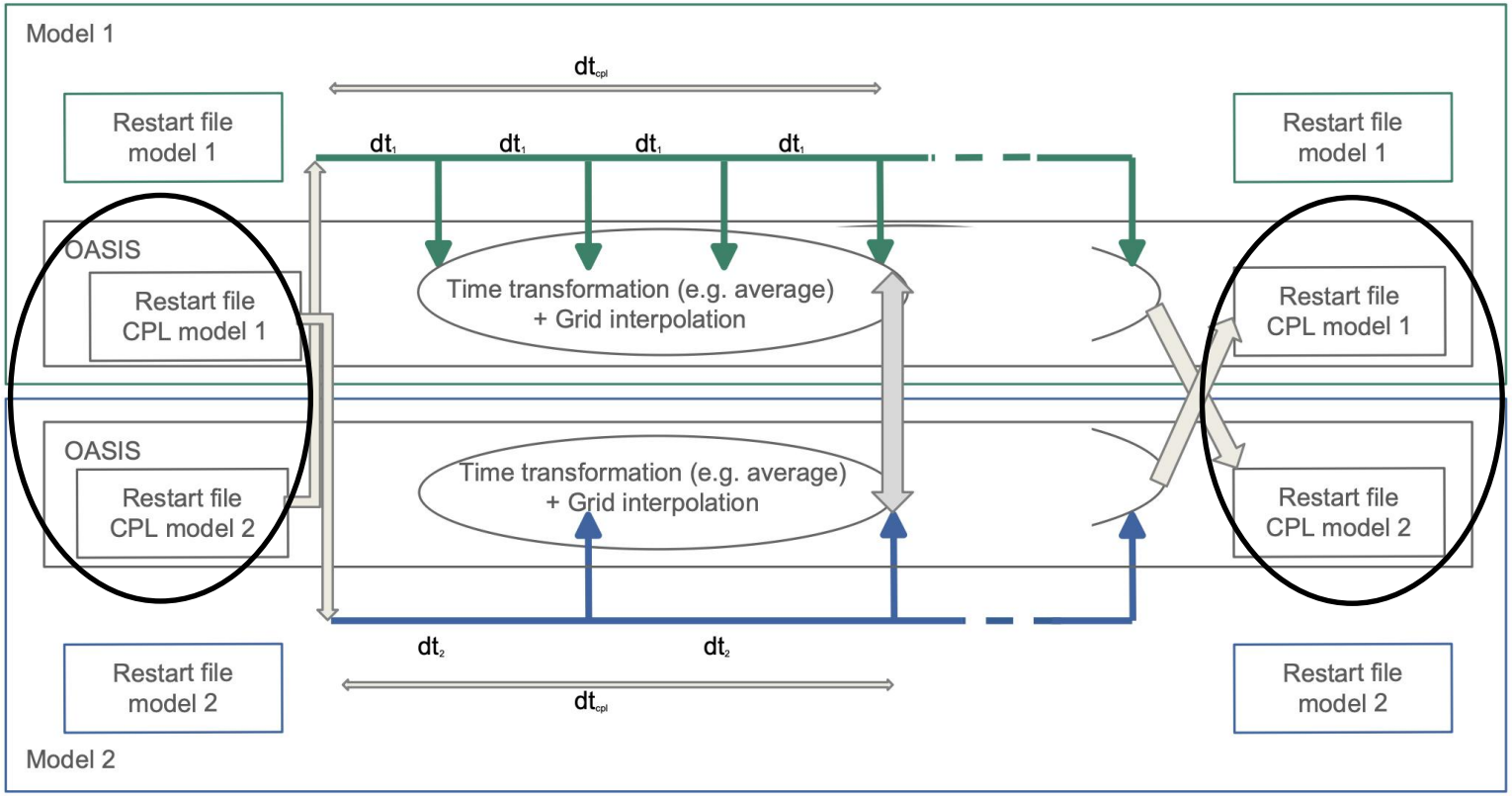
- time transformations (LOCTRANS):

INSTANT no time transformation, the instantaneous field is transferred
ACCUMUL the accumulated field over the coupling period is exchanged
AVERAGE the averaged field over the coupling period is transferred
T_MIN the minimum value of the field for each source grid point over the coupling period is transferred T_MAX the maximum value of the field for each source grid point over the coupling period is transferred

- 2D spatial interpolations (SCRIPR):

BILINEAR interpolation based on a local bilinear approximation
BICUBIC interpolation based on a local bicubic approximation
CONSERV 1st or 2nd order conservative remapping
DISTWGT distance weighted nearest-neighbour interpolation (N neighbours)
GAUSWGT N nearest-neighbour interpolation weighted by their distance and a Gaussian function

Coupling sequence



OASIS-MCT additional files

- **Input files:**
  - **Restart files**: they have to be build for initialization, and they will be automatically written at the end of the simulation for the next one: oce.nc, atm.nc, wav.nc
  - **Namelist** file: namcouple

- **Grid generated files**: these files are requested for interpolations, they are automatically built at the beginning of the simulation by all models: grids.nc, masks.nc, areas.nc
- **Grid interpolation generated files**: these files are built automatically by OASIS according to the previously cited grid files, and to SCRIPR settings in the namcouple: rmp_ww3t_to_ocnt_DISTWGT.nc, rmp_ocnt_to_ww3t_DISTWGT.nc ...

- **Log files**: several log files are produced by OASIS, they should be checked at the end of the simulation or if something goes wrong during the simulation:
  nout.000000 : OASIS log file
  crocox.timers_0000, wwatch.timers_0000 : OASIS log file for time statistics
  debug.root.01, debug.root.02 : log files for the master processor for each model
  debug.notroot.01 : log files for other processors for each model

In practice

In practice: summary of steps

(1) Get the source codes

(2) Set-up your configuration architecture and environment

(3) **Compile** first OASIS and then your codes (CROCO, WRF...) **with the same netcdf libraries and compilers**

(4) Perform **pre-processing** for your different models

(5) Define the **namelists** and **input files** for OASIS and the different models

(6) **Run** the simulation : all models are simultaneously launched

(7) **Outputs**: check log and ouptut files

We can alternatively **use the coupling toolbox** to perform steps 5-6 more easily (and CROCO compilation)

Tools: the coupling toolbox

Coupling toolbox philosophy and workflow:

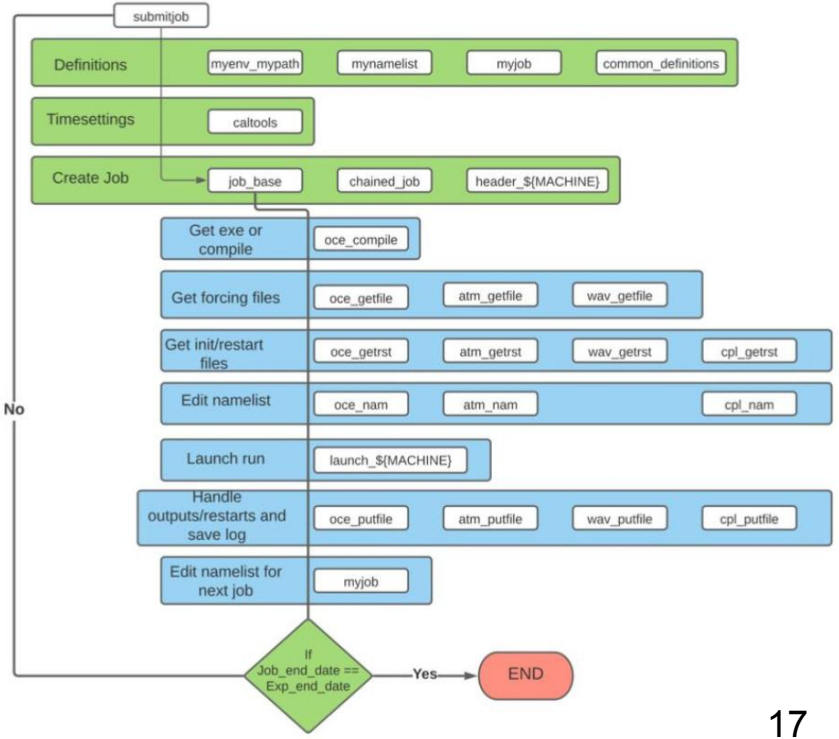
The user edit:

* **myenv_mypath.sh** : environment settings, and paths
* **mynamelist.sh** : settings for the experiment (which models, time stepping, input files…)
* **myjob.sh** : settings for the job (dates notably)

Then the user launch the job with **./submitjob.sh**

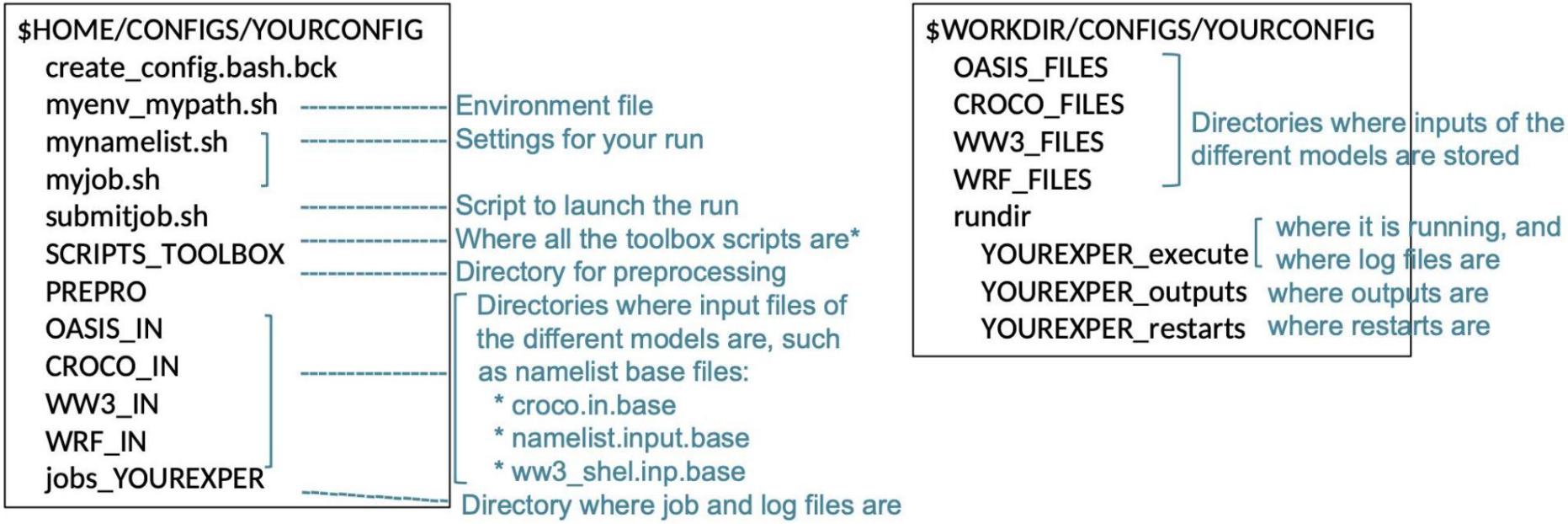
<u>The coupling toolbox manages:</u>

- CROCO compilation if requested
- getting models input files
- preparing OASIS restart files
- editing namelists (for models and OASIS)
- launching the run
- putting output files
- eventually looping for another job



Tools: the coupling toolbox

Coupling toolbox: configuration architecture



* More details: https://croco-ocean.gitlabpages.inria.fr/croco_doc/tutos/tutos.16.coupling.tools.html

Tutorial: let’s go

Outline

- Coupling CROCO with a toy model :
  => simple: a toy model is a little model that mimics another (atmospheric or wave) model
  what it does: it read a model output file (here from WRF or WW3)
  and exchange variables with CROCO through OASIS (send / receive coupled fields)

  - First, you will do all the steps “by hand”
  - Then you will use the coupling toolbox

- Then you will launch a truly coupled run with WRF or WW3

Tutorials available here :
https://croco-ocean.gitlabpages.inria.fr/croco_doc/tutos/tutos.16.coupling.html

Reminder: summary of steps

(1) Get the source codes

(2) Set-up your configuration architecture and environment

(3) **Compile** first OASIS and then your codes (CROCO, WRF...) **with the same netcdf libraries and compilers**

(4) Perform **pre-processing** for your different models

(5) Define the **namelists** and **input files** for OASIS and the different models

(6) **Run** the simulation : all models are simultaneously launched

(7) **Outputs**: check log and outptut files

We can alternatively **use the coupling toolbox** to perform steps 5-6 more easily (and CROCO compilation)

Coupling with a toy “by hand”

(1) Get the source codes

(0) Login and environment file

```
ssh -X userXX@172.20.254.3

cp /home/COMMONDATA/bashrc.netcdf.gcc11 .
source bashrc.netcdf.gcc11
```

(1) Get the source codes

=> already done by us, and copied to your home at the beginning

Refer to the 'Download' tutorial on CROCO doc to do it by yourself on your machine

```
cp -r /home/COMMONDATA/codes/CROCO .
cp -r /home/COMMONDATA/codes/WRF .
cp -r /home/COMMONDATA/codes/OASIS .
mkdir DATA
ln -s /home/COMMONDATA/data_tutos/* DATA/.
mkdir CONFIGS
```

Architecture reminder

Suggested work architecture:

$HOME/CROCO
- croco
- croco_tools

$HOME/OASIS
- oasis3-mct

$HOME/WRF
- WRF
- WPS

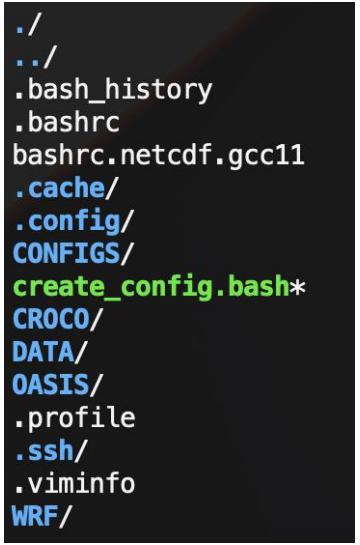
Source codes

$HOME/DATA
- CROCO_DATASETS
- SODA_BENGUELA_LR
- CFSR_GRIB2
- PREPARED_FILES_BENGUELA

Datasets from global reanalyses

$HOME/CONFIGS
- BENGUELA_LR
- BENGUELA_CPL

Your model configurations



(2) Set-up your configuration architecture and environment

$HOME

cp CROCO/croco/create_config.bash .

Edit create_config.bash (e.g. with vi)
=> you need to set-up the paths !

3 options of configuration architectures available:

- "all-dev": for dev of analytical tests
- "all-prod": for production climatological / interannual simulations => provides additional scripts
- "all-prod-cpl" : for coupled simulations (ww3, wrf)=> provides additional scripts

=> choose « all-prod-cpl »

```
#
# Define the paths for your architecture and your dev or prod choice
# ------------------------------------------------------------------

# croco source directory
# ----------------------
CROCO_DIR=$HOME/CROCO/croco

# croco_tools directory
# ---------------------
TOOLS_DIR=$HOME/CROCO/croco_tools

# Configuration name
# ------------------
MY_CONFIG_NAME=BENGUELA_CPL

# Home and Work configuration directories
# ---------------------------------------
MY_CONFIG_HOME=$HOME/CONFIGS
MY_CONFIG_WORK=$HOME/CONFIGS

# Options of your configuration
# -----------------------------
## default option : all-dev for the usual ("all-in") architecture, for
#options=( all-dev )

## example for production run architecture
#options=( all-prod )

## example for production run architecture and coupling with external
options=( all-prod-cpl )

## example for specified options:
#options=( oce-prod prepro inter )
```

(2) Set-up your configuration architecture and environment

$HOME

=> Run the create_config script:

```
./create_config.bash
```

=> It will create a BENGUELA_CPL configuration in your CONFIGS directory

```
cd CONFIGS/
ls -l
cd BENGUELA_CPL
ls -l
```

```
./
../
create_config.bash.bck*
CROCO_FILES/
CROCO_IN/
DATA/
myenv_mypath.sh*
myjob.sh*
mynamelist.sh*
OASIS_IN/
PREPRO/
README_coupling_tools
SCRIPTS_TOOLBOX/
submitjob.sh*
TOY_FILES/
TOY_IN/
WRF_FILES/
WRF_IN/
WW3_FILES/
WW3_IN/
```

(2) Set-up your configuration architecture and environment

$HOME/CONFIGS/BENGUELA_CPL

create_config.bash.bck ---------- Backup of create_config script
myenv_mypath.sh ---------- Environment file

mynamelist.sh
myjob.sh ---------- Scripts for setting up the coupled simulation

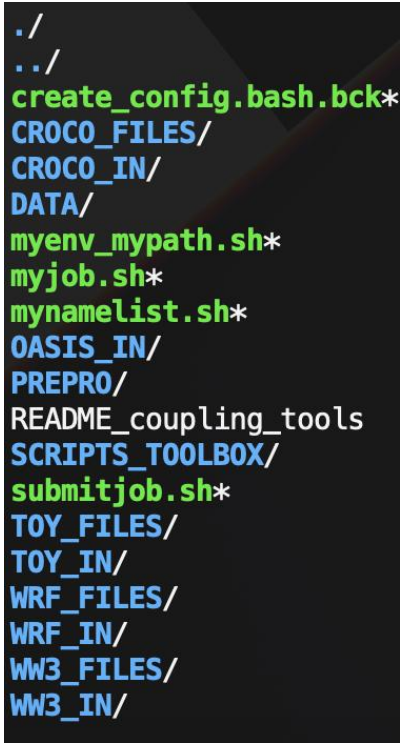
submitjob.sh ---------- Script for launching the job
SCRIPTS_TOOLBOX ---------- Coupling toolbox

PREPRO ---------- Directory for preprocessing (all codes)

CROCO_IN ---------- Directory for CROCO compilation and settings
CROCO_FILES ---------- Directory for CROCO inputs and outputs files

WRF_IN ---------- Directory for WRF compilation and settings/namelist
WRF_FILES ---------- Directory for WRF inputs and outputs files

OASIS_IN ---------- Directory for OASIS namelists



(2) Set-up your configuration architecture and environment

$HOME/CONFIGS/BENGUELA_CPL

Edit **myenv_mypath.sh** to set all the necessary environment variables, modules, and paths for the machine: check the file and eventually edit paths if necessary:

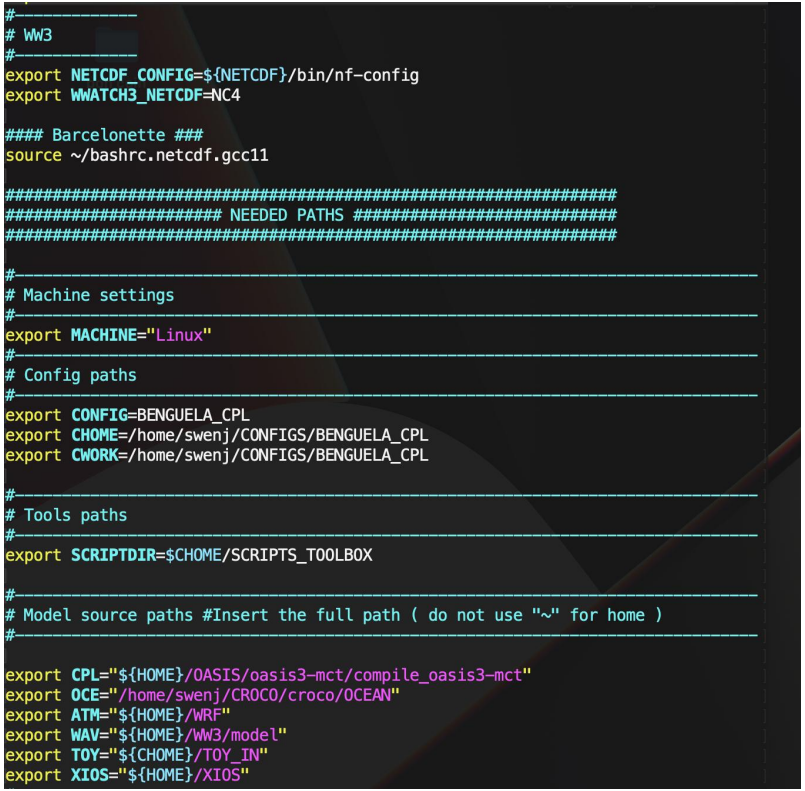
- you need to add source ~/bashrc.netcdf.gcc11 at the end of the environment settings

- you need to change the oasis directory :

export CPL="${HOME}/OASIS/oasis3-mct/compile_oasis3-mct"

Then source it:

```
source myenv_mypath.sh
```



```
#-------------
# WW3
#-------------
export NETCDF_CONFIG=${NETCDF}/bin/nf-config
export WWATCH3_NETCDF=NC4

#### Barcelonette ###
source ~/bashrc.netcdf.gcc11

#############################################################
#################### NEEDED PATHS ###########################
#############################################################

#-------------------------------------------------------------------------------
# Machine settings
#-------------------------------------------------------------------------------
export MACHINE="Linux"
#-------------------------------------------------------------------------------
# Config paths
#-------------------------------------------------------------------------------
export CONFIG=BENGUELA_CPL
export CHOME=/home/swenj/CONFIGS/BENGUELA_CPL
export CWORK=/home/swenj/CONFIGS/BENGUELA_CPL

#-------------------------------------------------------------------------------
# Tools paths
#-------------------------------------------------------------------------------
export SCRIPTDIR=$CHOME/SCRIPTS_TOOLBOX

#-------------------------------------------------------------------------------
# Model source paths #Insert the full path ( do not use "~" for home )
#-------------------------------------------------------------------------------

export CPL="${HOME}/OASIS/oasis3-mct/compile_oasis3-mct"
export OCE="/home/swenj/CROCO/croco/OCEAN"
export ATM="${HOME}/WRF"
export WAV="${HOME}/WW3/model"
export TOY="${CHOME}/TOY_IN"
export XIOS="${HOME}/XIOS"
```

(3) Compilation

(3) Compile first OASIS and then your codes (CROCO, WRF...)

- First compile **OASIS** => already done here by us. Library is in $HOME/OASIS/oasis3-mct/compile_oasis3-mct

- Then, compile your models in coupled mode (**with the same netcdf libraries and compilers**)

* For **CROCO**:

```
cd $HOME/CONFIGS/BENGUELA_CPL/CROCO_IN
```

in cppdefs.h, define: **OA_COUPLING** or/and
OW_COUPLING, and **MRL_WCI**
MPI (**mandatory**)

in jobcomp: edit OASIS library path:
PRISM_ROOT_DIR=$CPL

=> Compile

```
./jobcomp
```

(3) Compilation

(3) Compile first OASIS and then your codes (CROCO, WRF...)

- First compile **OASIS** => already done here by us. Library is in $HOME/OASIS/oasis3-mct/compile_oasis3-mct

- Then, compile your models in coupled mode (**with the same netcdf libraries and compilers**)

  * For the **TOY** model : => already done by us

    You need to copy the relevant executable (and Makefile in case you want to compile by yourself)

```
cd TOY_IN
cp /home/COMMONDATA/codes/TOY/* .
```

(If you want to compile: check Makefile which is already set-up for the current config., compile : make)

(4) Pre-processing

(4) Perform pre-processing for your different models

* For **CROCO**: => already done in your BENGUELA_LR configuration or done by us also

You just need to link the inputs files to your directory:

```
cd $HOME/CONFIGS/BENGUELA_CPL/CROCO_FILES
ln -s $HOME/DATA/CROCO_FILES/* .
```

* For **the TOY model**: => output files from WRF and WW3 available

Link the inputs files to your directory:

```
cd $HOME/CONFIGS/BENGUELA_CPL/TOY_FILES
ln -s $HOME/DATA/TOY_FILES/* .
```

Then create the toy input files from WRF output:

```
$SCRIPTDIR/OASIS_SCRIPTS/create_oasis_toy_files.sh
wrfout_d01_20050101_20050201_fortoya.nc toy_atm.nc wrf '2,249'
```

or WW3 output:

```
$SCRIPTDIR/OASIS_SCRIPTS/create_oasis_toy_files.sh
ww3_20050101_20050131.nc toy_wav.nc ww3 '1,124'
```

(5) Define namelist and input files

$HOME/CONFIGS/BENGUELA_CPL

First create a directory to run the model by hand, and copy the necessary files:
- namelists : namcouple, TOYNAMELIST.nam, croco.in
- executables : crocox, toyexe
- input files : toy_atm.nc, grid_atm.nc or toy_wav.nc, grid_wav.nc

```
mkdir run_byhand
cd run_byhand

cp ../TOY_IN/namcouple_example_oa namcouple
cp ../TOY_IN/TOYNAMELIST.nam_example_oa TOYNAMELIST.nam
cp ../TOY_IN/toy_model toyexe
cp ../TOY_FILES/toy_atm.nc .
cp ../TOY_FILES/grid_atm.nc .

cp ../CROCO_IN/croco crocox
cp $HOME/CROCO/croco/OCEAN/croco.in .
```

(5) Define namelist and input files

Set up the **namcouple**:

```
#########################################################
# This is a typical input file for OASIS3-MCT.
# Keywords used in previous versions of OASIS3
# but now obsolete are marked "Not used"
# Don't hesitate to ask precisions or make suggestions (oasishelp@cerfacs.fr).
#
# Any line beginning with # is ignored. Blank lines are not allowed.
#
#########################################################
# NFIELDS: total number of fields being exchanged
$NFIELDS
9
#########################################################
# NBMODEL: number of models and their names (6 characters)
$NBMODEL
2 toyexe crocox
#########################################################
# RUNTIME: total simulated time for the actual run in seconds (<I8)
$RUNTIME
259200
#########################################################
...
```

9 fields exchanged :

- 3 from Ocean => Atmosphere
- 6 from Atmosphere => Ocean

Name of the model [for oasis]

Run duration 3 days [in second]

for each variable:

```
#~~~~~~~~~~~
# SST (K)
#~~~~~~~~~~~
CROCO_SST TOY__SST 1 10800 2 oce.nc EXPORTED
41 42 56 50 ocnt toyt LAG=3600
R  0  R  0
LOCTRANS SCRIPR
AVERAGE
DISTWGT LR SCALAR LATLON 1 4
```

(5) Define namelist and input files

Set up the **croco.in**:

Change NTIMES
Change the paths to the input files
And to the output files
Change the NAVG and NWRT

```
title:
        BENGUELA TEST MODEL
time_stepping: NTIMES   dt[sec]  NDTFAST  NINFO
                 72      3600      60       1
time_stepping_nbq: NDTNBQ     CSOUND_NBQ     VISC2_NBQ
                     1          1000          0.01
S-coord: THETA_S,   THETA_B,    Hc (m)
           7.0d0     2.0d0      200.0d0

start_date:
2000-01-01 00:00:00

end_date:
2000-02-01 00:00:00

xios_origin_date:
2014-01-04 00:00:00

output_time_steps: DT_HIS(H), DT_AVG(H), DT_RST(H)
                     1       6       12

grid:  filename
    ../CROCO_FILES/croco_grd.nc
forcing: filename
    ../CROCO_FILES/croco_frc.nc
bulk_forcing: filename
    ../CROCO_FILES/croco_blk.nc
climatology: filename
    ../CROCO_FILES/croco_clm.nc
boundary: filename
    ../CROCO_FILES/croco_bry.nc
initial: NRREC / filename
          1
    ../CROCO_FILES/croco_ini.nc
restart:          NRST, NRPFRST / filename
                   720    -1
    ./croco_rst.nc

history: LDEFHIS, NWRT, NRPFHIS / filename
            T       24     0
    ./croco_his.nc
averages: NTSAVG, NAVG, NRPFAVG / filename
           1      24     0
    ./croco_avg.nc
```

(5) Define namelist and input files

Set up the **TOYNAMELIST.nam**:

```
&NAM_OASIS NB_TIME_STEPS=24,
       DELTA_T=10800,
       GRID_FILENAME='grid_atm.nc' /

&NAM_FCT_SEND CTYPE_FCT='FILES',
         CNAME_FILE='toy_atm.nc',
         VALUE=10 /

&NAM_RECV_FIELDS NB_RECV_FIELDS=3,
            CRCVFIELDS(1)='TOY__SST',
            CRCVFIELDS(2)='TOY_UOCE',
            CRCVFIELDS(3)='TOY_VOCE' /

&NAM_SEND_FIELDS NB_SEND_FIELDS=6,
            CSNDFIELDS(1)='TOY_TAUX',
            CSNDFIELDS(2)='TOY_TAUY',
            CSNDFIELDS(3)='TOY_TAUM',
            CSNDFIELDS(4)='TOYSTFLX',
            CSNDFIELDS(5)='TOYSRFLX',
            CSNDFIELDS(6)='TOY__EMP' /
```

Number of time step of 3 hours read and send

"dt" of the atm toy model in second = 3h. It drives the coupling frequency

The wrf output file read by the atm toy model

The fields received from croco through oasis coupler

The fields send to croco through oasis coupler

(5) Define namelist and input files

Create restart files for OASIS: see create_oasis_restart_from_calm_conditions.sh

For the atmospheric variables:

```
$SCRIPTDIR/OASIS_SCRIPTS/create_oasis_restart_from_calm_conditions.sh
../TOY_FILES/wrfout_d01_20050101_20050201_fortoya.nc atm.nc wrf "TOY_V_01 TOY_U_01 TOY_TAUX TOY_TAUY
TOY_TAUM TOYSRFLX TOYSTFLX TOY__EMP  TOY_PSFC"
```

For the wave variables :

```
$SCRIPTDIR/OASIS_SCRIPTS/create_oasis_restart_from_calm_conditions.sh ../TOY_FILES/ww3_20050101_20050131.nc
wav.nc ww3 "TOY_T0M1 TOY___HS TOY__DIR TOY_TWOX TOY_TWOY TOY_TAWX TOY_TAWY TOY__CHA"
```

For the oceanic variables:

```
 $SCRIPTDIR/OASIS_SCRIPTS/create_oasis_restart_from_calm_conditions.sh ../CROCO_FILES/croco_grd.nc oce.nc croco
"CROCO_SST CROCO_SSH CROCO_NOCE CROCO_EOCE"
```

(6) Run the simulation

```
mpirun -n 1 toyexe : -n 2 crocox
```

(7) Outputs: check log and outptut files

Outputs are:

croco_his.nc
croco_avg.nc

Log files are:

nout.000000
crocox.timers_0000
toyexe.timers_0000
debug.01.000000
debug.02.000001
debug.02.000000

OASIS log files

OUTPUT_TOY.txt
croco.log

Model log files

Coupling with an atmospheric or wave model **using the coupling toolbox**

Using the coupling toolbox

First copy input files for the models : we have already perform pre-processing for you.

* For **WRF**: => already done by us

You just need to link the inputs files to your directory:

```
cd $CWORK/WRF_FILES
ln -s $HOME/DATA/WRF_FILES/* .
```

* For **CROCO**: => already done in your BENGUELA_LR configuration or done by us also

You just need to link the inputs files to your directory:

```
cd $CWORK/CROCO_FILES
ln -s $HOME/DATA/CROCO_FILES/* .
```

Using the coupling toolbox

Coupling toolbox philosophy and workflow:

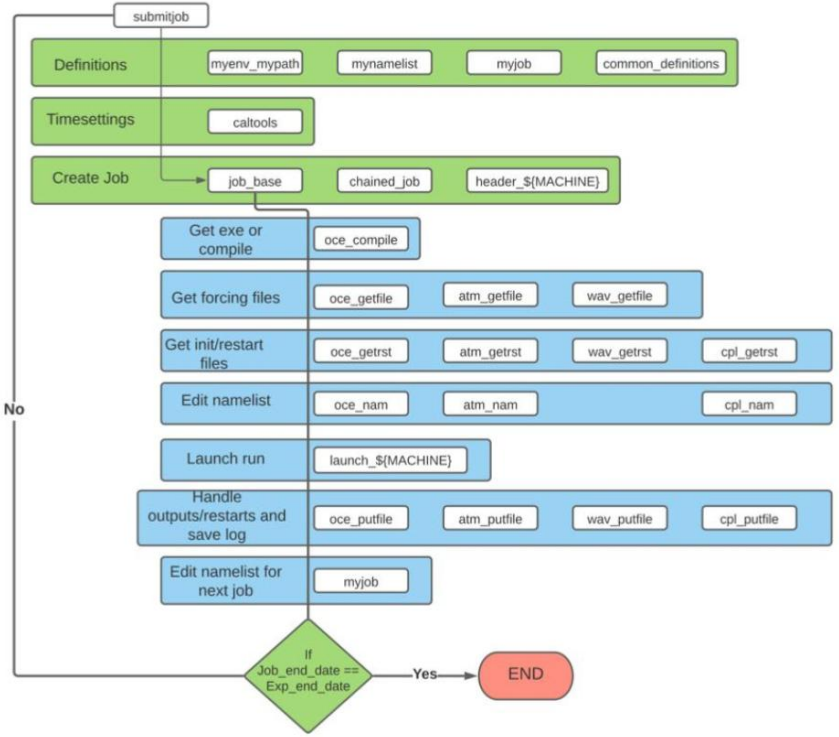
The user edit:

* **myenv_mypath.sh** : environment settings, and paths
* **mynamelist.sh** : settings for the experiment (which models, time stepping, input files…)
* **myjob.sh** : settings for the job (dates notably)

Then the user launch the job with **./submitjob.sh**

<u>The coupling toolbox manages:</u>

- CROCO compilation if requested
- getting models input files
- preparing OASIS restart files
- editing namelists (for models and OASIS)
- launching the run
- putting output files
- eventually looping for another job



Using the coupling toolbox

$HOME/CONFIGS/BENGUELA_CPL

Only 3 files need to be modified:

myenv_mypath.sh	environment settings, and paths
mynamelist.sh	settings for the experiment (which models, time stepping, input files…)
myjob.sh	settings for the job (dates, mpi settings…)

Using the coupling toolbox

$HOME/CONFIGS/BENGUELA_CPL

Edit **myenv_mypath.sh** to set all the necessary environment variables, modules, and paths for the machine: check the file and eventually edit paths if necessary:

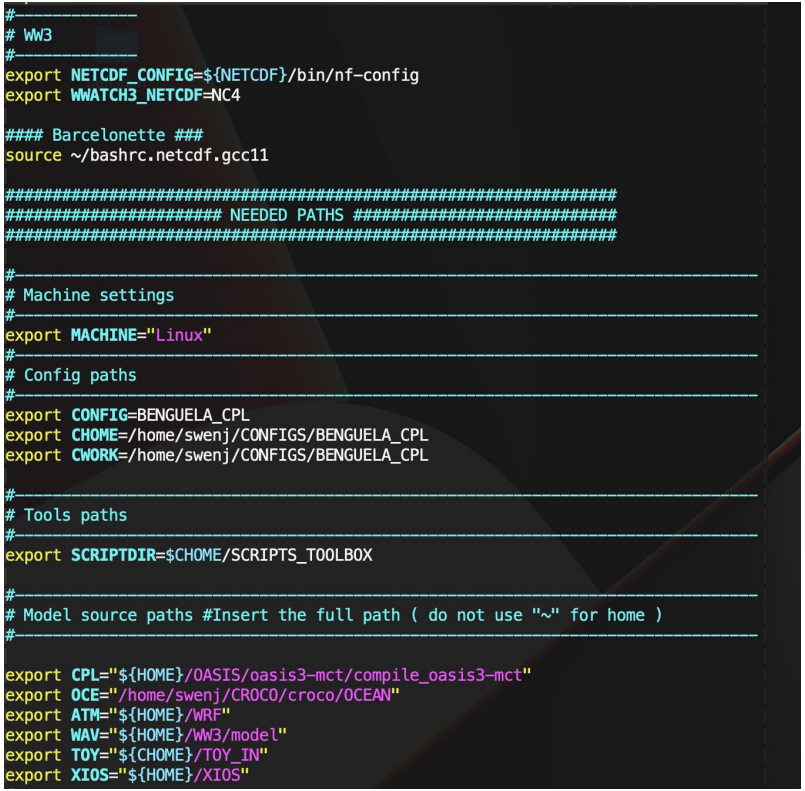
- you need to add source ~/bashrc.netcdf.gcc11 at the end of the environment settings

- you need to change the oasis directory :

export CPL="${HOME}/OASIS/oasis3-mct/compile_oasis3-mct"

Then source it:

source myenv_mypath.sh



Using the coupling toolbox

$HOME/CONFIGS/BENGUELA_CPL

Edit **mynamelist.sh**

Set up for OA coupled run

RUNtype=oa
USE_ATM=1
USE_OCE=1
others=0

Choose online compilation of CROCO

ONLINE_COMP=1

```
#################################################
########################## USER CHANGES ##########
#################################################
#
export CEXPER=BENGUELA_CPL_exp1  # (max 30. CHAR) Nam
export RUNtype=oa # Kind of run launched. Summaries w
 See in OASIS_IN dir for more o,w,a order details
#
export USE_ATM=1
export USE_XIOS_ATM=0
export USE_OCE=1
export USE_XIOS_OCE=0
export USE_WAV=0
export USE_TOYATM=0
export USE_TOYOCE=0
export USE_TOYWAV=0
```

```
#------------------------------------
# OCE
#------------------------------------
# namelist [Info: grid size is directly

# Online Compilation
# Creates croco executable depending on
#     In cppdefs.h options that can be
IOS/AGRIF/AGRIF_2WAY
#     In param.h it modifies the grid s
export ONLINE_COMP=1
```

Using the coupling toolbox

$HOME/CONFIGS/BENGUELA_CPL

Edit **myjob.sh**

Choose the period of the run:

=> just a few days

JOB_DUR_MTH=0

JOB_DUR_DAY=5

Choose the MPI decomposition:

=> 1 CPU for CROCO, 4 for WRF

NP_OCEX=1

NP_OCEY=1

NP_ATM=2

```
#-------------------------------------
# Job submission settings
#-------------------------------------
# Job walltime
export TIMEJOB=1800

# Project Id (on which hours are taken, if
export projectid=""

# Permissions given to output files
export permission="755"
#-------------------------------------
# Run date settings
#-------------------------------------
# Your run can be divided into several job

# Start date of the first Job
export YEAR_BEGIN_JOB=2005
export MONTH_BEGIN_JOB=1
export DAY_BEGIN_JOB=1

# Duration of each Job
export JOB_DUR_MTH=0
export JOB_DUR_DAY=10

# How many jobs do you want to launch?
export NBJOB=1

# Do we start from a restart?
export RESTART_FLAG="FALSE"
```

```
#-------------------------------------
# Number of core used
# ------------------------------------
# mpi launch command: ccc_mprun :for i
MPI_LAUNCH_CMD=$MPI_LAUNCH
export SERIAL_LAUNCH_WAV="$MPI_LAUNCH -

# nb of CPUs for each model
# < insert here CPU > !!! DO NOT REMOVE
export NP_OCEX=1
export NP_OCEY=1
export NP_WAV=14
export NP_ATM=4
export NP_TOY=1
export NP_XIOS_ATM=1
export NP_XIOS_OCE=1

# additional MPI Settings for ATM (WRF
export atm_nprocX=-1      # -1 for auto
export atm_nprocY=-1      # -1 for auto
export atm_niotaskpg=0    # 0 for defau
export atm_niogp=1        # 1 for defau
```

Using the coupling toolbox

$HOME/CONFIGS/BENGUELA_CPL

Launch the script to submit the job and run the simulation:

```
./submitjob.sh >& job.log
```

Using the coupling toolbox

$HOME/CONFIGS/BENGUELA_CPL

Check log and output files:

```
rundir
jobs_BENGUELA_CPL_exp1
myjob.sh
job.log
```

job.log ---------- your direct log file

jobs_BENGUELA_CPL_exp1 ---------- where you can find the job that was launched, the setting files used, and various log files

```
./
../
croco.log
crocox.timers_0000
job_BENGUELA_CPL_exp1_20050101_20050110.sh*
myjob.sh*
mynamelist.sh*
namelist.input*
nout.000000
out_run.txt
rsl.error.0000
rsl.out.0000
```

rundir ---------- where the run has been run and where outputs are stored

Log files are in execute

```
BENGUELA_CPL_exp1_execute/
BENGUELA_CPL_exp1_outputs/
BENGUELA_CPL_exp1_restarts/
```

myjob.sh ---------- Scripts for setting up the next coupled simulation

Output and log files

In case of error, you should check:

- The job output file: in $CHOME/jobs_YOUREXPER :
  YOUREXPER_YYYYMMDD_YYYYMMDD.o*
- The models' log files: either in
  $C**HOME/jobs_YOUREXPER**/YYYYMMDD_YYYYMMDD or in
  $**CWORK**/rundir/**YOUREXPER_execute**/YYYYMMDD_YYYYMMDD
  croco.log rsl.error.0000 log.ww3
- OASIS log files:
  nout.000000 debug.0?.000000

NB! Usually OASIS errors are in debug.root.0X, and model erros are either in their log, or in the standard output (may be in the batch log if no redirection)

Typical issues are:

- Files not found: check your file names, and location
- Unconsitent dimensions of the grids in the different files: check models grid files, OASIS grids and masks files, OASIS remapping weight files, namelists of models and OASIS (namcouple)
- Unconsistency in exchanged variables: check namcouple
- Model blow up: check the log files, if blow up is due to CFL (unrealistic speed, or segmentation fault) decrease the model time step

NB! If grids.nc or rmp*.nc files exist they will not be re-generated (useful for restart run, but can be source of error…)

Output and log files

Batch log file: YOUREXPER_YYYYMMDD_YYYYMMDD.o*

OASIS

Generated grid files:
- grids.nc
- masks.nc
- areas.nc

NB! If grids.nc or rmp*.nc files exist they will not be re-generated (useful for restart run, but can be source of error…)

Generated grid interpolation files:
- rmp_ww3t_to_ocnt_DISTWGT.nc
- rmp_ocnt_to_ww3t_DISTWGT.nc …

Generated restart files:
- rst_oce.nc, rst_atm.nc, rst_wav.nc
- (overwritten at the end of each simulation)

Logs:
- nout.000000
- crocox.timers_0000, wwatch.timers_0000
- debug.root.01, debug.root.02, …
- debug.notroot.01, debug.notroot.02, …

CROCO

Output files:
- croco_his.nc
- croco_avg.nc
- croco_rst.nc

Logs:
- croco.log
- + eventually standard output redirected to batch log file if LOGFILE cppkey not defined

WW3

Output files:
- out_grd.ww3 => ww3.DATE.nc
- out_pnt.ww3 => ww3.DATE_spec.nc

Logs:
- log.ww3
- output.ww3

WRF

Output files:
- wrfout_d01_DATE
- wrfxtrm_d01_DATE
- wrfrst_d01_DATE

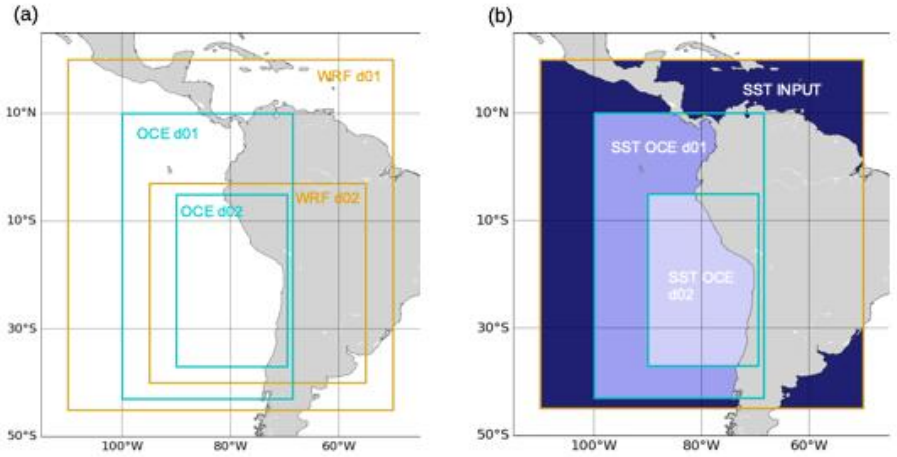
Logs:
- rsl.error.0000
- rsl.out.0000

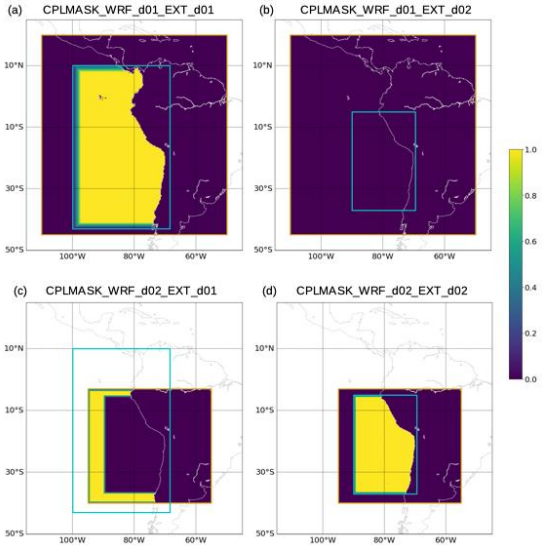
Appendices

Coupling with nests

- nesting and cplmask

- Possible to couple multiple domains → use of a « coupling mask »
  + with WRF moving nest : coupling on the parent domain (with interpolation from moving nest(s))

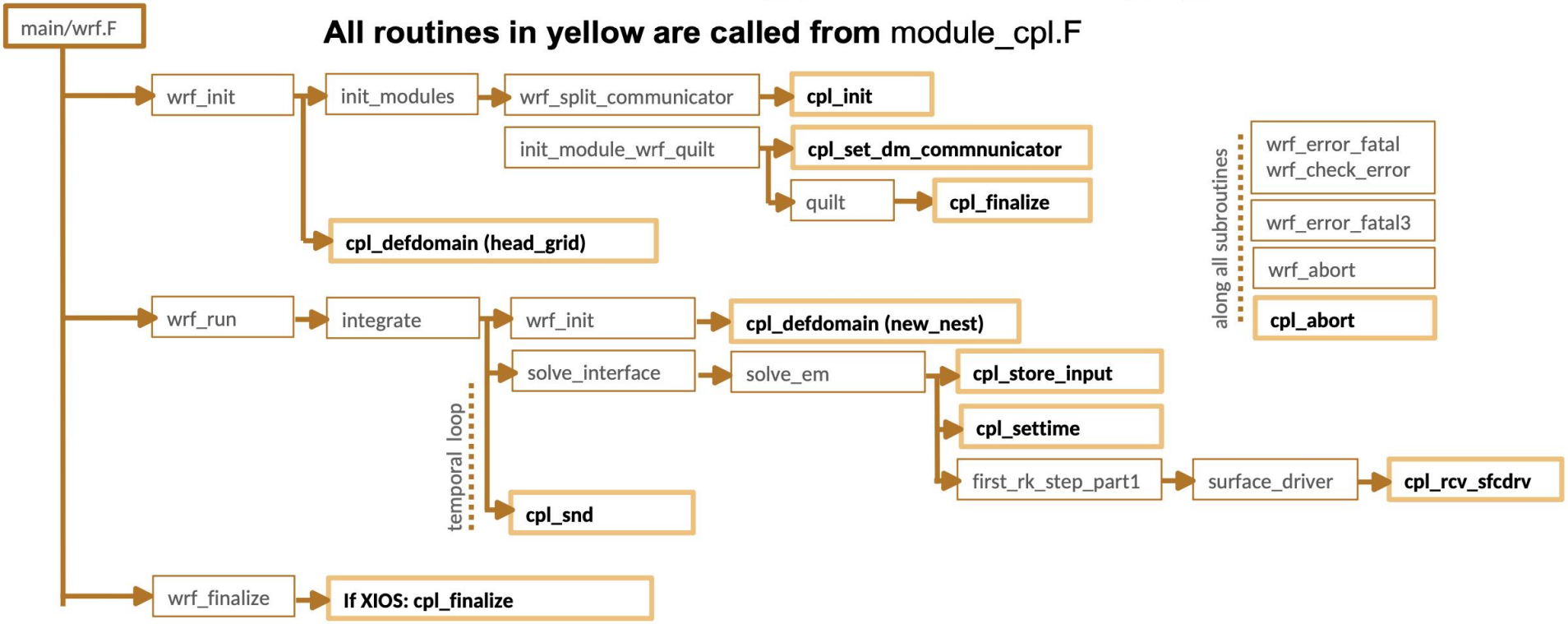




Coupling with OASIS: in WRF

2 new modules: frame/module_cpl.F **and** frame/module_cpl_oasis3.F

All routines in yellow are called from module_cpl.F



Coupling with OASIS: in WW3