
# OASIS3-MCT User Guide

*OASIS3-MCT_1.0*

*Edited by:*
*S. Valcke, T. Craig, L. Coquart*
*CERFACS/CNRS SUC URA No1875*

CERFACS TR/CMGC/12/49

July 2012

## Copyright Notice



## How to get assistance?

Assistance can be obtained as listed below.

## Phone Numbers and Electronic Mail Adresses


| Name | Phone | Affiliation | e-mail |
|---|---|---|---|
| Laure Coquart | | CNRS | oasishelp(at)cerfacs.fr |
| Sophie Valcke | | CERFACS | |


## How to get documentation ?

The documentation can be downloaded from the OASIS web site under the URL :
http://oasis.enes.org

# Contents

.

## ACKNOWLEDGMENTS

We would like to thank the main past or present developers of OASIS are (in alphabetical order, with the name of their institution at the time of their contribution to OASIS):

Arnaud Caubel (LSCE/IPSL & FECIT/Fujitsu), Damien Declat (CERFACS), Italo Epicoco (CMCC), Veronika Gayler (MPI-M&D), Josefine Ghattas (CERFACS), Jean Latour (CERFACS & Fujitsu-Fecit), Eric Maisonnave (CERFACS), Silvia Mocavero (CMCC), Elodie Rapaport (CERFACS), Hubert Ritzdorf (CCRLE-NEC), Sami Saarinen (ECMWF), Eric Sevault (Météo-France), Laurent Terray (CERFACS), Olivier Thual (CERFACS), Reiner Vogelsang (SGI Germany), Li Yan (CERFACS).

We also would like to thank the following people for their help and suggestions in the design of the OASIS software (in alphabetical order, with the name of their institution at the time of their contribution to OASIS):

Dominique Astruc (IMFT), Chandan Basu (NSC, Sweden), Sophie Belamari (Météo-France), Dominique Bielli (Météo-France), Gilles Bourhis (IDRIS), Pascale Braconnot (IPSL/LSCE), Sandro Calmanti (Météo-France), Christophe Cassou (CERFACS), Yves Chartier (RPN), Jalel Chergui (IDRIS), Philippe Courtier (Météo-France), Philippe Dandin (Météo-France), Michel Déqué (Météo-France), Ralph Doescher (SMHI), Jean-Louis Dufresne (LMD), Jean-Marie Epitalon (CERFACS), Laurent Fairhead (LMD), Uwe Fladrich (SMHI), Marie-Alice Foujols (IPSL), Gilles Garric (CERFACS), Eric Guilyardi (CERFACS), Charles Henriet (CRAY France), Pierre Herchuelz (ACCRI), Maurice Imbard (Météo-France), Luis Kornblueh (MPI-M), Stephanie Legutke (MPI-M&D), Claire Lévy (LODYC), Olivier Marti (IPSL/LSCE), Sébastien Masson (IPSL/LOCEAN) Claude Mercier (IDRIS), Pascale Noyret (EDF), Andrea Piacentini (CERFACS), Marc Pontaud (Météo-France), Adam Ralph (ICHEC), René Redler (MPI-M), Tim Stockdale (ECMWF), Rowan Sutton (UGAMP), Véronique Taverne (CERFACS), Jean-Christophe Thil (UKMO), Nils Wedi (ECMWF).

# Chapter 1

# Introduction

In 1991, CERFACS started the development of a software interface to couple existing numerical General Circulation Models of the ocean and of the atmosphere. Today, the OASIS3.3 coupler, which is the result of more than 20 years of evolution is used by about 30 modelling groups in Europe, Australia, Asia and North America on the different computing platforms. The list of coupled models realized with OASIS3 and OASIS2 can be found in tables C.1 and C.2 in Appendix C.

OASIS sustained development is ensured by a collaboration between CERFACS and the Centre National de la Recherche Scientifique (CNRS) and its maintainance and user support is presently reinforced with additinal resources coming from IS-ENES project funded by the EU (FP7 - GA no 228203).

The current OASIS3-MCT version was significantly refactored with respect to OASIS3.3. OASIS3-MCT is now interfaced with the Model Coupling Toolkit[1] (Larson et al 2005) (Jacob et al 2005) developed by the Argonne National Laboratory in the USA. MCT implements fully parallel regridding (as a parallel matrix vector multiplication) and parallel distributed exchanges of the coupling fields, based on pre-computed regridding weights and addresses. Its design philosophy, based on flexibility and minimal invasiveness, is close to the OASIS approach. MCT has proven parallel performance and is, most notably, the underlying coupling software used in National Center for Atmospheric Research Community Earth System Model 1 (NCAR CESM1).

OASIS3-MCT is a portable set of Fortran 77, Fortran 90 and C routines. Low-intrusiveness, portability and flexibility are OASIS3-MCT key design concepts. At run-time, there is no longer a separate coupler executable: OASIS3-MCT acts as a coupling library, which main function is to interpolate and exchange the coupling fields between the component models of a coupled system. OASIS3-MCT supports coupling of general two dimensional fields. Unstructured grids are also supported using a one dimension degeneration of the two dimensional structures. Thanks to MCT, all transformations, including regridding, are executed in parallel on the set of source or target component processes and all couplings are now executed in parallel directly between the components via Message Passing Interface (MPI). OASIS3-MCT also supports parallel file I/O using netcdf.

In spite of the significant changes in underlying implementation, usage of OASIS3-MCT in the codes has largely remained unchanged with respect to OASIS3.3. To communicate with another model, or to perform I/O actions, a component model needs to include few specific calls to the OASIS3-MCT coupling library, which Application Programmig Interface used in component models is unchanged. The use statement has been updated and now requires a single "use mod_prism" or "use mod_oasis" statement instead of the various use statements required in prior OASIS3 versions. The *namcouple* configuration file is also largely unchanged relative to OASIS3, although several options are either not used or not supported. There is a new transformation in *namcouple* i.e. `MAPPING` which allows a user to specify a mapping file generated externally. Some features like vector mapping and second order mapping have been delayed in implementation while other new features like parallel mapping have been added. And currently, only

[1] MCT, see www.mcs.anl.gov/research/projects/mct/

MPI1 job launching is supported.

First tests done with up to 8000 cores on the Bullx Curie machine at the TGCC are very encouraging and it is therefore very likely that OASIS3-MCT will provide an efficient and easy-to-use coupling solution.

## 1.1 Step-by-step use of OASIS3-MCT

To use OASIS3-MCT for coupling models (and/or perform I/O actions), one has to follow these steps:

1. Obtain OASIS3-MCT source code (see chapter 1.2).
2. Identify the coupling or I/O fields and adapt the component models to allow their exchange with the OASIS3-MCT coupling library based on MPI1 message passing. The OASIS3-MCT coupling library uses NetCDF and therefore can be used to perform I/O actions from/to disk files. For more detail on how to interface a model with OASIS3-MCT, see chapter 2.

   The tutorial coupled model gives a practical example of a coupled model; the sources are given in directories `examples/tutorial` ; more detail on the `tutorial` and how to compile and run it can be found in chapter 6.
3. Define all coupling and I/O parameters and the transformations required to adapt each coupling field from its source model grid to its target model grid; on this basis, prepare OASIS3-MCT configuring file *namcouple* (see chapter 3).

   OASIS3-MCT supports different interpolation algorithms as is described in chapter 4. Regridding files can be compute online using the SCRIP options or offline and read using the MAPPING transformation.
4. Generate required auxiliary data files (see chapter 5).
5. Compile OASIS3-MCT, the component models and start the coupled experiment. Chapter 6 describes how to compile and run OASIS3-MCT and the `tutorial` toy coupled model.

If you need extra help, do not hesitate to contact us (see contact details on the back of the cover page).

## 1.2 OASIS3-MCT sources

OASIS3-MCT and `tutorial` toy coupled model sources are available from CERFACS SVN server. To obtain more detail on how to download the sources, please fill in the registration form at https://verc.enes.org/oasis/download/oasis-registration-form .

OASIS3-MCT directory structure is the following one:

```
- oasis3-mct/lib/psmile          OASIS3-MCT coupling library
                /scrip           SCRIP interpolation library
                /mct             Model Coupling Toolkit Coupling Software

- oasis3-mct/doc                 OASIS3-MCT User Guide

- oasis3-mct/util/make_dir       Utilities to compile OASIS3-MCT

- oasis3-mct/examples/tutorial   Environment to run the tutorial toymodel
```

## 1.3 Licenses and Copyrights

### 1.3.1 OASIS3-MCT license and copyright statement

This software and ancillary information called OASIS3-MCT is free software. CERFACS has rights to use, reproduce, and distribute OASIS3-MCT. The public may copy, distribute, use, prepare derivative works and publicly display OASIS3-MCT under the terms of the Lesser GNU General Public License (LGPL) as published by the Free Software Foundation, provided that this notice and any statement of authorship are reproduced on all copies. If OASIS3-MCT is modified to produce derivative works, such modified software should be clearly marked, so as not to confuse it with the OASIS3-MCT version available from CERFACS.

The developers of the OASIS3-MCT software are researchers attempting to build a modular and user-friendly coupler accessible to the climate modelling community. Although we use the tool ourselves and have made every effort to ensure its accuracy, we can not make any guarantees. We provide the software to you for free. In return, you–the user–assume full responsibility for use of the software. The OASIS3-MCT software comes without any warranties (implied or expressed) and is not guaranteed to work for you or on your computer. Specifically, CERFACS and the various individuals involved in development and maintenance of the OASIS3-MCT software are not responsible for any damage that may result from correct or incorrect use of this software.

### 1.3.2 MCT copyright statement

Modeling Coupling Toolkit (MCT) Software

Copyright 2011, UChicago Argonne, LLC as Operator of Argonne National Laboratory. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: ”This product includes software developed by the UChicago Argonne, LLC, as Operator of Argonne National Laboratory.” Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

   This software was authored by:

   - Argonne National Laboratory Climate Modeling Group, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL 60439
   - Robert Jacob, tel: (630) 252-2983, E-mail: jacob@mcs.anl.gov
   - Jay Larson, E-mail: larson@mcs.anl.gov
   - Everest Ong
   - Ray Loy
4. WARRANTY DISCLAIMER. THE SOFTWARE IS SUPPLIED ”AS IS” WITHOUT WARRANTY OF ANY KIND. THE COPYRIGHT HOLDER, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF ENERGY, AND THEIR EMPLOYEES: (1) DISCLAIM ANY WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, (2) DO NOT ASSUME ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF THE SOFTWARE, (3) DO NOT REPRESENT THAT USE OF THE SOFTWARE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS, (4) DO NOT WARRANT THAT THE SOFTWARE WILL FUNCTION UNINTERRUPTED, THAT IT IS ERROR-FREE OR THAT ANY ERRORS WILL BE CORRECTED.

5. LIMITATION OF LIABILITY. IN NO EVENT WILL THE COPYRIGHT HOLDER, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF ENERGY, OR THEIR EMPLOYEES: BE LIABLE FOR ANY INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL OR PUNITIVE DAMAGES OF ANY KIND OR NATURE, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS OR LOSS OF DATA, FOR ANY REASON WHATSOEVER, WHETHER SUCH LIABILITY IS ASSERTED ON THE BASIS OF CONTRACT, TORT (INCLUDING NEGLIGENCE OR STRICT LIABILITY), OR OTHERWISE, EVEN IF ANY OF SAID PARTIES HAS BEEN WARNED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGES.

### 1.3.3 The SCRIP 1.4 license copyright statement

The SCRIP 1.4 copyright statement reads as follows:

"Copyright 1997, 1998 the Regents of the University of California. This software and ancillary information (herein called SOFTWARE) called SCRIP is made available under the terms described here. The SOFTWARE has been approved for release with associated LA-CC Number 98-45. Unless otherwise indicated, this SOFTWARE has been authored by an employee or employees of the University of California, operator of Los Alamos National Laboratory under Contract No. W-7405-ENG-36 with the United States Department of Energy. The United States Government has rights to use, reproduce, and distribute this SOFTWARE. The public may copy, distribute, prepare derivative works and publicly display this SOFTWARE without charge, provided that this Notice and any statement of authorship are reproduced on all copies. Neither the Government nor the University makes any warranty, express or implied, or assumes any liability or responsibility for the use of this SOFTWARE. If SOFTWARE is modified to produce derivative works, such modified SOFTWARE should be clearly marked, so as not to confuse it with the version available from Los Alamos National Laboratory."

# Chapter 2

# Interfacing a component model with OASIS3-MCT

At run-time, the OASIS3-MCT coupling layer supports coupling data between two components as well as interpolation and transformation of coupling fields. Different communication techniques have been historically developed in OASIS. With OASIS3-MCT, communication is performed by MCT based on Message Passing Interface 1 protocol (the keyword $CHANNEL in the configuration file *namcouple* has to be `MPI1`, see chapter 3).

For a practical test case using the OASIS3-MCT library, see the sources in `examples/tutorial` and more details in chapter 6.

To communicate with another component model or to perform I/O actions, a component model needs to be interfaced with the OASIS3-MCT library, which sources can be found in `oasis3-mct/lib/psmile` directory. The OASIS3-MCT library supports:

- parallel communication between parallel component models,
- an ability to couple a component on a subset of it's processeses only,
- automatic sending and receiving actions at appropriate times following user's choice indicated in the *namcouple*,
- time integration or accumulation of the coupling fields,
- some transformations such as mapping (interpolation) between grids,
- I/O actions from/to files.

To adapt a component model to OASIS3-MCT, specific calls of the following classes have to be implemented in the code:

1. Initialisation (section 2.2)
2. Grid data file definition (section 2.3)
3. Partition definition (section 2.4)
4. coupling-I/O field declaration (section 2.5)
5. End of definition phase (section 2.6)
6. coupling-I/O field sending and receiving (section 2.7)
7. Termination (section 2.8)
8. Optional auxiliary routines (section 2.9)

Finally, in section 2.10, different coupling algorithms are illustrated and details on how to reproduce them with OASIS3-MCT are provided. More information on the `LAG` and `SEQ` indices are also given in that section.

## 2.1 Use of OASIS3-MCT library

To use OASIS3-MCT library, a user needs to add in his code:

- `USE mod_oasis`

  ** OR **
- `USE mod_prism`

Both use statements are valid and use of just one or the other is recommended in a particular component model. A single use statement now provides all the methods that required multiple use statements in previous OASIS3 versions. The methods, datatypes, and capabilities are identical for both the `mod_prism` or `mod_oasis` interfaces. The only difference is the name of the interface. The interface in module `mod_prism` is provided for backwards compatability with prior versions of OASIS3. Use of module `mod_oasis` is now recommended provides access to a set of updated routine names that will continue to evolve in the future, always ensuring backward compatibility. In the following sections, both the `mod_prism` and `mod_oasis` interface names is defined and a single description of the interface arguments is provided.

## 2.2 Initialisation

### 2.2.1 Coupling initialisation

- `CALL oasis_init_comp (compid, model_name, ierror)`
- `CALL prism_init_comp_proto (compid, model_name, ierror)`
  - `compid [INTEGER; OUT]`: component model ID
  - `model_name [CHARACTER*6; IN]`: component model name (as in *namcouple*)
  - `ierror [INTEGER; OUT]`: returned error code.

  This routine must called by all component processes to initialise the coupling.[1]

### 2.2.2 Communicator for internal parallelisation

- `CALL oasis_get_localcomm (local_comm, ierror )`
- `CALL prism_get_localcomm_proto (local_comm, ierror )`
  - `local_comm [INTEGER; OUT]`: value of local communicator
  - `ierror [INTEGER; OUT]`: returned error code.

  If needed, this routine may be called by the component processes to get the value of a local communicator to be used by the component for its internal parallelisation.

  This may be needed as all component models started in a pseudo-MPMD mode with MPI1 share automatically the same MPI_COMM_WORLD communicator. Another communicator has to be used for the internal parallelisation of each component. OASIS3-MCT creates this local communicator based on the name given to `oasis_init_comp/prism_init_comp_proto` routine; its value is returned as the first argument of the routine, `local_comm`.

[1] The model may call MPI_Init explicitly, but if so, has to call it before calling `prism_init_comp_proto`; in this case, the model also has to call MPI_Finalize explicitly, but only after calling `prism_terminate_proto`.

### 2.2.3 Coupling through a subset of the component model processes

If only a subset of the component processes participate in the coupling (e.g. a component is setup to run on 80 processes but 16 of those processes are associated with a distinct task, like I/O), a communicator gathering only these processes must be defined, with either `oasis/prism_create_couplcomm` or `oasis/prism_set_couplcomm`.

If such communicator does not exist yet in the code, the component processes should use, to create it:

- `CALL oasis_create_couplcomm(icpl, local_comm, coupl_comm, kinfo)`
- `CALL prism_create_couplcomm(icpl, local_comm, coupl_comm, kinfo)`
  - `icpl [INTEGER; IN]`: coupling process flag
  - `local_comm [INTEGER; IN]`: MPI communicator with all processes of the component
  - `coupl_comm [INTEGER; OUT]`: returned MPI communicator gathering only component processes participating in the coupling
  - `kinfo [INTEGER; OUT; OPTIONAL]`: returned error code

  This routine creates a coupling communicator for a subset of processes. It must be called by all component processes with `icpl=1` for processes participating in the coupling and with `icpl=MPI_UNDEFINED` for the others. Argument `local_comm` is the MPI communicator associated with all processes of the component. The new coupling communicator is returned in `coupl_comm` argument and the internal coupling communicator is also set to that value.

If this communicator already exist in the code, the component should use, to provide it to OASIS3-MCT:

- `CALL oasis_set_couplcomm(coupl_comm, kinfo)`
- `CALL prism_set_couplcomm(coupl_comm, kinfo)`
  - `coupl_comm [INTEGER; IN]`: MPI communicator gathering only component processes participating in the coupling
  - `kinfo [INTEGER; OUT; OPTIONAL]`: returned error code

  This routine allows to provide a local coupling communicator to OASIS3-MCT, given that it already exists in the code. The value of `coupl_comm` must be the value of this local coupling communicator for the processes participating to the coupling and it must be `MPI_COMM_NULL` for processes not involved in the coupling.

These routines should be called after the `oasis_init_comp/prism_init_comp_proto` but before the grid, partition, or coupling field declaration. All OASIS3-MCT interface routines, besides the grid definition (see section 2.3) and the "puts" and " gets" per se (see section 2.7), are collective and must be called by all processes. In particular, the coupling fields sent or received by the component must be declared by all component processes with `oasis_def_var` / `prism_def_var_proto` (see section 2.5) and the field partition must be described across all coupling processes with `oasis_def_partition` / `prism_def_partition_proto` (but with `ig_paral(:)=0` for the processes not involved in the coupling, see section 2.4.)

Here is a coding sample of how to use these routines:

```
CALL oasis_init_comp (comp_id, comp_name, ierror )
CALL oasis_get_localcomm ( localComm, ierror )

!--- create communicator gathering coupling processes (every other)
CALL MPI_Comm_Rank ( localComm, mype, ierror )
couplingpe = .false.
if (mod(mype,2) == 0) couplingpe = .true.
icpl = MPI_UNDEFINED
if (couplingpe) icpl = 1
```

```
CALL MPI_COMM_Split(localComm,icpl,1,couplComm,ierror)
!
!--- provide this communicator to OASIS3-MCT
CALL oasis_set_couplcomm(couplComm, ierror)

! The call to MPI_COMM_Split and oasis_set_couplcomm could be replaced by
! CALL oasis_create_couplcomm(icpl,localComm,couplComm,ierror)

CALL oasis_def_partition ( ... )
CALL oasis_def_var ( ... )
CALL oasis_enddef ( ... )

!--- do loop
   ! ...
   if (couplingpe) CALL oasis_put( ... )
   ! ...
   if (couplingpe) CALL oasis_get( ... )
   ! ...
!--- enddo

CALL oasis_terminate ( ... )
```

### 2.2.4 Separate executable not coupling at all

For a model that is launched by oasis as a separate executable but that is NOT coupling at all, the only calls that should be made are :

```
CALL oasis_init_comp (comp_id, comp_name, ierror )
CALL oasis_get_localcomm ( localComm, ierror )   ! optional
CALL oasis_enddef ( ... )
CALL oasis_terminate ( ... )
```

## 2.3 Grid data file definition

With OASIS3-MCT, the grid data files *grids.nc, masks.nc* and *areas.nc* are required only for certain operations (see also section 5.1), i.e. *grids.nc*, and *masks.nc* for `SCRIPR` (see section 4.3) and *masks.nc* and *areas.nc* for `CONSERV` (see section 4.4). These grid data files can be created by the user before the run or can be written directly at run time by the **master process of each component model** with the following routines. These routines can be used by the component models to add grid fields to the grid files but grid fields are **never** overwritten in the grid files. These routines have to be called just after `prism_init_comp`.

- `CALL oasis_start_grids_writing (flag)` or
- `CALL prism_start_grids_writing (flag)`
  - `flag [INTEGER; OUT]`: not used

  Obsolete in OASIS3-MCT; exists however for upward compatibility.

- `CALL oasis_write_grid (cgrid, nx, ny, lon, lat)` or
- `CALL prism_write_grid (cgrid, nx, ny, lon, lat)`
  - `cgrid [CHARACTER*4; IN]`: grid name prefix (see 3.3)
  - `nx [INTEGER; IN]` : first grid dimension (x)

  - ny [INTEGER; IN] : second grid dimension (y)
  - lon [REAL, DIMENSION(nx,ny); IN) : array of longitudes (degrees East)
  - lat [REAL, DIMENSION(nx,ny); IN) : array of latitudes (degrees North)

  Writes the model grid longitudes and latitudes. Longitudes must be given in degrees East in the interval -360.0 to 720.0. Latitudes must be given in degrees North in the interval -90.0 to 90.0. Note that if some grid points overlap, it is recommended to define those points with the same number (e.g. 90.0 for both, not 450.0 for one and 90.0 for the other) to ensure automatic detection of overlap by OASIS3-MCT (which is essential to have a correct conservative remapping SCRIPR/CONSERV, see section 4.3).

- CALL oasis_write_corner (cgrid, nx, ny, nc, clon, clat) or
- CALL prism_write_corner (cgrid, nx, ny, nc, clon, clat)
  - cgrid [CHARACTER*4; IN]: grid name prefix
  - nx [INTEGER; IN] : first grid dimension (x)
  - ny [INTEGER; IN] : second grid dimension (y)
  - nc [INTEGER; IN] : number of corners per grid cell (always 4 in the version)
  - lon [REAL, DIMENSION (nx,ny,nc);IN] : array of corner longitudes (in degrees_East)
  - lat [REAL, DIMENSION (nx,ny,nc);IN] : array of corner latitudes (in degrees_North)

  Writes the grid cell corner longitudes and latitudes (counterclockwise sense). Longitudes must be given in degrees East in the interval -360.0 to 720.0. Latitudes must be given in degrees North in the interval -90.0 to 90.0. Note also that cells larger than 180.0 degrees in longitude are not supported. Writing of corners is optional as corner information is needed only for SCRIPR/CONSERV (see section 4.3). If called, needs to be called after oasis/prism_write_grid.

- CALL oasis_write_angle (cgrid, nx, ny, angle) or
- CALL prism_write_angle (cgrid, nx, ny, angle)
  - cgrid [CHARACTER*4; IN]: grid name prefix
  - nx [INTEGER; IN] : first grid dimension (x)
  - ny [INTEGER; IN] : second grid dimension (y)
  - angle [REAL, DIMENSION (nx,ny);IN] : array of angles

  Obsolete in OASIS3-MCT as vector interpolation is not supported. See SCRIPR/CONSERV in section 4.3.

- CALL oasis_write_mask (cgrid, nx, ny, mask) or
- CALL prism_write_mask (cgrid, nx, ny, mask)
  - cgrid [CHARACTER*4; IN]: grid name prefix
  - nx [INTEGER; IN] : first grid dimension (x)
  - ny [INTEGER; IN] : second grid dimension (y)
  - mask [INTEGER, DIMENSION(nx,ny) ;IN] : mask array (be careful about the OASIS historical convention (!): 0 = not masked, 1 = masked)

  Writes the model grid mask.

- CALL oasis_write_area (cgrid, nx, ny, area) or
- CALL prism_write_area (cgrid, nx, ny, area)
  - cgrid [CHARACTER*4; IN]: grid name prefix
  - nx [INTEGER; IN] : first grid dimension (x)

  - `ny [INTEGER; IN]` : second grid dimension (y)
  - `area [REAL, DIMENSION(nx,ny); IN]` : array of grid cell areas

  Writes of the model grid cell areas. Needed only for `CONSERV` operation (see section 4.4).

- `CALL prism_terminate_grids_writing()`
- `CALL oasis_terminate_grids_writing()`

  Terminates grids writing (required if some of the above grid writing routines are called).

## 2.4 Partition definition

The coupling fields sent or received by a component model are usually scattered among the different component processes. With OASIS3-MCT, all processes exchanging coupling data have to define their local partition in the global index space. If only a subset of the processes actually exchange coupling data, the processes not implied in the coupling have to call this routine to describe a null partition (i.e. with `ig_paral(:)=0`).

- `CALL oasis_def_partition (il_part_id, ig_paral, ierror)` or
- `CALL prism_def_partition_proto (il_part_id, ig_paral, ierror)`
  - `il_part_id [INTEGER; OUT]`: partition ID
  - `ig_paral [INTEGER, DIMENSION(:), IN]`: vector of integers describing the local partition in the global index space; has a different expression depending on the type of the partition; in OASIS3-MCT, 4 types of partition are supported: Serial (no partition), Apple, Box, and Orange (see below).
  - `ierror [INTEGER; OUT]`: returned error code.

### 2.4.1 Serial (no partition)

This is the choice for a monoprocess model. In this case, we have `ig_paral(1:3)`:

- `ig_paral(1)` = 0 (indicates a Serial "partition")
- `ig_paral(2)` = 0
- `ig_paral(3)` = the total grid size.

### 2.4.2 Apple partition

Each partition is a segment of the global domain, described by its global offset and its local size. In this case, we have `ig_paral(1:3)`:

- `ig_paral(1)` = 1 (indicates an Apple partition)
- `ig_paral(2)` = the segment global offset
- `ig_paral(3)` = the segment local size

Figure 2.1 illustrates an Apple partition over 3 processes.

### 2.4.3 Box partition

Each partition is a rectangular region of the global domain, described by the global offset of its upper left corner, and its local extents in the X and Y dimensions. The global extent in the X dimension must also be given. In this case, we have `ig_paral(1:5)`:

- `ig_paral(1)` = 2 (indicates a Box partition)

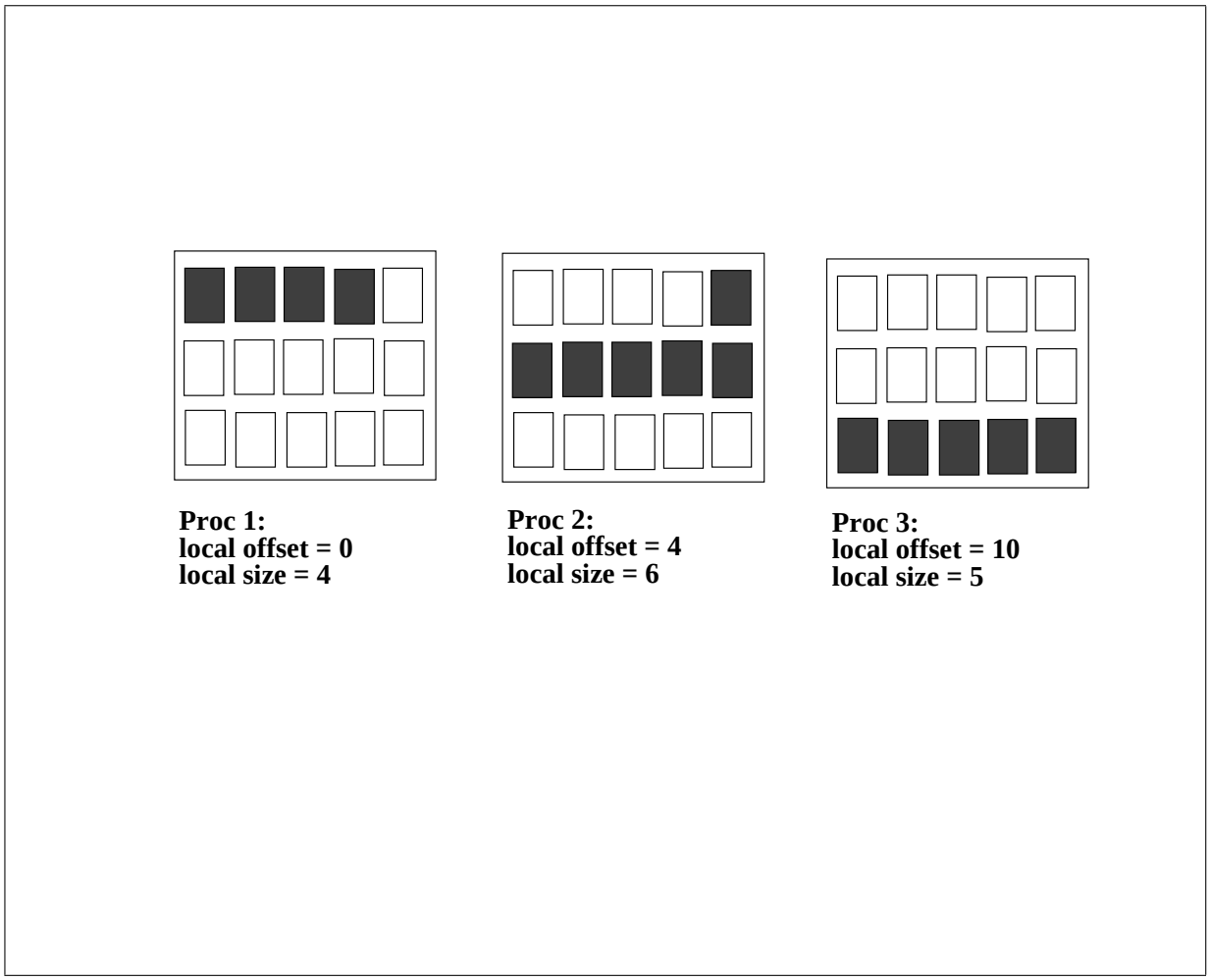



**Figure 2.1:** Apple partition. It is assumed here that the index start at 0 in the upper left corner.

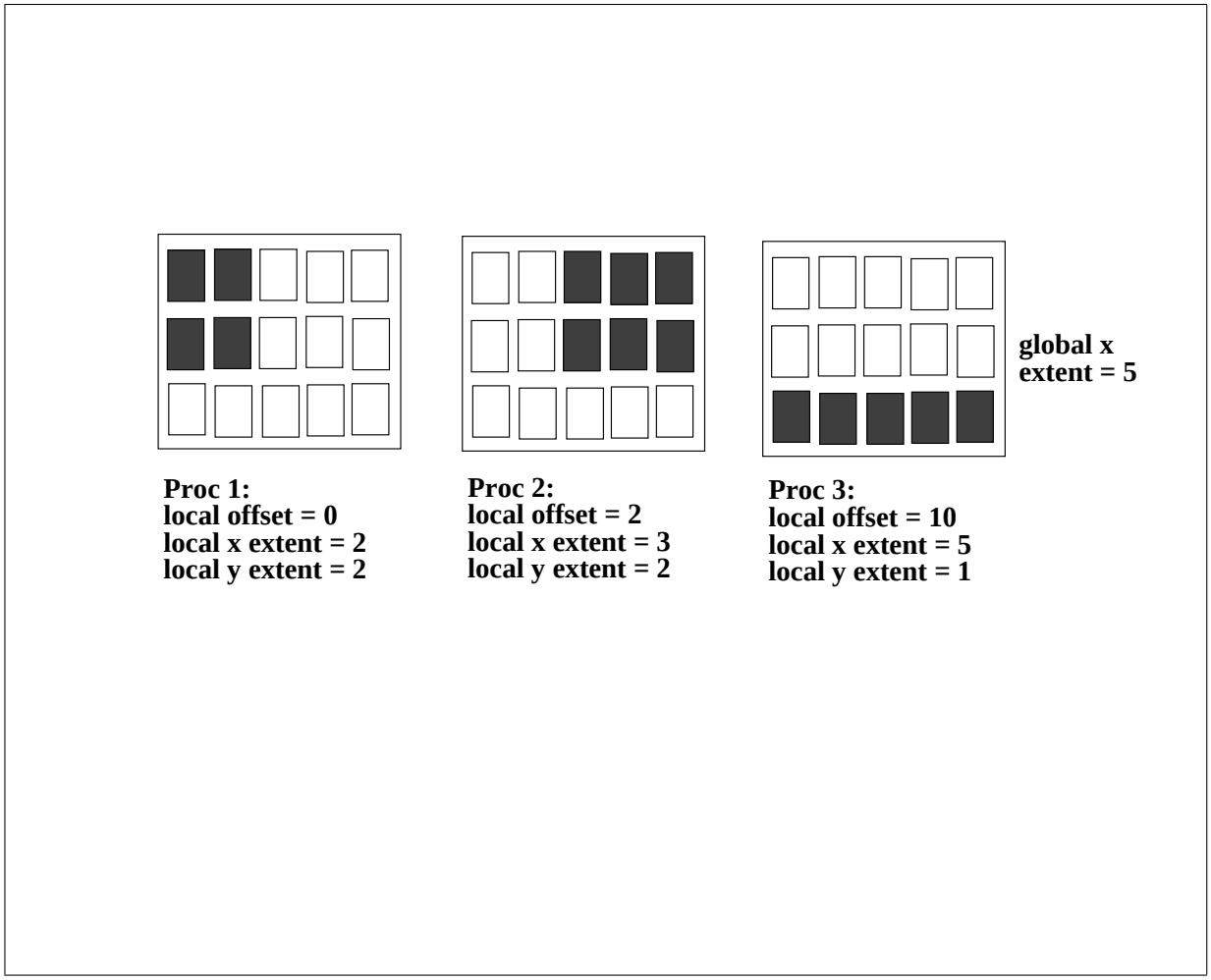



**Figure 2.2:** Box partition. It is assumed here that the index start at 0 in the upper left corner.

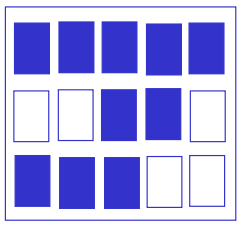
- `ig_paral(2)` = the upper left corner global offset
- `ig_paral(3)` = the local extent in x
- `ig_paral(4)` = the local extent in y
- `ig_paral(5)` = the global extent in x.

Figure 2.2 illustrates a Box partition over 3 processes.

### 2.4.4 Orange partition

Each partition is an ensemble of segments of the global domain. Each segment is described by its global offset and its local extent. In this case, we have `ig_paral(1:N)` where `N = 2 + 2*number of segments`

- `ig_paral(1)` = 3 (indicates a Orange partition)
- `ig_paral(2)` = the total number of segments for the partition (limited to 200 presently, see note for ig_paral(4) for Box partition above)
- `ig_paral(3)` = the first segment global offset
- `ig_paral(4)` = the first segment local extent
- `ig_paral(5)` = the second segment global offset
- `ig_paral(6)` = the second segment local extent
- ...



**Proc 1:**
**nbr of segments = 3**

**1st segment offset = 0**
**1st segment size = 5**
**2nd segment offset = 7**
**2nd segment size = 2**
**3rd segment offset = 10**
**3rd segment size = 3**

**Figure 2.3:** Orange partition for one process. It is assumed here that the index start at 0 in the upper left corner.

- `ig_paral(N-1)` = the last segment global offset
- `ig_paral(N)` = the last segment local extent

Figure 2.3 illustrates an Orange partition with 3 segments for one process. The other process partitions are not illustrated.

## 2.5 Coupling field declaration

All component processes declares the fields sent or received by the component during the simulation. This is true even for processes not implied in the coupling.

- `CALL oasis_def_var (var_id, name, il_part_id, var_nodims, kinout, var_actual_shape, var_type, ierror)` or
- `CALL prism_def_var_proto(var_id, name, il_part_id, var_nodims, kinout, var_actual_shape, var_type, ierror)`
  - `var_id [INTEGER; OUT]`: coupling field ID
  - `name [CHARACTER*8; IN]`: field symbolic name (as in the *namcouple*)
  - `il_part_id [INTEGER; IN]`: partition ID (see section 2.4)
  - `var_nodims [INTEGER, DIMENSION(2); IN]`: var_nodims(1) is the rank of field array (1 or 2); var_nodims(2) is the number of bundles (always 1 for OASIS3-MCT).

- kinout [INTEGER; IN]: OASIS_In or PRISM_In (i.e. = 15) for fields received by the model; OASIS_Out, PRISM_Out (i.e. = 14) for fields sent by the model [2].
- var_actual_shape [INTEGER, DIMENSION(2*var_nodims(1)); IN]: vector of integers giving the minimum and maximum index for each dimension of the coupling field array; for OASIS3, the minimum index has to be 1 and the maximum index has to be the extent of the dimension.
- var_type [INTEGER; IN]: type of coupling field array; put OASIS_Real or PRISM_Real (i.e. = 4) for single or double precision real arrays. All coupling data is treated as double precision in the coupling layer, but conversion to or from single precision data is supported in the interface.
- ierror [INTEGER; OUT]: returned error code.

## 2.6 End of definition phase

All component processes close the definition phase.

- CALL oasis_enddef (ierror)
- CALL prism_enddef_proto(ierror)
  - ierror [INTEGER; OUT]: returned error code.

## 2.7 Sending "put" and receiving "get" actions

### 2.7.1 Sending a coupling (or I/O) field

In the model time step loop, each process sends its part of the coupling (or I/O) field.

- CALL oasis_put (var_id, date, field_array, info)
- CALL prism_put_proto(var_id, date, field_array, info)
  - var_id [INTEGER; IN]: field ID (from corresponding oasis_def_var / prism_def_var_proto, see section 2.5)
  - date [INTEGER; IN]: number of seconds in the run at the time of the call (by convention at the beginning of the timestep)
  - field_array [REAL, IN]: coupling (or I/O) field array
  - info [INTEGER; OUT]: returned info code:
    - OASIS_Sent(=4) if the field was sent to another model
    - OASIS_LocTrans (=5) if the field was only used in a time transformation (not sent, not output)
    - OASIS_ToRest (=6) if the field was written to a restart file only
    - OASIS_Output (=7) if the field was written to an output file only
    - OASIS_SentOut (=8) if the field was both written to an output file and sent to another model (directly or via OASIS3 main process)
    - OASIS_ToRestOut (=9) if the field was written both to a restart file and to an output file.
    - OASIS_Ok (=0) otherwise and no error occurred.

To ensure a proper use of the oasis_put / prism_put_proto, one has to take care of the following aspects:

[2] Parameters OASIS_In, PRISM_In, OASIS_Out, PRISM_Out are defined in oasis3-mct/lib/psmile/src/mod_oasis_parameters.F90

- This routine may be called by the model at each timestep. The convention, for the `date` argument is to indicate the time at the beginning of the timestep. The sending is actually performed only if the time obtained by adding the field lag (`LAG` in the *namcouple*) to the `date` corresponds to a time at which it should be activated, given the coupling or I/O period indicated by the user in the *namcouple* (see section 3).
- If the convention for `date` is followed, the first coupling of a run should occur at time=0 and the final coupling should occur at time = runtime - cpl_period, where runtime is the total time of the run and cpl_period is the coupling period.
- The total run time should match an integer number of coupling periods.
- If a coupling field has a positive lag, the coupling field that matches the `oasis_get` / `prism_get_proto` at time=0 will come from a coupling restart file written by the last active `oasis_put` / `prism_put_proto` of the previous run (see section 2.10). For a coupling field with a positive lag, the coupling restart file (see section 5.2) is automatically overwritten by the `oasis_put` / `prism_put_proto` when the time (i.e. `date+LAG`) equals to the total runtime.
- If a local time transformation is indicated for the field by the user in the *namcouple* (INSTANT, AVERAGE, ACCUMUL, T_MIN or T_MAX, see section 4), it is automatically performed and the resulting field is finally sent at the coupling or I/O frequency. For non-instant transformations, partially transformed fields will be written to the restart file at the end of the run for use on the next model startup; this is a bug fix new in OASIS3-MCT.

### 2.7.2 Receiving a coupling (or I/O) field

In the model time stepping loop, each process receives its part of the coupling field.

- `CALL oasis_get (var_id, date, field_array, info)`
- `CALL prism_get_proto(var_id, date, field_array, info)`
  - `var_id [INTEGER; IN]`: field ID (from corresponding prism_def_var_proto)
  - `date [INTEGER; IN]`: number of seconds in the run at the time of the call (by convention at the beginning of the timestep)
  - `field_array [REAL, OUT]`: I/O or coupling field array
  - `info [INTEGER; OUT]`: returned info code:
    - OASIS_Recvd(=3) if the field was received from another model
    - OASIS_FromRest (=10) if the field was read from a restart file only
    - OASIS_Input (=11) if the field was read from an input file only
    - OASIS_RecvOut (=12) if the field was both received from another model and written to an output file
    - OASIS_FromRestOut (=13) if the field was both read from a restart file and written to an output file
    - OASIS_Ok (=0) otherwise and no error occurred.

This routine may be called by the model at each timestep. The `date` argument is automatically analysed and the receiving action is actually performed only if `date` corresponds to a time for which it should be activated, given the period indicated by the user in the *namcouple*. An exchange at the beginning of the run at time=0 is expected.

## 2.8 Termination

- `CALL oasis_terminate (ierror)`
- `CALL prism_terminate_proto(ierror)`

  - ierror [INTEGER; OUT]: returned error code.

  All processes of the component model must terminate the coupling by calling this routine[3] (normal termination).

## 2.9 Auxiliary routines

Not all auxiliary routines that were in OASIS3.3 are currently available.

- CALL oasis_abort (compid, routine_name, abort_message)
- CALL prism_abort_proto(compid, routine_name, abort_message)
  - compid [INTEGER; IN]: component model ID (from oasis_init_comp or prism_init_comp_proto)
  - routine_name[CHARACTER*; IN]: name of calling routine
  - abort_message[CHARACTER*; IN]: message to be written out.

  If a process needs to abort voluntarily, it should do so by calling oasis_abort/prism_abort_proto. This will ensure a proper termination of all processes in the coupled model communicator. This routine writes the name of the calling model, the name of the calling routine, and the message to the process debug file (see $NLOGPRT in section 3.2). This routine cannot be called before prism_init_comp_proto.

- CALL oasis_get_debug(debug_value)
- CALL prism_get_debug(debug_value)
  - debug_value [INTEGER; OUT]: debug value

  This routine may be called at any time to retrieve the current OASIS3-MCT internal debug level (see $NLOGPRT in section 3.2). This is useful if the user wants to return the original debug value after changing it or if a user wants to key off the oasis debug level for model debug diagnostics.

- CALL oasis_set_debug(debug_value)
- CALL prism_set_debug(debug_value)
  - debug_value [INTEGER; IN]: debug value

  This routine may be called at any time to change the debug level in oasis. This method allows users to vary the debug level at different points in the model integration.

- CALL oasis_get_intercomm(new_comm, cdnam, kinfo)
- CALL prism_get_intercomm(new_comm, cdnam, kinfo)
  - new_comm [INTEGER; OUT]: mpi intercomm communicator
  - cdnam [CHARACTER*; IN]: other model name
  - kinfo [INTEGER; OUT; OPTIONAL]: returned error code

  This routine sets up an MPI intercomm communicator between the root processors of two components, the local component and the component associated with cdnam. This must be called by both components at the same time otherwise a deadlock will occur. In addition, this call is collective across the tasks of the two components but other components are not involved.

- CALL oasis_get_intracomm(new_comm, cdnam, kinfo)
- CALL prism_get_intracomm(new_comm, cdnam, kinfo)
  - new_comm [INTEGER; OUT]: mpi intracomm communicator
  - cdnam [CHARACTER*; IN]: other model name

[3] If the process called MPI_Init (before calling oasis_init_comp or prism_init_comp_proto), it must also call MPI_Finalize explicitly, but only after calling oasis_terminate_proto or prism_terminate_proto.

- `kinfo [INTEGER; OUT; OPTIONAL]`: returned error code

This routine sets up an MPI intracomm communicator between the root processors of two components, the local component and the component associated with cdnam. This must be called by both components at the same time otherwise a deadlock will occur. In addition, this call is collective across the tasks of the two components but other components are not involved.

## 2.10 Coupling algorithms - SEQ and LAG concepts

Using the OASIS3-MCT coupling library, the user has full flexibility to reproduce different coupling algorithms. In the components, the sending and receiving routines, respectively `oasis_put/prism_put_proto` and `oasis_get/prism_get_proto`, can be called at each model timestep, with the appropriate `date` argument giving the actual time (at the beginning of the timestep), expressed in "number of seconds since the start of the run" (see section 2.7.1). This `date` argument is automatically analysed by the coupling library and depending on the coupling period and the lag (LAG) chosen by the user for each coupling field in the configuration file *namcouple*, different coupling algorithms can be reproduced without modifying anything in the component model codes themselves.

With OASIS3-MCT, the SEQ index is no longer needed in the *namcouple* input to specify the sequencing order of different coupling fields. The sequence (SEQ) index in the *namcouple* file now provides the coupling layer with an ability to detect a deadlock before it happens and exit.

The lag concept and indices are explained in more detail below and some examples are provided.

### 2.10.1 The lag concept

The lag (LAG) value tells the coupler to modify the time at which that data is sent (put) by the amount of the lag. The lag must be expressed in "number of seconds" and can be positive or negative but should never be larger (in absolute magnitude) than the coupling period of any field due to problems with restart-ability and dead-locking. When a component model calls a `oasis_put/prism_put_proto`, the value of the lag is automatically added to the value of the `date` argument and the "put" is actually performed when the sum `date+lag` is a coupling time; in the target component, this "put" will match a `oasis_get/prism_get_proto` for which the `date` argument is the same coupling time. The lag only shifts the time data is sent. It cannot be used to shift the time data is received yet.

When the lag is positive, a restart file must be available to initiate the coupling and in those cases, the restart file is then updated at the end of the run. A positive lag acts like a send occurred before the model started. In fact, for a field with positive lag, the source component model automatically reads the field in the restart file during the coupling initialization phase (below the `oasis_enddef/prism_enddef_proto`) and send the data to match the `oasis_get/prism_get_proto` performed at time=0 in the target component model. The final coupling data on the source side will then be automatically written to the restart file for use in the next run.

When there is a lag, the first and last instance of the source field in the debug netCDF file (EXPOUT fields, see section 3.3) always correspond respectively to the field read from and written to the restart file.

1. LAG concept first example

   A first coupling algorithm, exploiting the LAG concept, is illustrated on figure 2.4.

   On the 4 figures in this section, short black arrows correspond to `oasis_put/prism_put_proto` or `oasis_get/prism_get_proto` called in the component model that do not lead to any "put" or receiving action; long black arrows correspond to `oasis_put/prism_put_proto` or `oasis_get/prism_get_proto` called in the component models that do actually lead to a "put" or "get" action; long red arrows correspond to `oasis_put/prism_put_proto` or `oasis_get/prism_get_proto` called in the component models that lead to a reading or writing of the coupling field from or to a coupling restart file.

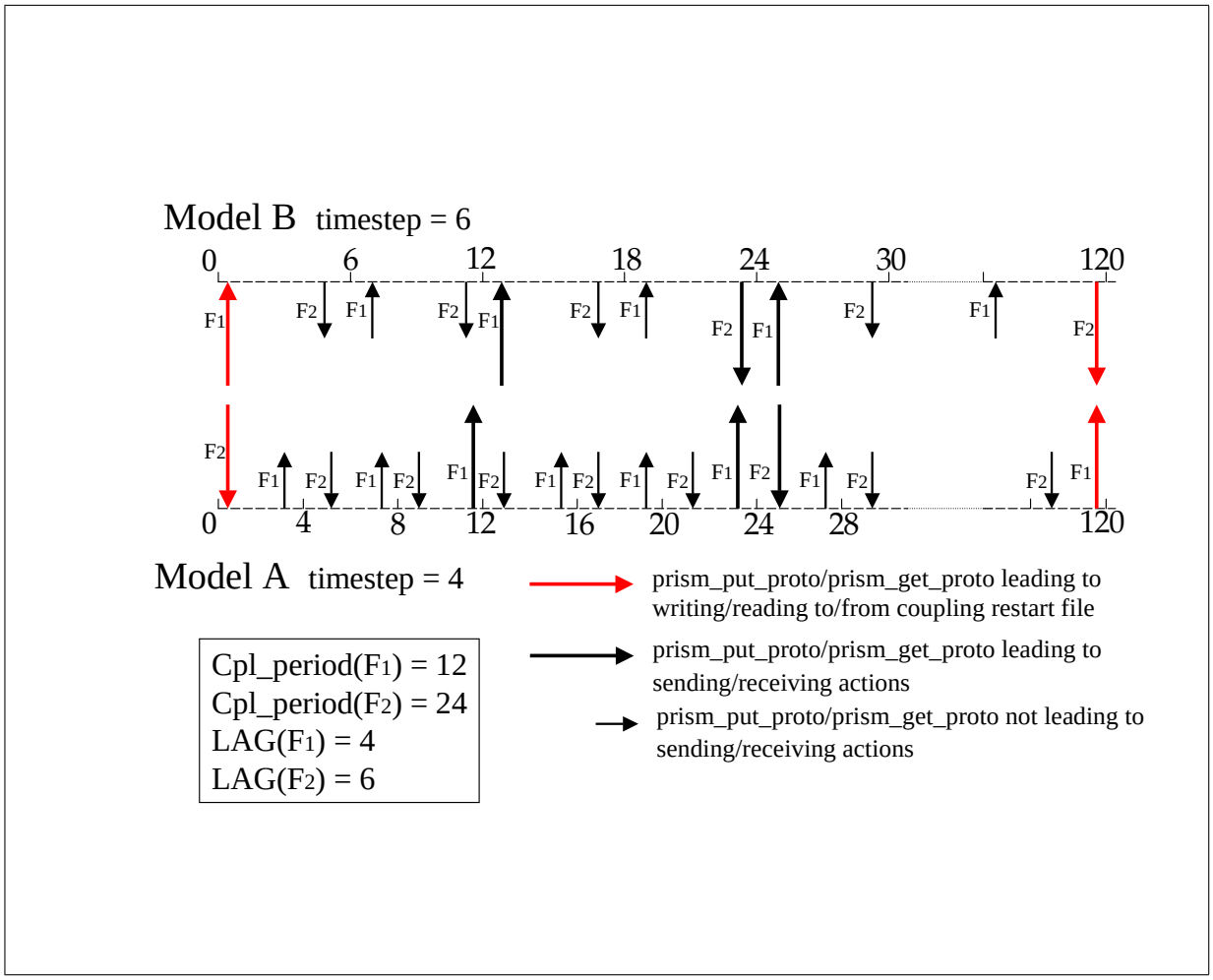



**Figure 2.4:** LAG concept first example

During a coupling timestep, model A receives $F_2$ and then sends $F_1$; its timestep length is 4. During a coupling timestep, model B receives $F_1$ and then sends $F_2$; its timestep length is 6. $F_1$ and $F_2$ coupling periods are respectively 12 and 24. If $F_1$/$F_2$ "put" action by model A/B was used at a coupling timestep to match the model B/A "get" action, a deadlock would occur as both models would be initially waiting on a "get" action. To prevent this, $F_1$ and $F_2$ produced at the timestep before have to be used to match respectively the model B and model A "get" actions.

This implies that a lag of respectively 4 and 6 seconds must be defined for $F_1$ and $F_2$. For $F_1$, the `oasis_put/prism_put_proto` performed at time 8 and 20 by model A will then lead to "put" actions (as 8 + 4 = 12 and 20 + 4 = 24 which are coupling periods) that match the "get" actions performed at times 12 and 24 below the `oasis_get/prism_get_proto` called by model B. For $F_2$, the `oasis_put/prism_put_proto` performed at time 18 by model B then leads to a "put" action (as 18 + 6 = 24 which is a coupling period) that matches the "get" action performed at time 24 below the `oasis_get/prism_get_proto` called by model A.

At the beginning of the run, as their LAG index is greater than 0, the first `oasis_get/prism_get_proto` of $F_1$ and $F_2$ will automatically be fulfilled with fields read from their respective coupling restart files. The user therefore has to create those coupling restart files before the first run in the experiment. At the end of the run, $F_1$ having a lag greater than 0, is automatically written to its coupling restart file below the last $F_1$ `oasis_put/prism_put_proto` as the `date+lag` equals the total run time. The analogue is true for $F_2$. These values will automatically be read in at the beginning of the next run below the respective `oasis_get/prism_get_proto`.

2. LAG concept second example

   A second coupling algorithm exploiting the LAG concept is illustrated on figure 2.5. During its timestep, model A receives $F_2$, sends $F_3$ and then $F_1$; its timestep length is 6. During its timestep, model B receives $F_1$, receives $F_3$ and then sends $F_2$; its timestep length is also 6. $F_1$, $F_2$ and $F_3$ coupling periods are both supposed to be equal to 12.

   For $F_1$ and $F_2$ the situation is similar to the first example. If $F_1$/$F_2$ "put" action by model A/B was used at a coupling timestep to match the model B/A "get" action, a deadlock would occur as both models would be waiting on a "get" action. To prevent this, $F_1$ and $F_2$ produced at the timestep before have to be used to match the model A and model B "get" actions, which means that a lag of 6 must be defined for both $F_1$ and $F_2$. For both coupling fields, the `oasis_put/prism_put_proto` performed at times 6 and 18 by the source model then lead to "put" actions (as 6 + 6 = 12 and 18 + 6 = 24 which are coupling periods) that match the "get" action performed at time 12 and 24 below the `oasis_get/prism_get_proto` called by the target model.

   For $F_3$, sent by model A and received by model B, no lag needs to be defined: the coupling field produced by model A at the coupling timestep can be "consumed" by model B without causing a deadlock situation.

   As in the first example, the `oasis_get/prism_get_proto` performed at the beginning of the run for $F_1$ and $F_2$, will automatically receive data read from their coupling restart files, and the last `oasis_put/prism_put_proto` performed at the end of the run automatically write them to their coupling restart file. For $F_3$, no coupling restart file is needed nor used as at each coupling period the coupling field produced by model A can be directly "consumed" by model B.

   We see here how the introduction of appropriate LAG indices results in "get" in the target model, coupling fields produced by the source model the timestep before; this is, in some coupling configurations, essential to avoid deadlock situations.

### 2.10.2 The sequence concept

The order of coupling operations in the system is determined solely by the order of calls to send (put) and receive (get) data in the models in conjunction with the setting of the lag in the *namcouple*. Data that is received (get) is always blocking while data that is sent (put) is non-blocking with respect to the model

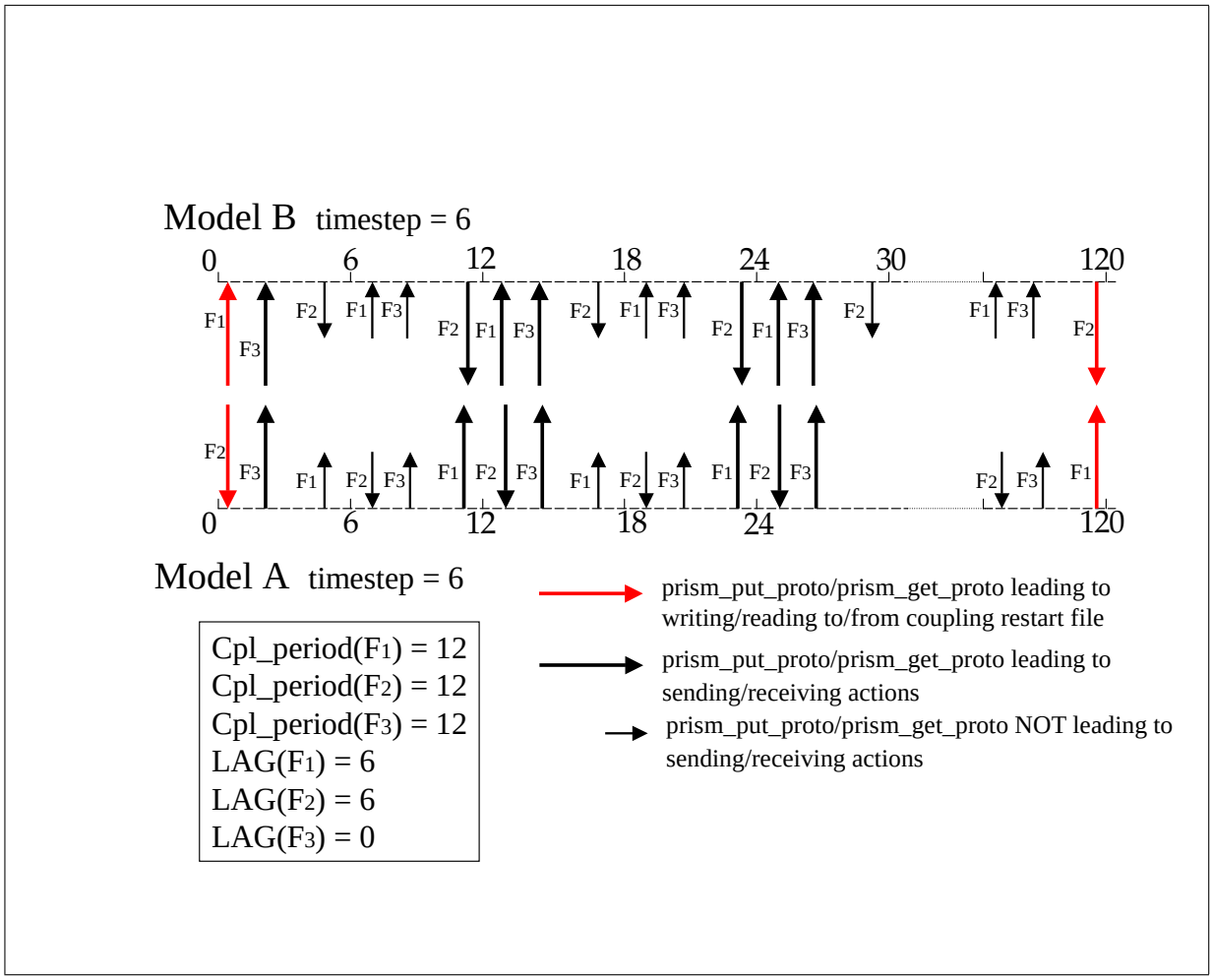



**Figure 2.5:** LAG concept second example

making that call. It is possible to deadlock the system if the relative orders of puts and gets in different models are not compatible.

With OASIS3-MCT, the sequence (SEQ) index in the *namcouple* file now provides the coupling layer with an ability to detect a deadlock before it happens and exit. It does this by tracking the order of get and put calls in models compared to the SEQ specified in the *namcouple*. If there are any inconsistencies, the model will abort gracefully with a useable error message before the system deadlocks. If there are any coupling dependencies in the system, use of the SEQ index is recommended for diagnosis but has no impact on the ultimate solution and is NOT required.

Take the following two examples. In both examples, there are two models, each "put" a field to the other at every coupling period without any lags. In the first case, there is no dependency and each model sends (puts) the data first and then receives the data second at each timestep.

```
 model1         model2
 ------         ------
put(fld1)      put(fld2)
get(fld2)      get(fld1)
```

The put from model1 for fld1 is received by the get in model2 and the put from model2 for fld2 is received by the get in model1. In this case, there is no sequencing dependency and the value of SEQ must be identical (or unset) in the *namcouple* description of the fld1 and fld2 coupling. If SEQ is set to 1 for fld1 and 2 for fld2 in this case, then the model will abort because at runtime, the coupling will detect, in model 2, that fld2 was sent before fld1 was received which is out of sequence as defined by the SEQ settings.

In the next example, there is a dependency in the sequencing.

```
 model1         model2
 ------         ------
put(fld1)      get(fld1)
               fld2=g(fld1)
get(fld2)      put(fld2)
```

In model2, fld2 depends on fld1. If this dependency is known, then there is a benefit in using SEQ=1 for fld1 and SEQ=2 for fld2. At runtime, if the sequencing of both model1 and model2 do not match the above diagram, the model will abort gracefully. For instance, if model2 has the dependency shown above but model1 does not have consistent ordering of the put and get as required by model2, then if SEQ is unused, the model will deadlock and hang. If SEQ is set properly, the coupling layer will detect that the required sequence has not been followed and will exit gracefully with an error message.

Again, the SEQ namecouple setting is only diagnostic and is not required.

# Chapter 3

# The configuration file *namcouple*

The OASIS3-MCT configuration file *namcouple* contains, below pre-defined keywords, all user's defined information necessary to configure a particular coupled run.

The *namcouple* is a text file with the following characteristics:

- the keywords used to separate the information can appear in any order;
- the number of blanks between two character strings is non-significant;
- all lines beginning with # are ignored and considered as comments.
- **blank lines are not allowed.**

The first part of *namcouple* is devoted to configuration of general parameters such as the number of models involved in the simulation or the number of fields. The second part gathers specific information on each coupling (or I/O) field, e.g. their coupling period, the list of transformations or interpolations to be performed by OASIS3-MCT and associated configuring lines (described in more details in chapter 4), etc.

In OASIS3-MCT, several *namcouple* inputs have been deprecated but, for backwards compatibility, they are still allowed. These inputs will be noted in the following text using the notation "UNUSED" and not fully described. Information below these keywords is obsolete in OASIS3-MCT; it will not be read and will not be used.

In the next sections, a simple *namcouple* example is given and all configuring parameters are described. The additional lines containing the different parameters required for each transformation are described in section 4. A realistic *namcouple* can be found in `oasis3-mct/examples/tutorial/data_oasis3/` directory.

## 3.1 An example of a simple *namcouple*

The following simple *namcouple* configures a run into which an ocean, an atmosphere and an atmospheric chemistry models are coupled. The ocean provides only the SOSSTSST field to the atmosphere, which in return provides the field CONSFTOT to the ocean. One field (COSENHFL) is exchanged from the atmosphere to the atmospheric chemistry, and one field (SOALBEDO) is read from a file by the ocean.

```
#############################################################
# First section
#
 $SEQMODE
#
 $CHANNEL
#
```

```
 $NFIELDS
    4
#
 $JOBNAME
#
 $NBMODEL
    3  ocemod   atmmod  chemod  55  70   99
#
 $RUNTIME
    432000
#
 $INIDATE
#
 $MODINFO
#
 $NLOGPRT
   2
#
 $CALTYPE
#
#############################################################################
# Second section
 $STRINGS
#
# Field 1
 SOSSTSST SISUTESU 1 86400  5  sstoc.nc  EXPORTED
 182  149  128  64  toce  atmo   LAG=+14400  SEQ=+1
 P 2 P 0
 LOCTRANS CHECKIN MAPPING  BLASNEW CHECKOUT
#
  AVERAGE
  INT=1
  map_toce_atmo_120315.nc src opt
  CONSTANT     273.15
  INT=1
#
# Field 2
 CONSFTOT SOHEFLDO 6 86400  4   flxat.nc  EXPORTED
 atmo   toce  LAG=+14400  SEQ=+2
 P 0 P 2
 LOCTRANS  CHECKIN  SCRIPR CHECKOUT
#
  ACCUMUL
  INT=1
  BILINEAR LR SCALAR LATLON 1
  INT=1
#
# Field 3
 COSENHFL  SOSENHFL  37  86400   1  flda3.nc  IGNOUT
 atmo   atmo LAG=+7200
 LOCTRANS
```

```
 AVERAGE
#
# Field 4
 SOALBEDO SOALBEDO  17  86400  0  SOALBEDO.nc  INPUT
#############################################################################
```

## 3.2 First section of *namcouple* file

The first section of *namcouple* uses some predefined keywords prefixed by the $ sign to locate the related information. The $ sign must be in the second column. The first ten keywords are described hereafter:

- `$SEQMODE`: UNUSED
- `$CHANNEL`: UNUSED
- `$NFIELDS`: On the line below this keyword is the total number of field entries in the second part of the *namcouple*. If more than one field are described on the same line (new in OASIS3-MCT, see section B) this counts as only one entry.
- `$JOBNAME`: UNUSED
- `$NBMODEL`: On the line below this keyword is the number of models running in the given experiment followed by `CHARACTER*6` variables giving their names, which must correspond to the name announced by each model when calling `oasis_init_comp` or `prism_init_comp_proto` (second argument, see section 2.2).

  Then the user may indicate on the same line the maximum Fortran unit number used by the models. In the example, Fortran units above 55, 70, and 99 are free for respectively the ocean, atmosphere, and atmospheric chemistry models. **In all cases, OASIS3-MCT library assumes, during the initialization phase, that units 1025 and 1026 are free and temporarily uses these units to read the *namcouple* and to write corresponding log messages to file `nout.000000`.** After the initialization phase, OASIS3-MCT will still suppose that units above 1024 are free, unless maximum unit numbers are indicated here in the *namcouple*.
- `$RUNTIME`: On the line below this keyword is the total simulated time of the run, expressed in seconds.
- `$INIDATE`: UNUSED
- `$MODINFO`: UNUSED
- `$NLOGPRT`: The line below this keyword refers to the amount of information that will be written to the OASIS3-MCT debug files for each model and processor. The value of `$NLOGPRT` may be:
  - 0 : one file debug.root.xx is open by the master proces of each model and one file debug_notroot.xx is open for all the other processes of each model to write only error information.
  - 1 : one file debug.root.xx is open by the master proces of each model to write information equivalent to level 10 (see below); one file debug_notroot.xx is open for all the other processes of each model to write error information.
  - 2 : one file debug.xx.xxxxxx is open by each process of each model to write normal production diagnostics
  - 5 : as for 2 with in addition some initial debug info
  - 10: as for 5 with in addition the routine calling tree
  - 12: as for 10 with in addition some routine calling notes
  - 15: as for 12 with even more debug diagnostics
  - 20: as for 15 with in addition some extra runtime analysis
  - 30: full debug information

This value can be changed at runtime with the `oasis_set_debug` or `prism_set_debug` routine (see section 2.9).

- `$CALTYPE`: UNUSED

## 3.3 Second section of *namcouple* file

The second part of the *namcouple*, starting after the keyword `$STRINGS`, contains coupling information for each coupling (or I/O) field. Its format depends on the field status given by the last entry on the field first line (`EXPORTED`, `IGNOUT` or `INPUT` in the example above). The field may be :

- `AUXILARY`: UNUSED
- `EXPORTED`: exchanged between component models and transformed by OASIS3-MCT
- `EXPOUT`: exchanged, transformed and also written to two debug NetCDF files, one before the sending action in the source model below the `oasis_put` or `prism_put_proto` call (after local transformations `LOCTRANS` and `BLASOLD` if present), and one after the receiving action in the target model below the `prism_get_proto` call (after all transformations). This status should be used when debugging the coupled model only. The name of the debug NetCDF file (one per field) is automatically defined based on the field and component model names.
- `IGNORED`: with OASIS3-MCT, this setting is equivalent to and converted to EXPORTED
- `IGNOUT`: with OASIS3-MCT, this setting is equivalent to and converted to EXPOUT
- `INPUT`: read in from the input file by the target model below the `oasis_get` or `prism_get_proto` call at appropriate times corresponding to the input period indicated by the user in the *namcouple*. See section 5.3 for the format of the input file.
- `OUTPUT`: written out to an output debug NetCDF file by the source model below the `oasis_put` or `prism_put_proto` call, after local transformations `LOCTRANS` and `BLASOLD`, at appropriate times corresponding to the output period indicated by the user in the *namcouple*.

### 3.3.1 Second section of *namcouple* for `EXPORTED` and `EXPOUT` fields

The first 3 lines for fields with status `EXPORTED` and `EXPOUT` are as follows:

```
SOSSTSST SISUTESU 1 86400  5  sstoc.nc  EXPORTED
182  149    128  64  toce  atmo   LAG=+14400 SEQ=+1
P 2 P 0
```

where the different entries are:

- Field first line:
  - `SOSSTSST` : symbolic name for the field in the source model (`CHARACTER*8`). It has to match the argument `name` of the corresponding field declaration in the source model; see `oasis_def_var` or `prism_def_var_proto` in section 2.5.
  - `SISUTESU` : symbolic name for the field in the target model (`CHARACTER*8`). It has to match the argument `name` of the corresponding field declaration in the target model; see `oasis_def_var` or `prism_def_var_proto` in section 2.5.
  - 1 : UNUSED but still required for parsing
  - 86400 : coupling and/or I/O period for the field, in seconds.
  - 5 : number of transformations to be performed by OASIS3 on this field.
  - sstoc.nc : name of the coupling restart file for the field (`CHARACTER*8`); mandatory even if no coupling restart file is effectively used. (for more detail, see section 5.2);
  - `EXPORTED` : field status.

- Field second line:
  - 182 : number of points for the source grid first dimension (optional)[1].
  - 149 : number of points for the source grid second dimension (optional)[1].
  - 128 : number of points for the target grid first dimension (optional)[1].
  - 64 : number of points for the target grid second dimension (optional)[1].
  - toce : prefix of the source grid name in grid data files (see section 5.1) (`CHARACTER*4`)
  - atmo : prefix of the target grid name in grid data files (`CHARACTER*4`)
  - `LAG=+14400`: optional lag index for the field expressed in seconds
  - `SEQ=+1`: optional sequence index for the field (see section 2.10)
- Field third line
  - P : source grid first dimension characteristic ('P': periodical; 'R': regional).
  - 2 : source grid first dimension number of overlapping grid points.
  - P : target grid first dimension characteristic ('P': periodical; 'R': regional).
  - 0 : target grid first dimension number of overlapping grid points.

The fourth line gives the list of transformations to be performed for this field. In addition, there is one or more configuring lines describing some parameters for each transformation. These additional lines are described in more details in the chapter 4.

### 3.3.2 Second section of *namcouple* for `OUTPUT` fields

The first 2 lines for fields with status `OUTPUT` are as follows:

```
COSHFTOT  COSHFTOT   7   86400  0  fldhftot.nc OUTPUT
atmo   atmo
```

where the different entries are as for `EXPOUT` fields, except that the source symbolic name must be repeated twice on the field first line, the restart file name is needed only if `LOCTRANS` transformations are present, there is no output file name on the first line, and no grid dimensions and no LAG or SEQ index on the second line. The name of the output file is automatically defined based on the field and component model names.

The third line is `LOCTRANS` if this transformation is chosen for the field. Note that `LOCTRANS` is the only transformation supported for `OUTPUT` fields.

### 3.3.3 Second section of *namcouple* for `INPUT` fields

The first and only line for fields with status `INPUT` is:

```
SOALBEDO SOALBEDO  17  86400  0  SOALBEDO.nc  INPUT
```

where the different entries are:

- `SOALBEDO`: symbolic name for the field in the target model (`CHARACTER*8` repeated twice)
- 17: index in auxiliary file cf_name_table.txt (see above for EXPORTED fields)
- 86400: input period in seconds
- 0: number of transformations (always 0 for `INPUT` fields)
- `SOALBEDO.nc`: `CHARACTER*32` giving the input file name (for more detail on its format, see section 5.3)
- `INPUT`: field status.

[1]The 2D dimensions of the grids must be provided in the *namcouple* so to have 2D fields in the debug files; otherwise, the fields in the debug files will be 1D.

# Chapter 4

# Transformations and interpolations

Different transformations and 2D interpolations are available in OASIS3-MCT to adapt the coupling fields from a source model grid to a target model grid. In the following paragraphs, a description of each transformation with its corresponding configuring lines is given. Features that are now deprecated (non functional) compared to prior versions will be noted with the string UNUSED but not described.

## 4.1 Time transformations

- **LOCTRANS**:

  LOCTRANS requires one configuring line on which a time transformation, automatically performed below the call to oasis_put or prism_put_proto, should be indicated:

  ```
  # LOCTRANS operation
    $TRANSFORM
  ```

  where $TRANSFORM can be

  - INSTANT: no time transformation, the instantaneous field is transferred;
  - ACCUMUL: the field accumulated over the previous coupling period is exchanged (the accumulation is simply done over the arrays field_array provided as third argument to the oasis_put or prism_put_proto calls, not weighted by the time interval between these calls);
  - AVERAGE: the field averaged over the previous coupling period is transferred (the average is simply done over the arrays field_array provided as third argument to the oasis_put or prism_put_proto calls, not weighted by the time interval between these calls);
  - T_MIN: the minimum value of the field for each source grid point over the previous coupling period is transferred;
  - T_MAX: the maximum value of the field for each source grid point over the previous coupling period is transferred;
  - ONCE: UNUSED ; **not supported in OASIS3-MCT.**

  With OASIS3-MCT, time transformations are supported more generally with use of the coupling restart file. The coupling restart file allows the partial time transformation to be saved at the end of a run for exact restart at the start of the next run. For that reason, coupling restart filenames are now required for all *namcouple* transformations that use LOCTRANS (with non INSTANT values). In this mode, OASIS3-MCT will exit gracefully with an error message if a restart filename is not provided. This is the reason an optional restart file is now provided on the OUTPUT *namcouple* input line.

## 4.2 The pre-processing transformations

- **REDGLO** UNUSED
- **INVERT**: UNUSED
- **MASK**: UNUSED
- **EXTRAP**: UNUSED
- **CHECKIN**:

  CHECKIN calculates the global minimum, the maximum and the sum of the the source field values and prints them to the OASIS3-MCT debug file (for the master process of the source component model only). This operation does not transform the field.

  The generic input line is as follows, even if $NINT has no impact in OASIS3-MCT:

```
 # CHECKIN operation
     $INT = $NINT
```

- **CORRECT**: UNUSED
- **BLASOLD**:

  BLASOLD allows the source field to be scaled and allows a scalar to be added to the field. The prior ability to perform a linear combination of the current coupling field with other coupling fields has been deprecated in OASIS3-MCT. This transformation occurs before the interpolation *per se*.

  This transformation requires at least one configuring line with two parameters:

```
# BLASOLD operation
     $XMULT   $NBFIELDS
```

  where $XMULT is the multiplicative coefficient of the source field. $NBFIELDS must be 0 if no scalar needs to be added or 1 if a scalar needs to be added. In this last case, an additional input line is required where $AVALUE is the scalar to be added to the field :

```
     CONSTANT  $AVALUE
```

## 4.3 The remapping (interpolation)

- **MAPPING**:

  The MAPPING keyword is used to specify an input file to be read and used for mapping (ie. regridding or interpolation); the MAPPING file must follow the SCRIPR format. This is an alternative method to SCRIPR for setting the mapping file.

  In the current implementation, each pair of source and target points in the MAPPING file can be linked by only one weight, i.e. remappings such as SCRIPR/BICUBIC involving at each source grid point the value of the field, of the gradients and the cross-gradient, or second-order conservative remapping are not supported. Besides this restriction, MAPPING can be used with any mapping file produced by any regridding algorithm.

  This transformation requires at least one configuring line with one filename and two optional string values:

```
     $MAPNAME  $MAPLOC  $MAPSTRATEGY
```

  – $MAPNAME is the name of the mapping file to read. This is a netcdf file consistent with the SCRIPR map file format (see section 5.3).

  – $MAPLOC is optional and can be either src or dst. With src, the mapping will be done in parallel on the source processors before communication to the destination model and processors; this is the default. With dst, the mapping is done on the destination processors after the data is sent from the source model on the source grid.

– $MAPSTRATEGY is optional and can be either bfb, sum, or opt. In bfb mode, the mapping is done using a strategy that produces bit-for-bit identical results regardless of the grid decompositions without leveraging a partial sum computation. With sum, the transform is done using the partial sum approach which generally introduces roundoff level changes in the results on different processor counts. Option opt allows the coupling layer to choose either approach based on an analysis of which strategy is likely to run faster. Usually, partial sums will be used if the source grid has a higher resolution than the target grid as this should reduce the overall communication.

Note that if SCRIPR (see below) is used to calculate the remapping file, MAPPING can still be listed in the namcouple to specify a name for the remapping file generated by SCRIPR different from the default and/or to specify a $MAPLOC or $MAPSTRATEGY option.

- **SCRIPR**:

  SCRIPR gathers the interpolation techniques offered by Los Alamos National Laboratory SCRIP 1.4 library (Jones 1999)[1]. SCRIPR routines are in oasis3-mct/lib/scrip. See the SCRIP 1.4 documentation in oasis3/doc/SCRIPusers.pdf for more details on the interpolation algorithms. In the current implementation, each pair of source and target points in the MAPPING file can be linked by only one weight, i.e. remappings such as SCRIPR/BICUBIC involving at each source grid point the value of the field, of the gradients and the cross-gradient, or second-order conservative SCRIPR/CONSERV/SECOND remapping are not supported.

  When the SCRIP library performs a remapping, it first checks if the file containing the corresponding remapping weights and addresses exists. If it exists, it reads them from the file; if not, it calculates them and store them in a file. The file is created in the working directory and is called rmp_*srcg*_to_*tgtg*_*INTTYPE*_*NORMAOPT*.nc, where *srcg* and *tgtg* are the acronyms of respetively the source and the target grids, *INTTYPE* is the interpolation type, i.e. DISTWGT, GAUSWGT, BILINEAR (**not BILINEA as in OASIS3.3**) or CONSERV -see below, and *NORMAOPT* is the normalization option, i.e. DESTAREA, FRACAREA or FRACNNEI for CONSERV only -see below). One has to take care that the remapping file will have the same name even if other details, like the grid masks, are changed. When reusing a remapping file, one has to be sure that it was generated in exactly the same conditions than the ones it is used for.

  The following types of interpolations are available:

  – DISTWGT performs a distance weighted nearest-neighbour interpolation (N neighbours). All types of grids are supported.
    * Masked target grid points: the zero value is associated to masked target grid points.
    * Non-masked target grid points having some of the N source nearest neighbours masked: a nearest neighbour algorithm using the remaining non masked source nearest neighbours is applied.
    * Non-masked target grid points having all of the N source nearest neighbours masked: by default, the nearest non-masked source neighbour is used (logical ll_nnei hard-coded to .true. in oasis3-mct/lib/scrip/src/remap_distwgt.F; same default behaviour as OASIS3.3).

  The configuring line is:

  ```
  # SCRIPR (for DISWGT)
     $CMETH $CGRS $CFTYP $REST $NBIN $NV $ASSCMP $PROJCART
  ```

  * $CMETH = DISTWGT.
  * $CGRS is the source grid type (LR, D or U)- see annexe A.
  * $CFTYP is the field type: SCALAR. The option VECTOR, which in fact leads to a scalar treatment of the field (as in the previous versions), is still accepted. **VECTOR_I or**

[1] See also http://climate.lanl.gov/Software/SCRIP/ and the copyright statement in appendix 1.3.3.

**VECTOR_J, i.e. vector fields, are not supported anymore in OASIS3-MCT.**. See "Support of vector fields with the SCRIPR remappings" below.

* `$REST` is the search restriction type: `LATLON` or `LATITUDE` (see SCRIP 1.4 documentation SCRIPusers.pdf).
* `$NBIN` the number of restriction bins (see SCRIP 1.4 documentation SCRIPusers.pdf). Note that for D or U grid, the restriction may influence sligthly the result near the borders of the restriction bins.
* `$NV` is the number of neighbours used.
* `$ASSCMP`: UNUSED; **vector fields are not supported anymore in OASIS3-MCT.** See "Support of vector fields with the SCRIPR remappings" below.
* `$PROJCART`: UNUSED; **vector fields are not supported anymore in OASIS3-MCT.** See "Support of vector fields with the SCRIPR remappings" below.

– `GAUSWGT` performs a N nearest-neighbour interpolation weighted by their distance and a gaussian function. All grid types are supported.

* Masked target grid points: the zero value is associated to masked target grid points.
* Non-masked target grid points having some of the N source nearest neighbours masked: a nearest neighbour algorithm using the remaining non masked source nearest neighbours is applied.
* Non-masked target grid points having all of the N source nearest neighbours masked: by default, the nearest non-masked source neighbour is used, i.e. logical `ll_nnei` hard-coded to `.true.` in `oasis3-mct/lib/scrip/src/remap_gauswgt.F`; **this is NOT the same default behaviour as OASIS3.3**; to have the same default behaviour as in OASIS3.3, put `ll_nnei=.false.`.

The configuring line is:

```
# SCRIPR (for GAUSWGT)
    $CMETH  $CGRS  $CFTYP  $REST  $NBIN  $NV $VAR
```

where: all entries are as for `DISTWGT`, except that:

* `$CMETH = GAUSWGT`
* `$VAR`, which must be given as a REAL value (e.g 2.0 and not 2), defines the weight given to a neighbour source grid point as proportional to $exp(-1/2 \cdot d^2/\sigma^2)$ where $d$ is the distance between the source and target grid points, and $\sigma^2 = \$VAR \cdot \overline{d}^2$ where $\overline{d}^2$ is the average distance between two source grid points (calculated automatically by OASIS3-MCT).

– `BILINEAR` performs an interpolation based on a local bilinear approximation (see details in chapter 4 of SCRIP 1.4 documentation SCRIPusers.pdf)

For `BILINEAR`, Logically-Rectangular (LR) and Reduced (D) source grid types are supported.

* Masked target grid points: the zero value is associated to masked target grid points.
* Non-masked target grid points having some of the source points normally used in the bilinear or bicubic interpolation masked: a N nearest neighbour algorithm using the remaining non masked source points is applied.
* Non-masked target grid points having all bilinear neighbours masked: by default, the nearest non-masked source neighbour is used (logical `ll_nnei` hard-coded to `.true.` in `oasis3-mct/lib/scrip/src/remap_bilinear.F`; **this is NOT the same default behaviour as OASIS3.3**; to have the same default behaviour as in OASIS3.3, put `ll_nnei=.false.`.

The configuring line is:

```
# SCRIPR  (for BILINEAR)
    $CMETH  $CGRS  $CFTYP  $REST  $NBIN
```

where:

* $CMETH = BILINEAR
* $CGRS is the source grid type (LR or D)
* $CFTYP, $NBIN are as for DISTWGT.
* $REST is as for DISTWGT, except that only LATITUDE is possible for a Reduced (D) source grid.

– BICUBIC: **as opposed to OASIS3.3, OASIS3-MCT does not currently support this mapping option because it is higher order.**

– CONSERV performs 1st order conservative remapping, which means that the weight of a source cell is proportional to area intersected by the target cell. **Note that 2nd order conservative mapping is not supported yet with OASIS3-MCT**.

The configuring line is:

```
# SCRIPR (for CONSERV)
    $CMETH  $CGRS  $CFTYP  $REST  $NBIN  $NORM  $ORDER
```

where:

* $CMETH = CONSERV
* $CGRS is the source grid type: LR, D and U are supported for first-order remapping if the grid corners are given by the user in the grid data file grids.nc ; only LR is supported if the grid corners are not available in grids.nc and therefore have to be calculated automatically by OASIS3.
* $CFTYP, $REST, $NBIN are as for DISTWGT.
* $NORM is the NORMalization option:
  · FRACAREA: The sum of the non-masked source cell intersected areas is used to NORMalise each target cell field value: the flux is not locally conserved, but the flux value itself is reasonable.
  · DESTAREA: The total target cell area is used to NORMalise each target cell field value even if it only partly intersects non-masked source grid cells: local flux conservation is ensured, but unreasonable flux values may result.
  · FRACNNEI: as FRACAREA, except that at least the source nearest unmasked neighbour is used for unmasked target cells that intersect only masked source cells.
* $ORDER: FIRST for first order remapping respectively (see SCRIP 1.4 documentation). **Note that 2nd order conservative mapping is not supported yet with OASIS3-MCT.**

**Precautions related to the use of the SCRIPR/CONSERV remapping**

– For the 1st order conservative remapping: the weight of a source cell is proportional to area of the source cell intersected by target cell. Using the divergence theorem, the SCRIP library evaluates this area with the line integral along the cell borders enclosing the area. As the real shape of the borders is not known (only the location of the 4 corners of each cell is known), the library assumes that the borders are linear in latitude and longitude between two corners. This assumption becomes less valid closer to the pole and for latitudes above the north_thresh or below the south_thresh values (see oasis3-mct/lib/scrip/remap_conserv.F, the library evaluates the intersection between two border segments using a Lambert equivalent azimuthal projection. Problems were observed in some cases for the grid cell located around this north_thresh or south_thresh latitude.

– Another limitation of the SCRIP 1st order conservative remapping algorithm is that is also supposes, for line integral calculation, that $sin(latitude)$ is linear in longitude on the cell

borders which again is in general not valid close to the pole.

- For a proper consevative remapping, the corners of a cell have to coincide with the corners of its neighbour cell, with no "holes" between the cells.
- If two cells of one same grid overlay, at least the one with the greater numerical index must be masked for a proper conservative remapping. For example, if the grid cells with i=1 overlays the grid cells with i=imax, the latter must be masked. If this is not the case given the mask defined in *masks.nc*, OASIS3-MCT must be compiled with the CPP key `TREAT_OVERLAY` which will ensure that these rules are respected. This CPP key was introduced in OASIS3.3.
- A target grid cell intersecting no source cell (either masked or non masked) at all i.e. falling in a "hole" of the source grid will in all cases get a zero value.
- If a target grid cell intersects only masked source cells, it will still get a zero value unless the `FRACNNEI` normalisation option is used, in which case it will get the nearest non masked neighbour value. **Note that the option of having the value 1.0E+20 assigned to these target grid cell intersecting only masked source cells (for easier identification) is not yet availble in OASIS3-MCT.**

**Support of vector fields with the SCRIPR remappings**

Vector mapping is NOT supported is not and will not be supported by OASIS3-MCT. For proper treatment of vector fields, the component model has to send the 3 components of the vector projected in a Cartesian coordinate system.

- **INTERP**: UNUSED
- **MOZAIC**: UNUSED; note that `MAPPING` (see above) is the NetCDF equivalent to `MOZAIC`.
- **NOINTERP**: UNUSED
- **FILLING**: UNUSED

## 4.4 The post-processing stage

- **CONSERV**:

  `CONSERV` ensures a global modification of the coupling field. This analysis requires the source and target grid mesh areas to be transfered to the coupler with `oasis_write_area/prism_write_area` (see section 2.3). **For a correct CONSERV operation, overlapping grid cells on the source grid or on the target grid must be masked.** In the *namcouple*, `CONSERV` requires one input line with one argument and one optional argument:

```
# CONSERV operation
     $CMETH  $CONSOPT
```

  where:

  - `$CMETH` is the method desired with the following choices
    - with `$CMETH = GLOBAL`, the field is integrated on both source and target grids, without considering values of masked points, and the residual (target - source) is uniformly distributed on the target grid; this option ensures global conservation of the field
    - with `$CMETH = GLBPOS`, the same operation is performed except that the residual is distributed proportionally to the value of the original field; this option ensures the global conservation of the field and does not change the sign of the field
    - with `$CMETH = BASBAL`, the operation is analogous to `GLOBAL` except that the non masked surface of the source and the target grids are taken into account in the calculation of the residual; this option does not ensure global conservation of the field but ensures that the energy received is proportional to the non masked surface of the target grid

    * with `$CMETH = BASPOS`, the non masked surface of the source and the target grids are taken into account and the residual is distributed proportionally to the value of the original field; therefore, this option does not ensure global conservation of the field but ensures that the energy received is proportional to the non masked surface of the target grid and it does not change the sign of the field.
  - `$CONSOPT` is an optional argument specifying the algorithm. `$CONSOPT` can be `bfb` or `opt`. The `bfb` option enforces a bit-for-bit transformation regardless of the grid decomposition or process count. The `opt` option carries out the conservation using an optimal algorithm using less memory and a faster approach. Option `bfb` is the default setting.
- **SUBGRID**: UNUSED
- **BLASNEW**:

  `BLASNEW` performs a scalar multiply or scalar add to any destination field. This is the equivalent of BLASOLD on the destination side. The prior feature that supported linear combinations of the current coupling field with any other fields after the interpolation has been deprecated.

  This analysis requires the same input line(s) as `BLASOLD`.
- **MASKP**: UNUSED
- **REVERSE**: UNUSED
- **CHECKOUT**:

  `CHECKOUT` calculates the global minimum, the maximum and the sum of the the target field values and prints them to the OASIS3-MCT debug file (for the master process of the target component model only). This operation does not transform the field. The generic input line is as for `CHECKIN` (see above).
- **GLORED**: UNUSED

# Chapter 5

# OASIS3-MCT auxiliary data files

OASIS3-MCT may use auxiliary data files, e.g. defining the grids of the models being coupled, containing the field coupling restart values or input data values, or the remapping weights and addresses.

## 5.1 Grid data files

With OASIS3-MCT, the grid data files *grids.nc, masks.nc* and *areas.nc* are required only for certain operations, i.e. *grids.nc*, and *masks.nc* for `SCRIPR` (see section 4.3) and *masks.nc* and *areas.nc* for `CONSERV` (see section 4.4). These grid data files can be created by the user before the run or can be written directly at run time by the **master process of each component model** using the grid data definition routines (see section 2.3). These routines can be used by the component models to add grid fields to the grid files but grid fields are **never** overwritten in the grid files. These files are netCDF.

The arrays containing the grid information are dimensioned `(nx, ny)`, where `nx` and `ny` are the grid first and second dimension. Unstructured grids are supported by setting the ny dimension to 1 and then nx is the total number of grid points.

1. *grids.nc*: contains the model grid longitudes and latitudes in single or double precision `REAL` arrays (depending on OASIS3-MCT compilation options). The array names must be composed of a prefix (4 characters), given by the user in the *namcouple* on the second line of each field (see section 3.3), and of a suffix (4 characters); this suffix is ".lon" or ".lat" for respectively the grid point longitudes or latitudes.

   If the `SCRIPR/CONSERV` remapping is used, longitudes and latitudes for the source and target grid **corners** must also be available in the *grids.nc* file as arrays dimensioned `(nx,ny,4)` or `(nbr_pts,1,4)` where `4` is the number of corners (in the counterclockwize sense). The names of the arrays must be composed of the grid prefix and the suffix ".clo" or ".cla" for respectively the grid corner longitudes or latitudes. As for the other grid information, the corners can be provided in *grids.nc* before the run by the user or directly by the model through specific calls (see section 2.3).

   Longitudes must be given in degrees East in the interval -360.0 to 720.0. Latitudes must be given in degrees North in the interval -90.0 to 90.0. Note that if some grid points overlap, it is recommended to define those points with the same number (e.g. 360.0 for both, not 450.0 for one and 90.0 for the other) to ensure automatic detection of overlap by OASIS3-MCT.

   The corners of a cell cannot be defined modulo 360 degrees. For example, a cell located over Greenwich will have to be defined with corners at -1.0 deg and 1.0 deg but not with corners at 359.0 deg and 1.0 deg.

   Cells larger than 180.0 degrees in longitude are not supported.

2. *masks.nc*: contains the masks for all component model grids in `INTEGER` arrays. **Be careful to note the historical OASIS convention: 0 -not masked i.e. active- or 1 -masked i.e. not active-**

**for each grid point**. The array names must be composed of the grid prefix and the suffix ".msk". This file, *masks* or *masks.nc*, is mandatory.

3. *areas.nc*: this file contains mesh surfaces for the component model grids in single or double precision `REAL` arrays (depending on OASIS3-MCT compilation options). The array names must be composed of the grid prefix and the suffix ".srf". The surfaces may be given in any units but they must be all the same. This file *areas.nc* is mandatory for `CONSERV/GLOBAL`,`/GLBPOS`, `/BASBAL`, and `/BASPOS`; it is not required otherwise.

## 5.2 Coupling restart files

At the beginning of a coupled run, some coupling fields may have to be initially read from their coupling restart file on their source grid (see the LAG concept in section 2.10). When needed, these files are also automatically updated by the last active `oasis_put` or `prism_put_proto` call of the run (see section 2.7.1) . **Warning**: the date is not written or read to/from the restart file; therefore, the user has to make sure that the appropriate coupling restart file is present in the working directory. The coupling restart files must follow the NetCDF format.

The name of the coupling restart file is given by the 6th character string on the first configuring line for each field in the *namcouple* (see section 3.3). Coupling fields coming from different models cannot be in the same coupling restart files, but for each model, there can be an arbitrary number of fields written in one coupling restart file.

In the coupling restart files, the fields must be provided on the source grid in single or double precision REAL arrays and, as the grid data files, must be dimensioned `(nx, ny)`, where `nx` and `ny` are the grid first and second dimension, except for fields given on unstructured for which the arrays are dimensioned `(nt,1)`, where `nt` is the total number of grid points. The shape and orientation of each restart field (and of the corresponding coupling fields exchanged during the simulation) must be coherent with the shape of its grid data arrays.

## 5.3 Input data files

Fields with status `INPUT` in the *namcouple* will, at runtime, simply be read in from a NetCDF input file by the target model below the `oasis_get` or `prism_get_proto` call, at appropriate times corresponding to the input period indicated by the user in the *namcouple*.

The name of the file must be the one given on the field first configuring line in the *namcouple* (see section 3.3.3). There must be one input file per `INPUT` field, containing a time sequence of the field in a single or double precision REAL array named with the field symbolic name in the *namcouple* and dimensioned `(nx,ny,time)` or `(nbr_pts,1,time)`. The time variable has to be an array `time(time)` expressed in "seconds since beginning of run". The "time" dimension has to be the unlimited dimension.

## 5.4 Transformation auxiliary data files

### 5.4.1 Files containing the remapping weights and addresses for `MAPPING`

The mapping files to be read using the MAPPING option are consistent with the files generated in OASIS3.3 or OASIS3-MCT using the SCRIP library (see 5.4.2 below). The files are netcdf and the key fields are num_links (the number of weights for each grid pair), src_grid_size (the 2d size of the source grid), dst_grid_size (the 2d size of the destination grid), num_wgts (the total number of weights), remap_matrix (the weights), dst_address (the global destination index for the weight), src_address (the global source index for the weight).

### 5.4.2 Auxiliary data files for `SCRIPR`

The NetCDF files containing the weights and addresses for the `SCRIPR` remappings (see section 4.3) are automatically generated at runtime by OASIS3-MCT. Their structure is described in detail in section 2.2.3 of the SCRIP documentation available in `oasis3-mct/doc/SCRIPusers.pdf`. In particular, they are netCDF files with the following one fields

- src_grid_size is a scalar integer indicating the total number of global grid cells for the source grid. This field in a netCDF dimension.
- dst_grid_size is a scalar integer indicating the total number of global grid cells for the destination (or target) grid. This field in a netCDF dimension.
- num_links is a scalar integer indicating the total number of associated grid pairs in the file. This is typically a large number. This field is a netCDF dimension.
- num_wgts is a scalar integer indicating the number of weights per associated grid pair. For first order mapping, this is 1. This field in a netCDF dimension.
- src_address is a one dimensional array of size num_links. It contains the integer source address associated with each weight. This field is a netCDF variable.
- dst_address is a one dimensional array of size num_links. It contains the integer destination address associated with each weight. This field is a netCDF variable.
- remap_matrix is a two dimensional array of size num_links, num_wgts. It contains the real weight value(s) associated with the source and destination address. This field is a netCDF variable.

# Chapter 6

# Compiling, running and debugging

## 6.1 Compiling OASIS3-MCT

Compiling OASIS3-MCT can be done in directory `oasis3-MCT/util/make_dir` with Makefile `TopMakefileOasis3` which must be completed with a header file `make.`*your_platform* specific to the compiling platform used and specified in `oasis3-MCT/util/make_dir/make.inc`. One of the header files distributed with the release can by used as a template. The root of the OASIS3-MCT tree can be anywhere and must be set in the variable `COUPLE` in the `make.`*your_platform* file.

The following commands are available:

- `make -f TopMakefileOasis3` or `make oasis3_psmile -f TopMakefileOasis3` (for backward compatibility):

  compiles all OASIS3-MCT libraries *mct*, *scrip* and *psmile*;

- `make realclean -f TopMakefileOasis3`:

  removes all OASIS3-MCT compiled sources and librairies.

Log and error messages from compilation are saved in the files COMP.log and COMP.err in make_dir.

During compilation, a new compiling directory, defined by variable `ARCHDIR` is created. After successful compilation, resulting executables are found in the compiling directory in `/bin`, libraries in `/lib` and object and module files in `/build`. If mod_ oasis is used in the models, it is enough to include the path of the psmile objects and modules for the compilation and to use the psmile library when linking. If mod_prism is used in the models, it is necessary to include the path of the psmile and of the mct objects and modules for the compilation and to use both the psmile and mct libraries when linking.

The following CPP keys are coded in OASIS3-MCT and can be specified in `CPPDEF` in `make.`*your_platform* file.

- `TREAT_OVERLAY`: To ensure, in `SCRIPR/CONSERV` remapping (see section 4.3), that if two cells of the source grid overlay, at least the one with the greater numerical index is masked (they also can be both masked); this is mandatory for this remapping. For example, if the grid line with i=1 overlaps the grid line with i=imax, it is the latter that must be masked; when this is not the case with the mask defined in *masks.nc*, this CPP key forces these rules are to be respected.

## 6.2 Running OASIS3-MCT

A practical example on how to run OASIS3-MCT and running it in a coupled system is provided in `oasis3-mct/examples/tutorial`. For more details, see the README there in.

## 6.3 Debugging

### 6.3.1 Debug files

If you experience problems while running your coupled model with OASIS3-MCT, you can obtain more information on what is happening by increasing `$NLOGPRT` in your *namcouple* (see also section 3.2).

Different output are written depending on NLOGPRT value:

- 0 : one file debug.root.xx is open by the master proces of each model and one file debug_notroot.xx is open for all the other processes of each model to write error information.
- 1 : one file debug.root.xx is open by the master proces of each model to write information equivalent to level 10 (see below); one file debug_notroot.xx is open for all the other processes of each model to write error information.
- 2 : one file debug.xx.xxxxxx is open by each process of each model to write normal production diagnostics.
- 5 : as for 2 with in addition some initial debug info.
- 10 : as for 5 with in addition the routine calling tree.
- 12 : as for 10 with in addition some routine calling notes.
- 15 : as for 12 with even more debug diagnostics.
- 20 : as for 15 with in addition some extra runtime analysis.
- 30 : full debug information.

### 6.3.2 Time statistics files

The variable TIMER_ Debug, defined in mod_ oasis_ timer, is used to obtain time statistics over all the processors for each routine.

Different output are written (in files named *.timers_ xxxx) depending on TIMER_ Debug value :

- TIMER_ Debug=0 : nothing is calculated, nothing is written.
- TIMER_ Debug=1 : the times are calculated and written in a single file by the processor 0 as well as the min and the max times over all the processors.
- TIMER_ Debug=2 : the times are calculated and each processor writes its own file ; processor 0 also writes the min and the max times over all the processors in its file.
- TIMER_ Debug=3 : the times are calculated and each processor writes its own file ; processor 0 also writes in its file the min and the max times over all processors and also writes in its file all the results for each processor.

The time given for each routine is the accumulated time over the entire run.

oasis_ timer_start() and oasis_ timer_ stop() are the routines used to calculate the time of each routine.

Below is an overview of the meaning of the routines for an example where a model receives field1 and sends field2 :

- 'total after init' : represents the total time of the simulation, implemented in mod_ prism_ method : oasis_ timer_ start() is called after initialisation (ie after oasis_ namcouple_ init and mct_ world_ init) and oasis_ timer_ stop() before mpi_ barrier et mpi_ finalize (routine oasis_ terminate).
- 'map definition' : oasis_ timer_ start() is called before and oasis_ timer_ stop() after mct_ gsmap_ init in mod_ oasis_ part (routine oasis_ def_ partition). It defines which source processor has to communicate with which target processor.
- 'cpl_ setup' : oasis_ timer_ start() is called before and oasis_ timer_ stop() after oasis_ coupler_ setup. It "reconciles the coupling stuff" called by oasis_ method_ enddef.

- 'cpl_ smatrd' : oasis_ timer_ start() is called before and oasis_ timer_ stop() after oasis_ coupler_ smatreaddnc in mod_ oasis_ coupler (routine oasis_ coupler_ setup). It does a distributed read of a NetCDF SCRIP file and return weights in a distributed SparseMatrix, and calls mct_ sMatP_ Init which initialises the sparse matrix vector multiplication.
- 'oasis_ advance_ init()' : oasis_ timer_ start() is called at the beginning of the routine and oasis_ timer_ stop() at the end of the routine (in mod_ oasis_ advance). It handles initial restart and communication of data for fields with positive lags.
- 'grcv_ 001' : in mod_ oasis_ advance (routine oasis_ advance_ run) oasis_ timer_ start() is called before and oasis_ timer_ stop() after mct_ recv for field 1.
- 'wout_ 001' : in mod_ oasis_ advance (routine oasis_ advance_ run) oasis_ timer_ start() is called before and oasis_ timer_ stop() after I/O of debug for field 1.
- 'gcpy_ 001' : in mod_ oasis_ advance (routine oasis_ advance_ run) after the get, oasis_ timer_ start() is called before and oasis_ timer_ stop() after copying received array field 1 in array(n).
- 'pcpy_ 002' : in mod_ oasis_ advance (routine oasis_ advance_ run) before the put, oasis_ timer_ start() is called before and oasis_ timer_ stop() after copying array(n) in array to send (i.e. with local transformation besides division for averaging) for field 2.
- 'pavg_ 002' : in mod_ oasis_ advance (routine oasis_ advance_ run) before the put, oasis_ timer_ start() is called before and oasis_ timer_ stop() after average as needed for field 2.
- 'pmap_ 002' : in mod_ oasis_ advance (routine oasis_ advance_ run) before the put, oasis_ timer_ start() is called before and oasis_ timer_ stop() after matrix vector multiplication for field 2.
- 'psnd_ 002' : in mod_ oasis_ advance (routine oasis_ advance_ run) oasis_ timer_ start() is called before and oasis_ timer_ stop() after the mct_ waitsend and mct_ isend for field 2.
- 'wout_ 002' : in mod_ oasis_ advance (routine oasis_ advance_ run) oasis_ timer_ start() is called before and oasis_ timer_ stop() after I/O of debug for field 2.

# Appendix A

# The grid types for the transformations

As described in section 4, the different SCRIP remappings OASIS3-MCT support different types of grids. The characteristics of these grids are detailed here:

- `'LR' grid`: The longitudes and the latitudes of 2D Logically-Rectangular (LR) grid points can be described by two arrays `longitude(i,j)` and `latitude(i,j)`, where i and j are respectively the first and second index dimensions. Streched or/and rotated grids are LR grids. Note that A, B, G, L, Y, or Z grids are all particular cases of LR grids.
- `'U' grid`: Unstructured (U) grids do have any particular structure. The longitudes and the latitudes of 2D Unstructured grid points must be described by two arrays `longitude(nbr_pts,1)` and `latitude(nbr_pts,1)`, where nbr_pts is the total grid size.
- `'D' grid` The Reduced (D) grid is composed of a certain number of latitude circles, each one being divided into a varying number of longitudinal segments. In OASIS3, the grid data (longitudes, latitudes, etc.) must be described by arrays dimensioned `(nbr_pts,1)`, where `nbr_pts` is the total number of grid points. There is no overlap of the grid, and no grid point at the equator nor at the poles. There are grid points on the Greenwich meridian.

# Appendix B

# Changes between previous OASIS3.3 and new OASIS3-MCT

## B.1 General architecture

- OASIS3-MCT is (only) a coupling library

  Much of the underlying implementation of OASIS3 was refactored to leverage the Model Coupling Toolkit (MCT). OASIS3-MCT is a coupling library to be linked to the component models and that carries out the coupling field transformations (e.g. remappings/interpolations) in parallel on either the source or target processes and that performs all communication in parallel directly between the component models; there is no central coupler executable anymore[1].

- `MAPPING` transformation to use a pre-defined mapping file

  With `MAPPING`, OASIS3-MCT has the ability to read a predefined set of weights and addresses (mapping file) specified in the *namcouple* to perform the interpolation/remapping. The user also has the flexibility to choose the location and the parallelization strategy of the remapping with specific `MAPPING` options (see section 4.3).

- Mono-process mapping file generation with the SCRIP library

  But as before, OASIS3-MCT can also generate the mapping file using the SCRIP library (Jones 1999). When this is the case, the mapping file generation is done on one process of the model components; all previous `SCRIPR` remapping schemes available in OASIS3.3 are still supported besides `BICUBIC` and `CONSERV/SECOND` (see `SCRIPR` in section 4.3).

- MPI2 job launching is NOT supported.

  Only MPI1 start mode is allowed. As before with the MPI1 mode, all component models must be started by the user in the job script in a pseudo-MPMD mode; in this case, they will automatically share the same `MPI_COMM_WORLD` communicator and an internal communicator created by OASIS3-MCT needs to be used for internal parallelization (see section 2.2.2).

## B.2 Changes in the coupling interface in the component models

- Use statement

  The different OASIS3.3 `USE` statements were gathered into one `USE mod_oasis` (or one `USE mod_prism`), therefore much simpler to use.

[1] As with OASIS3.3, the "put" calls are non-blocking but the "get" calls are blocking. As before, the user has to take care of implementing a coupling algorithm that will result in matching "put" and "get" calls to avoid deadlocks (see section 2.7). The lag (LAG) index works as before in OASIS3.3 (see section 2.10)

- Support of previous prism_xxx and new oasis_xxx interfaces

  OASIS3-MCT retains prior interface names of OASIS3.3 (e.g. prism_put_proto) to ensure full backward compatibility. However, new interface names such as oasis_put are also available and should be prefered. Both routine names are listed in chapter 2.

- Auxiliary routines not supported yet

  Auxiliary routines prism_put_inquire, prism_put_restart_proto, prism_get_freq are not supported yet.

- Support of components for which only a subset of processes participate in the coupling

  New routines oasis_create_couplcomm and oasis_set_couplcomm are now available to create or set a coupling communicator in the case only a subset of the component processes participate in the coupling (see section 2.2.3). But even in this case, all OASIS3-MCT interface routines, besides the grid definition (see section 2.3) and the "put" and " get" call per se (see section 2.7), are collective and must be called by all processes.

- New routines oasis_get_debug and oasis_set_debug

  New routines oasis_get_debug and oasis_set_debug are now available to respectively retrieve the current OASIS3-MCT internal debug level (set by $NLOGPRT in the *namcouple*) or to change it (see section 2.9).

## B.3 Functionality not offered anymore

- SCRIPR/BICUBIC and SCRIPR/CONSERV/SECOND remappings

  As in OASIS3.3, the SCRIP library can be used to generate the remapping/interpolation weights and addresses and write them to a mapping file. All previous SCRIPR remapping schemes available in OASIS3.3 are still supported besides BICUBIC and CONSERV/SECOND because these remapping involve at each source grid point the value of the field but also the value of the gradients of the field (which are not known or calculated). See also section 4.3.

- Vector field remapping

  Vector field remapping is not and will not be supported (see "Support of vector fields with the SCRIPR remappings" in section 4.3).

- Automatic calculation of grid mesh corners in SCRIPR/CONSERV

  For SCRIPR/CONSERV remapping, grid mesh corners will not be compute automatically if they are needed but not provided.

- Other transformations not supported
  - The following transformations are not available in OASIS3.3 and will most probably not be implemented as it should be not too difficult to implement the equivalent operations in the component models themselves: CORRECT, FILLING, SUBGRID, MASKP
  - LOCTRANS/ONCE is not explicitely offered as it is equivalent to defining a coupling period equal to the total runtime.
  - The following transformations are not available as they were already deprecated in OASIS3.3 (see OASIS3.3 User Guide): REDGLO, INVERT, REVERSE, GLORED
  - MASK and EXTRAP are not available but the corresponding linear extrapolation can be replaced by the more efficient option using the nearest non-masked source neighbour for target points having their original neighbours all masked. This is now the default option for SCRIPR/DISTWGT, GAUSWGT and BILINEAR interpolations. It is also included in SCRIPR/CONSERV if FRACNNEI normalization option is chosen (see section 4.3).
  - INTERP interpolations are not available; SCRIPR should be used instead.
  - MOZAIC is not available as MAPPING should be used instead.

  - NOINTERP does not need to be specified anymore if no interpolation is required.
  - Field combination with BLASOLD and BLASNEW ; these transformations only support multiplication and addition terms to the fields (see section 4.2).

- Using the coupler in interpolator-only mode

  This is not possible anymore as OASIS3-MCT is now only a coupling library. However, it is planned, in a further release, to provide a toy coupled model that could be use to check the quality of the remapping for any specific couple of grids.

- Coupling field CF standard names

  The file cf_name_table.txt is not needed or used anymore. The CF standard names of the coupling fields are not written to the debug files.

- Binary auxiliay files

  All auxiliary files, besides the *namcouple* must be NetCDF; binary files are not supported anymore.

## B.4 New functionality offered

- Better support of components for which only a subset of processes participate in the coupling

  In OASIS3.3, components for which only a subset of processes participated in the coupling were supported in a very restricted way. In fact, this subset had to be composed of the N first processes and N had to be specified in the *namcouple*. Now, the subset of processes can be composed of any of the component processes and does not have to be pre-defined in the *namcouple*. New routines oasis_create_couplcomm and oasis_set_couplcomm are now available to create or set a coupling communicator gathering only these processes (see section 2.2.3).

- Exact restart for LOCTRANS transformations

  If needed, LOCTRANS transformations write partially transformed fields in the coupling restart file at the end of a run to ensure an exact restart of the next run (see section 4.1). Therefore, fields of type OUTPUT now support a restart file specification in the *namcouple* to be used for LOCTRANS transformations.

- Support to couple multiple fields via a single communication.

  This is supported through colon delimited field lists in the *namcouple*, for example

  ```
  ATMTAUX:ATMTAUY:ATMHFLUX TAUX:TAUY:HEATFLUX 1 3600 3 rstrt.nc EXPORTED
  ```

  in a single *namcouple* entry. All fields will use the *namcouple* settings for that entry. In the component model codes, these fields are still sent ("put") or received ("get") one at a time. Inside OASIS3-MCT, the fields are stored and a single mapping and send or receive instruction is executed for all fields. This is useful in cases where multiple fields have the same coupling transformations and to reduce communication costs by aggregating multiple fields into a single communication.

- Matching one source filed with multiple targets

  A coupling field sent by a source component model can be associated with more than one target field and model (get). In that case, the source model needs to send ("put") the field only once and the corresponding data will arrive at multiple targets as specified in the *namcouple* configuration file. Different coupling frequencies and transformations are allowed for different coupling exchanges of the same field. The inverse feature is not allowed, i.e. a single target (get) field CANNOT be associated with multiple source (put) fields.

- The debug files

  The debug mode was greatly improved as compared to OASIS3.3. The level of debug information written out to the OASIS3-MCT debug files for each model process is defined by the $NLOGPRT value in the *namcouple*. All details are provided in section 3.2.

## B.5 Changes in the configuration file *namcouple*

- The *namcouple* configuration file of OASIS3-MCT is fully backward compatible with OASIS3.3. However, several *namcouple* keywords have been deprecated even if they are still allowed. These keywords are noted "UNUSED" in sections 3.2 and 3.3 and are not fully described. Information below these keywords will not be read and will not be used: $SEQMODE , $CHANNEL, $JOBNAME, $INIDATE, $MODINFO, $CALTYPE.
- Also the following inputs should not appear in the *namcouple* anymore as the related functionality are not supported anymore in OASIS3-MCT (see above): field status AUXILARY, time transformation ONCE, REDGLO, INVERT, MASK, EXTRAP, CORRECT, INTERP, MOZAIC, FILLING, SUBGRID, MASKP, REVERSE, GLORED.
- To get 2D fields in the debug output NetCDF files, the 2D dimensions of the grids must be provided in the *namcouple* (except if the field has the status OUTPUT); otherwise, the fields in the debug output files will be 1D.

## B.6 Other differences

- IGNORED and IGNOUT fields are converted to EXPORTED and EXPOUT respectively.
- The file `areas.nc` is not needed anymore to calculate some statistics with options CHECKIN and/or CHECKOUT (see section 4.2).
- SEQ index is no longer needed to ensure correct coupling sequencing within the coupler. Use of SEQ allows the coupling layer to detect potential deadlocks before they happen and to exit gracefully (see section 2.10.2).
- The I/O library mpp_io is no longer used to write the restart and output files.

# Appendix C

# Coupled models realized with OASIS

**Table C.1:** Use of OASIS3: centres, coupled models and computing platforms

| Centre | Coupled model | Platform |
|---|---|---|
| CERFACS (FR) | ARPEGE4_T63/NEMO-ORCA2-LIM/TRIP | Linux Cluster, CRAY XD1, VPP5000, NEC SX6-SX8 |
| CERFACS (FR) | ARPEGE_T63/NEMIX-ORCA2 | CRAY XD1 |
| CERFACS (FR) | ARPEGE_T359/NEMO-ORCA012 | NEC SX9 |
| CERFACS (FR) | ARPEGE_T799/NEMIX-ORCA025 | Bullx, Altix ICE |
| Météo-France (FR) | ARPEGE_V4.6/NEMO-ORCA2/NEMOmed8 | NEC-SX8 |
| Météo-France (FR) | ALADIN-Climat/NEMOmed8/TRIP | NEC-SX9 |
| Météo-France (FR) | ARPEGE_V5/NEMOV3-ORCA1/TRIP | NEC-SX9 |
| Météo-France (FR) | ARPEGE_V5.1/NEMO1 | IBM Power 6 |
| IPSL (FR) | LMDZ/NEMO-ORCA2/ - LMDZmed/NEMOmed8 | |
| IPSL (FR) | LMDz(144x142)/NEMO-ORCA2 | NEC SX8 |
| OMP (FR) | MESO-NH/SYMPHONIE | Linux Opteron cluster |
| LGGE (FR) | MAR/NEMO-LIM | |
| ECMWF | IFS_T399/NEMO-ORCA1 | IBM Power 6 |
| MPI-M (DE) | ECHAM5/MPIOM | IBM Power 6, SUN Linux |
| MPI-M (DE) | REMO/MPIOM | IBM Power 575 |
| Met Office (UK) | UM Atm(192x145)/NEMO-ORCA1 | IBM Power6, NEC SX6 |
| Met Office (UK) | UM Atm(432x325/NEMO-ORCA025 | IBM Power6 |
| NCAS/Reading (UK) | ECHAM4/NEMO-ORCA2 | NEC SX6-SX8 |
| NCAS/Reading (UK) | HadAM3/NEMO-ORCA2 | NEC SX8 |
| IFM-GEOMAR (DE) | ECHAM5_T63/NEMO-ORCA2 | NEC SX6, SX8, SX9 |
| CMCC (IT) | ECHAM5_T31L31/OPA8.2-ORCA2 | NEC SX9 |
| CMCC (IT) | ECHAM5_T159L31/OPA8.2-ORCA2 | IBM Power6 |
| CMCC (IT) | ECHAM5_T63L95/OPA8.2-ORCA2 | |
| CMCC (IT) | ECHAM5_T159/OPA8.2-ORCA2/ - NEMOMed_1/16 | |
| ENEA (IT) | RegCM/MITgcm | IBM-SP5 |
| LMNCP (IT) | WRF/ROMS | |
| SMHI (SE) | EC-Earth: IFS_T159/NEMO-ORCA1 | Linux Cluster |
| SMHI (SE) | EC-Earth: IFS_T799/NEMO-ORCA025 | Ekman AMD Opteron |
| KNMI (NL) | ECHAM5/MPIOM | NEC SX-8 |
| KNMI (NL) | EC-Earth: IFS_T159/NEMO-ORCA2 | SGI Altix, IBM Power |
| DMI (DK) | ECHAM(global)/HIRLAM (reg) | NEC SX6 |
| U.Bergen (NO) | MM5/ROMS | |
| ICHEC (IE) | EC-Earth: IFS_T159/NEMO-ORCA1 | SGI Altix ICE, Bull clust. |
| ICHEC (IE) | ROMS/WRF | |
| NUI Galway (IE) | ECHAM5/REMO/MPI-OM | |

Continued on next page

**Table C.1 – continued from previous page**

| Centre | Coupled model | Platform |
|---|---|---|
| ETH (CH) | COSMO-CLM/CLM | |
| ETH (CH) | COSMO-CLM/ROMS | |
| U. Castille (ES) | PROMES/U. Madrid ocean | IBM Power 6 - Intel Itanium |
| NHM Service (RS) | ECHAM5/MPIOM | Linux Fedora core 14 |
| CMC (CA) | GEM/NEMO | IBM Power5 |
| UQAM (CA) | GEMDM 3.3.2/NEMOv_2.3.0 | Linux PC |
| MM (MA) | ARPEGE-Climat_V5.1/NEMO-ORCA1 | IBM |
| INMT (TN) | ARPEGE-Climat_V5.1/NEMO-ORCA2 | IBM Regata Series |
| Oregon St U (USA) | PUMA/UVic | Linux cluster |
| Hawaii U (USA) | ECHAM4/POP | Linux cluster |
| JAMSTEC (JP) | ECHAM/OPA8.2 | NEC SX8 |
| JAMSTEC (JP) | ECHAM5_T106L31/NEMO_0.5 | NEC SX8 |
| Met.Nat.Center (CN) | GRAPES (201x161)/ECOM-si | IBM cluster |
| IAP (CN) | CREM(reg.)/POM2000 | SGI |
| IAP (CN) | ECHAM/MPIOM | Linux_x64 |
| CSIRO (AU) | ACCESS : UMv7.3/MOM4p1/CICE | SUN + SGI clusters |
| BoM (AU) | BAM3/ACOM2 | NEC SX6, SUN |
| BoM (AU) | TCLAPS/MOM | NEC SX-6 |
| U Tasmania (AU) | Data atm. model/MOM4 | SGI O3400 - Compaq5 |

**Table C.2:** Use of OASIS2: centres, coupled models and computing platforms

| Centre | Coupled model | Platform |
|---|---|---|
| CERFACS (FR) | ARPEGE3 OPA8.2/LIM 2deg | VPP5000 |
| CERFACS (FR) | ARPEGE3 OPA8.1 | VPP700 |
| CERFACS (FR) | ARPEGE2 OPAICE | CRAY C90 |
| IPSL (FR) | LMDz 96x71x19 - OPA/ORCA2 | VPP5000 |
| IPSL (FR) | LMDz 72x45x19 OPA/ORCA4 | VPP5000 |
| IPSL (FR) | LMDZ 120X90 - OPA ATL3 1/3 | |
| IPSL (FR) | LMDZ 120X90 - OPA ATL1 1 deg | |
| LODYC (FR) | IFS T195L31 OPA8.1 | |
| LODYC (FR) | ECHAM4 T30/T42 L14 OPA/ORCA2 | |
| Météo-France (FR) | ARPEGE medias OPAmed 1/8 deg | VPP5000 |
| Météo-France (FR) | ARPEGE3 - OPA 8.1 + Gelato | VPP5000 |
| Météo-France (FR) | ARPEGE 2 T31L19 - OPA8 TDH | CRAY J90 |
| ECMWF | IFS T63/T255 - E-HOPE 2deg/1deg | IBM Power 4 |
| ECMWF | IFS Cy23r4 T159L40 - E-HOPE 256L29 | VPP700 |
| ECMWF | IFS Cy23r4 T95L40 - E-HOPE 256L29 | VPP700 |
| ECMWF | IFS Cy15r8 T63L31 - E-HOPE 128L20 | VPP300 |
| MPI (DE) | ECHAM5 T42/L19 - C-HOPE T42+L20 | NEC-SX |
| MPI (DE) | PUMAT42/L19 - C-HOPE 2deg GIN | NEC-SX |
| MPI (DE) | EMAD - E-HOPE T42+L20 | CRAY C-90 |
| MPI (DE) | ECHAM5 T42/L19 - E-HOPE T42+L20 | NEC-SX |
| MPI (DE) | ECHAM4 T30/L19 - E-HOPE T42+L20 | CRAY T90 |
| MPI (DE) | ECHAM4 T30/L19 - E-HOPE T42+L20 | CRAY C90 |
| SMHI (SE) | RCA-HIRLAM (reg.) - RCO-OCCAM (reg.) | |
| CGAM (UK) | HadAM3 2.5x3.75 L20 - OPA/ORCA2 | T3E |
| SOC (UK) | Interm. Atm. GCM - OCCAM-Lite | |
| NOC (UK) | FORTE : IGCM3 (128 x 64) - MOMA (180 x 88) | |
| JPL (USA) | TRIDENT QTCM | Linux workstation |
| IRI (USA) | ECHAM4 - MOM3 | SGI Origin - IBM Power3 |
| JAMSTEC (JP) | ECHAM4 - OPA 8.2 | NEC SX5 |
| BMRC (AU) | BAM3 - ACOM2 | NEC SX6 |

# Bibliography


[Cassou et al 1998] Cassou, C., P. Noyret, E. Sevault, O. Thual, L. Terray, D. Beaucourt, and M. Imbard: Distributed Ocean-Atmosphere Modelling and Sensitivity to the Coupling Flux Precision: the CATHODe Project. *Monthly Weather Review*, 126, No 4: 1035-1053, 1998.

[Guilyardi et al 1995] Guilyardi, E., G. Madec, L. Terray, M. Déqué, M. Pontaud, M. Imbard, D. Stephenson, M.-A. Filiberti, D. Cariolle, P. Delecluse, and O. Thual. Simulation couplée océan-atmosphère de la variabilité du climat. *C.R. Acad. Sci. Paris*, t. 320, série IIa:683–690, 1995.

[Jacob et al 2005] Jacob, R., J. Larson, and E. Ong: MxN Communication and Parallel Interpolation in CCSM3 Using the Model Coupling Toolkit. *Int. J. High Perf. Comp. App.*, 19(3), 293-307 2005

[Jones 1999] Jones, P.: Conservative remapping: First- and second-order conservative remapping, Mon Weather Rev, 127, 2204-2210, 1999.

[Larson et al 2005] Larson, J., R. Jacob, and E. Ong: The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models. *Int. J. High Perf. Comp. App.*, 19(3), 277-292, 2005

[Noyret et al 1994] Noyret, P., E. Sevault, L. Terray and O. Thual. Ocean-atmosphere coupling. *Proceedings of the Fall Cray User Group (CUG) meeting*, 1994.

[Pontaud et al 1995] Pontaud, M., L. Terray, E. Guilyardi, E. Sevault, D. B. Stephenson, and O. Thual. Coupled ocean-atmosphere modelling - computing and scientific aspects. In *2nd UNAM-CRAY supercomputing conference, Numerical simulations in the environmental and earth sciences* Mexico-city, Mexico, 1995.

[Sevault et al 1995] Sevault, E., P. Noyret, and L. Terray. Clim 1.2 user guide and reference manual. *Technical Report TR/CGMC/95-47*, CERFACS, 1995.

[Terray and Thual 1995b] Terray, L. and O. Thual. Oasis: le couplage océan-atmosphère. *La Météorologie*, 10:50–61, 1995.

[Terray and Thual 1993] Terray, L. and O. Thual. Coupled ocean-atmosphere simulations. In *High Performance Computing in the Geosciences, proceedings of the Les Houches Workshop* F.X. Le Dimet Ed., Kluwer Academic Publishers B.V, 1993.

[Terray et al 1995] Terray, L., E. Sevault, E. Guilyardi and O. Thual OASIS 2.0 Ocean Atmosphere Sea Ice Soil User's Guide and Reference Manual *Technical Report TR/CGMC/95-46*, CERFACS, 1995.

[Terray et al 1995b] Terray, L. O. Thual, S. Belamari, M. Déqué, P. Dandin, C. Lévy, and P. Delecluse. Climatology and interannual variability simulated by the arpege-opa model. *Climate Dynamics*, 11:487–505, 1995

[Terray et al 1999] Terray, L., S. Valcke and A. Piacentini: OASIS 2.3 Ocean Atmosphere Sea Ice Soil, User's Guide and Reference Manual, *Technical Report TR/CMGC/99-37*, CERFACS, Toulouse, France, 1999.

[Valcke et al 2004] Valcke, S., A. Caubel, R. Vogelsang, and D. Declat: OASIS3 User's Guide (oasis3_prism_2-4), *PRISM Report No 2, 5th Ed.*, CERFACS, Toulouse, France, 2004.

[Valcke et al 2003] Valcke, S., A. Caubel, D. Declat and L. Terray: OASIS3 Ocean Atmosphere Sea Ice Soil User's Guide, *Technical Report TR/CMGC/03-69*, CERFACS, Toulouse, France, 2003.

[Valcke et al 2000] Valcke, S., L. Terray and A. Piacentini: OASIS 2.4 Ocean Atmosphere Sea Ice Soil, User's Guide and Reference Manual, *Technical Report TR/CMGC/00-10*, CERFACS, Toulouse, France, 2000.