PRISM Support Initiative
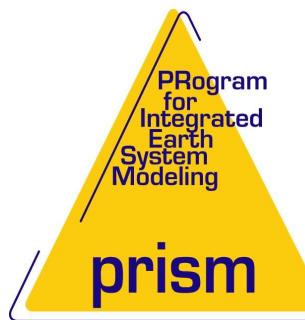
PRogram
for
Integrated
Earth
System
Modeling

prism

Adaptation of OASIS4 and TOYOA4
to the PRISM Standard directory structure
and Compile Environment

J. Ghattas, S. Valcke
CNRS, CERFACS

PSI Internal Report 2

1 Overview

This document first describes, in section 2, the directory structure of the OASIS4 coupler and its toy coupled model "TOYOA4", recently adapted to the PRISM Standard directory structure version 2.4. OASIS4 and TOYOA4 are currently developed using that directory structure on CERFACS CVS server `alter`. Section 3 provides detailed instructions on how to compile OASIS4 and TOYOA4, using or not the PRISM Standard Compile Environment (SCE). Section 4 gives details on how to run the toy coupled model TOYOA4. OASIS4 was previously developed using another directory structure on the `bedano` CVS server in CSCS (Switzerland); the modifications that were included in OASIS4 to migrate from the old directory structure to the new one are described in more detail in Appendix 1.1.

2 New source code directory structure
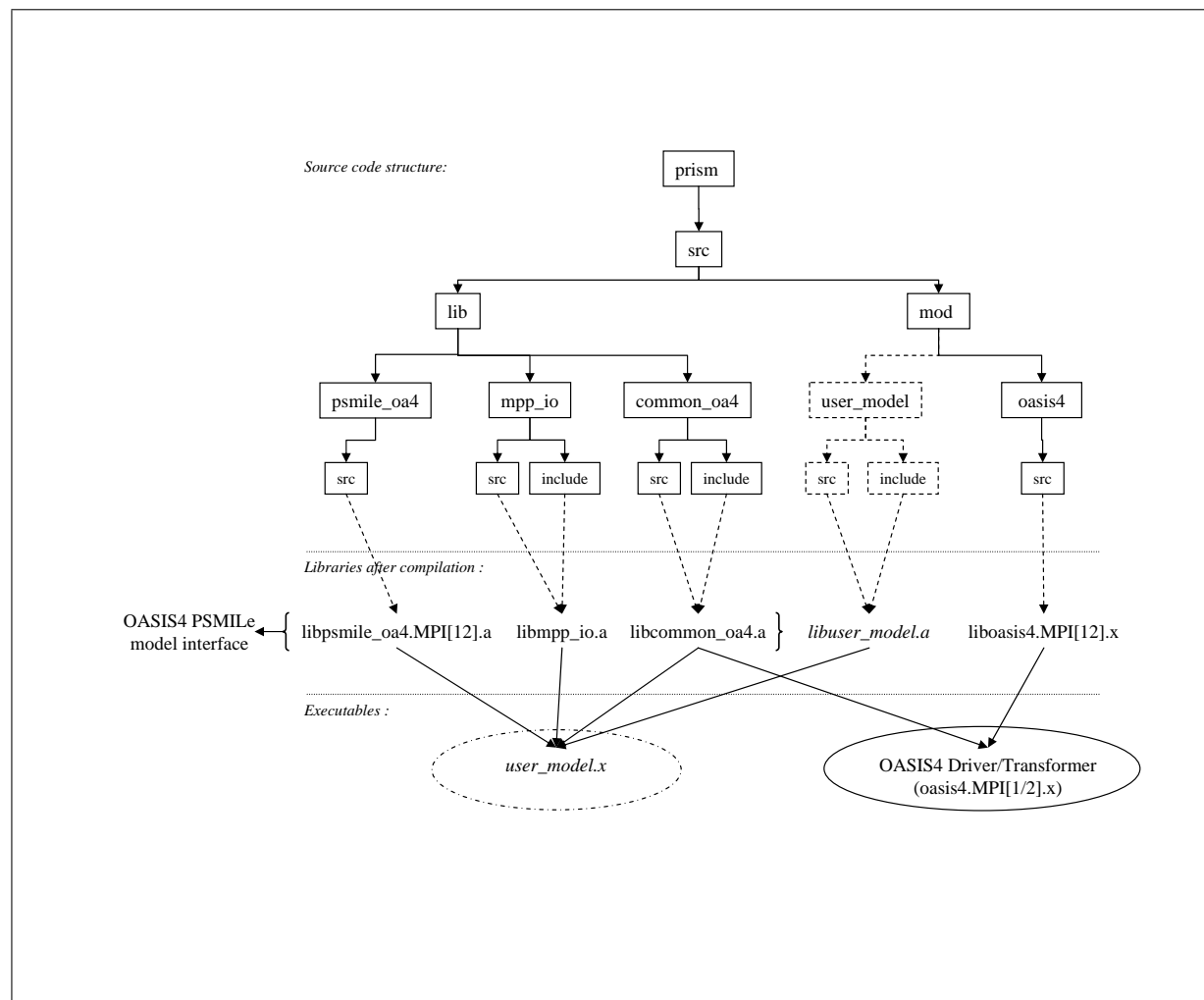
2.1 OASIS4 sources

Figure 1: New source directory structure

OASIS4 sources were divided into three directories under `prism/src/lib/` and one directory `prism/src/mod/oasis4/`. In this new structure, only a relatively small library *common_oa4* is used by both the OASIS4 Driver/Transformer executable, which at run time performs the interpolations, and by the OASIS4 PSMILe model interface, which needs to be linked to the component models for I/O and coupling exchanges. The different directories are:

- `prism/src/lib/common_oa4/`: this directory contains sources that are used both by the Driver/Transformer and the PSMILe model interface. After compilation, these sources becomes the *libcommon_oa4.a* library.

- `prism/src/lib/mpp_io/`: this directory contains the sources of the GFDL I/O library (1). After compilation, these sources form the library *libmpp_io.a*. Compiling and linking of this library to a component model is not mandatory if the PSMIle I/O functionality is not used (see compilation details in sections 3.1 and 3.2 below).

- `prism/src/lib/psmile_oa4/`: this directory contains the sources that form the main part of PSMILe model interface and become, after compilation the library *libpsmile_oa4.a*.

- `prism/src/mod/oasis4/`: this directory contains the main part of OASIS4 Driver/Transformer sources. Linked with the library *libcommon_oa4.a*, these sources form, after compilation, the OASIS4 Driver/Transformer executable named `oasis4.MPI1.x` or `oasis4.MPI2.x` (according to the choice of MPI1 or MPI2 done at compilation, see sections 3.1 and 3.2 below for details).
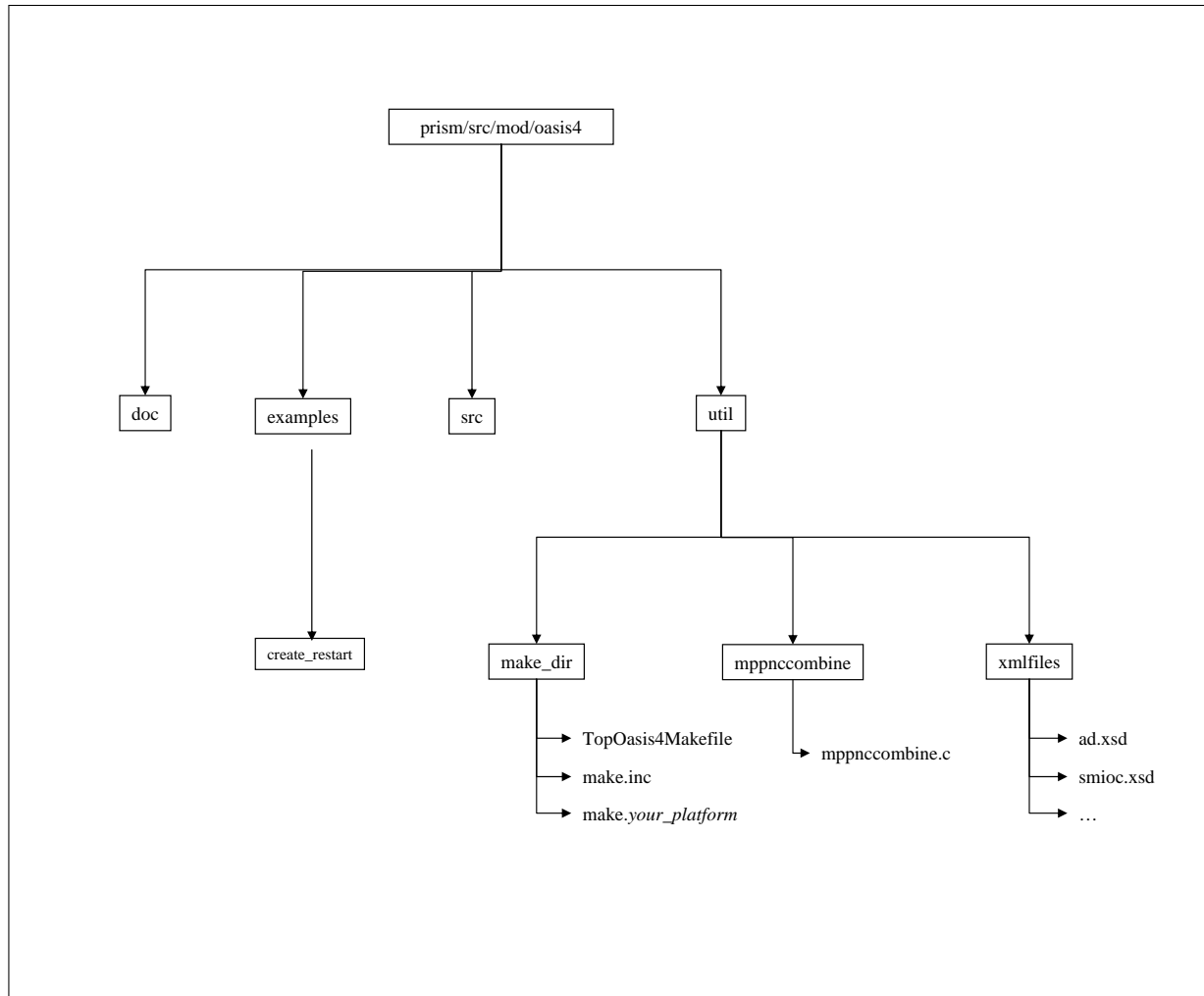
2.2 Other OASIS4 directories

Figure 2: Directories in prism/src/mod/oasis4

In the `prism/src/mod/oasis4` directory, three more directories `/doc`, `/examples` and `/util` are found:

- `/doc` contains OASIS4 documentation.

- `/examples` and its sub-directory `/create_restart` contains programs which provide example

on how to use the PSMILe `prism_put_restart` routine to create an OASIS4 coupling restart file (see the README therein).

- `/util` contains directory `/make_dir` into which a top makefile and platform dependent header files for compiling OASIS4 without using the SCE can be found (see section 3.1), directory `/xmlfiles` which contains the SCHEMAs of the different XML files used with OASIS4 (see chapter 5 of OASIS4 documentation), and directory `mppnccombine` which contains a program, `mppnccombine.nc`, which may be used to join together NetCDF data files representing a decomposed domain into a unified NetCDF file.

2.3 The toy coupled model TOYOA4 directory structure
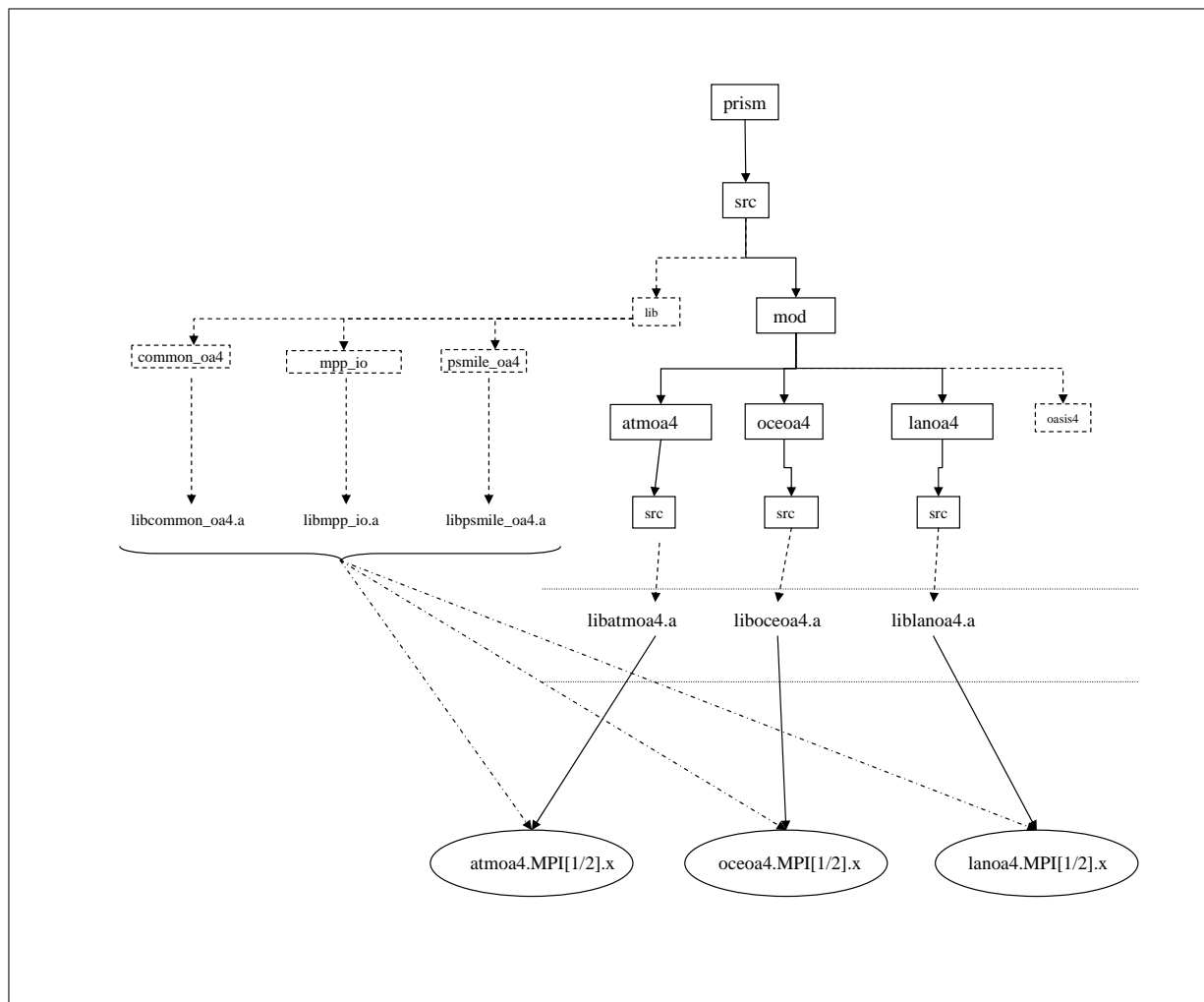

Figure 3: Directory structure for TOYOA4

TOYOA4 provides a practical example on how to use OASIS4 to couple 3 component models. The sources for each toy component model are included in the PRISM directory structure with one directory for each component, respectively in `/prism/src/mod/atmoa4`, `/oceoa4`, and `/lanoa4`. Section 3 details how to compile those three toy component models while section 4 explains how to run the resulting toy coupled model TOYOA4.

3 Compiling OASIS4 and TOYOA4

Compiling can be done using either the PRISM Standard Compile Environment (SCE) (see section 3.2) or using a top makefile `TopMakefileOasis4` and platform dependent header files (see section 3.1). For

both methods, the same low-level makefiles in each source directory are used. During compilation, a new directory branch is created /prism/*arch*, where *arch* is the name of the compiling platform architecture (e.g. *Linux*). After successful compilation, resulting executables are found in /prism/*arch*/bin, libraries in /prism/*arch*/lib and object and module files in /prism/*arch*/build.

3.1 Compilation with TopMakefileOasis4

Compiling OASIS4 and TOYOA4 using the top makefile TopMakefileOasis4 can be done in directory prism/src/mod/oasis4/util/make_dir. TopMakefileOasis4 must be completed with a header file make.*your_platform* specific to the compiling platform used and specified in prism/src/mod/oasis4/util/make_dir/make.inc. One of the files make.pgi_cerfacs, make.sx_frontend or make.aix can by used as a template. The root of the prism tree can be anywere and must be set in the variable PRISMHOME in the make.*your_platform* file. The choice of MPI1 or MPI2 is also done in the make.*your_platform* file (see CHAN therein).

The following commands are available:

- make -f TopMakefileOasis4

  compiles OASIS4 libraries *common_oa4*, *psmile_oa4* and *mpp_io* and creates OASIS4 Driver/Transformer executable oasis4.MPI[1/2].x ;

- make toyoa4 -f TopMakefileOasis4

  compiles OASIS4 libraries as above and creates OASIS4 and TOYOA4 executables oasis4.MPI[1/2].x, atmoa4.MPI[1/2].x, oceoa4.MPI[1/2].x and lanoa4.MPI[1/2].x ;

- make help -f TopMakefileOasis4

  displays help information ;

- make clean -f TopMakefileOasis4:

  cleans OASIS4 and TOYOA4 compiled files, but not the libraries ;

- make realclean -f TopMakefileOasis4:

  cleans OASIS4 and TOYOA4 compiled files including libraries.

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

For not compiling the mpp_io library, the variable LIBMPP must be left undefined in the file make.*your_platform*; in this case, the top makefile activates the CPP key key_noIO and only empty mpp_io files are compiled.

3.2 Compilation using the PRISM Standard Compiling Environment (SCE)

The PRISM Standard Compiling Environment (SCE) has been adapted for OASIS4. These modifications are available on CERFACS CVS server alter and will also be included in the next official release of the SCE on the new PRISM Subversion server at DKRZ in Hamburg.

Scripts and include files for the SCE are found in directory branch prism/util/compile (see figure 4). The toy model TOYOA4 using OASIS4 has been successfully compiled and run for the 3 platforms currently included in the SCE available from CERFACS CVS server: the NEC SX6 at DKRZ (*nodename* = ds, see /frames/include_ds), the IBM power4 at ECMWF (*nodename* = hpc, see /frames/include_hpc) and the Linux PC at CERFACS (*nodename* = kullen, see /frames/include_kullen). Details about including a new platform (i.e. a new *nodename*) in the SCE can be found in the SCE documentation (2).

For compiling OASIS4 and TOYOA4 within SCE, the compilation scripts first have to be created by using Create_COMP_cpl_models.ksh in prism/util/compile/frames:

./Create_COMP_cpl_models.ksh toyoa4 [*expid*] [*nodename*] [MPI1 *or* MPI2]

where the last 3 arguments are optional.

This will create 4 model compilation scripts:

- `prism/src/mod/atmoa4/COMP_atmoa4_`*expid.nodename*
- `prism/src/mod/lanoa4/COMP_lanoa4_`*expid.nodename*
- `prism/src/mod/oasis4/COMP_oasis4_`*expid.nodename*
- `prism/src/mod/oceoa4/COMP_oceoa4_`*expid.nodename*

and 1 library compilation script:

- `prism/util/COMP_libs.`*node_name.*

Then each model compilation script has to be executed in it directory; the library compilation script is executed automatically by each of the model compilation script.

During compilation, log and error messages are written into files with suffix .log and .err in the same directory than the compilation script. Log and error messages after compilation of the libraries are found in `prism/util/COMP_libs.log` and `COMP_libs.err`.

For compiling without mpp_io library, the variable `use_key_noIO` has to be changed to "yes" in the compile scripts for atmoa4, oceoa4 and lanoa4; in that case, only empty mpp_io files are compiled.
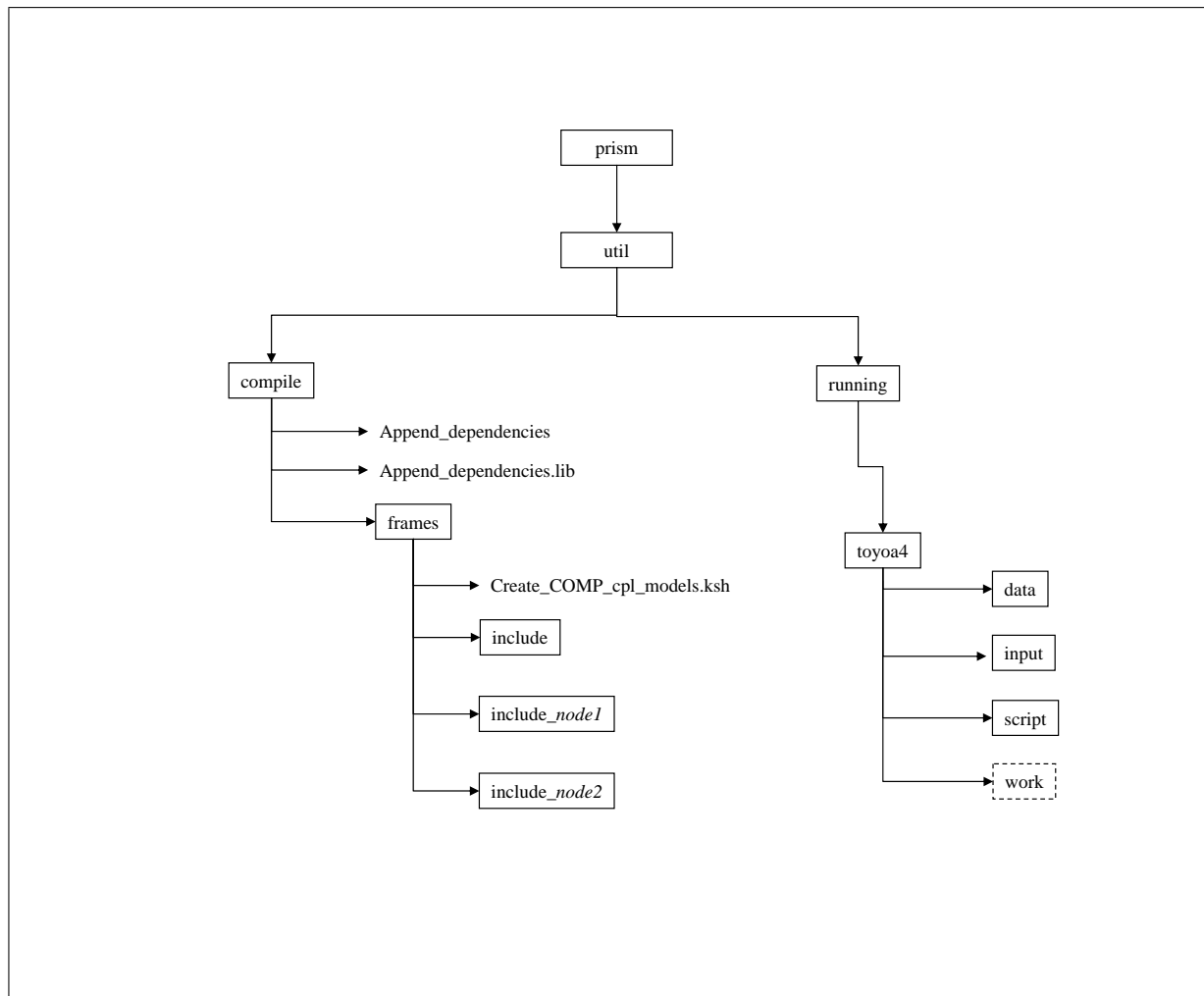
4 Running TOYOA4

Figure 4: Directories in prism/util

Input files, data and script for running TOYOA4 are found in prism/util/running/toyoa4, see figure 4. Note that TOYOA4 has not been adapted to PRISM Standard Running Environment.

NetCDF data files needed for running TOYOA4 are found in directory `/data`. The description and configuration XML files are found in directory `/input`. Running can be done with a run script `run_toyoa4` in directory `/script` which first will create the working directory `/work`; all files and executables needed for running are first copied into this working directory. The run script `run_toyoa4` was run on three platforms, Linux at CERFAXS, SX6 at DKRZ and IBM power4 at ECMWF, using MPI1 (which means that OASIS4 and 3 component model executables are started in the script. The script `run_toyoa4` is an example of running TOYOA4 and can be modified by the user for his/her platform.
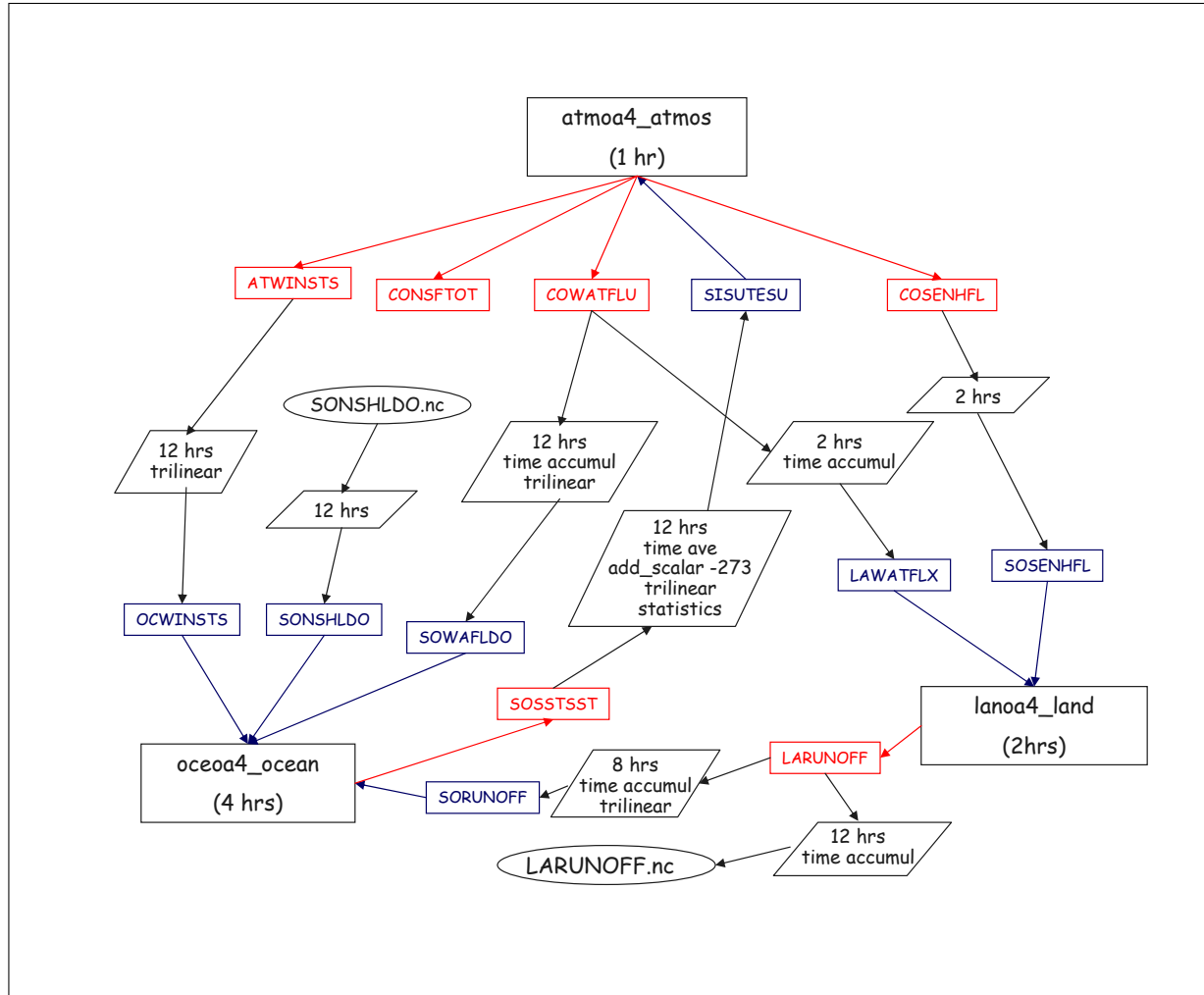

Figure 5: TOYOA4 toy coupled model coupling and I/O configuration

Figure 5 illustrates the coupling and I/O exchanges occuring between the 3 toy component models atmoa4, oceoa4, and lanoa4.

Both atmoa4 and lanoa4 work on a T31 Gaussian grid, but their parallel partitioning is a function of their number of processes which can be different. The third model, oceoa4, is not parallel and uses a a real ocean model cartesian, stretched and rotated grid of 182X149 grid points.

All coupling and I/O fields are scalar fields. The model atmoa4 declares 1 input field `SISUTESU`, and 4 output field `CONSFTOT, COSENHFL, COWATFLU, ATWINSTS` as is listed in its PMIOD file `atmoa4_atmos_pmiod.xml`. The model lanoa4 declares 2 input fields `LAWATFLX` and `SOSENHFL`, and 1 output field `LARUNOFF` as is listed in its PMIOD file `lanoa4_land_pmiod.xml`. The model oceoa4 declares 4 input fields `SONSHLDO, SOWAFLDO, SORUNOFF` and `OCWINSTS`, and 1 output field `SOSSTSST`.

At run-time, the OASIS4 Driver/Transformer and the PSMILe model interface linked to the component

models act according to the specifications written by the user in the configuration SMIOC XML files.

In the atmoa4 SMIOC file `atmoa4_atmos_smioc.xml`, it is specified that `ATWINSTS` will be sent to oceoa4, `COSENHFL` to lanoa4, `COWATFLU` both to oceoa4 and lanoa4, while `CONSFTOT` is not sent at all; it is also specified that `SISUTESU` will come from oceoa4. The lanoa4 SMIOC file `lanoa4_land_smioc.xml` specifies that `LARUNOFF` will both go to oceoa4 and be written to a file `LARUNOFF.nc` and that `LAWATFLX` and `SOSENHFL` will be received from atmoa4. Finally, in the oceoa4 SMIOC file `oceoa4_ocean_smioc.xml`, it is specified that `OCWINSTS` and `SOWAFLDO` will be received from atmoa4, `SORUNOFF` from lanoa4, while `SONSHLDO` will be read from a file `SONSHLDO.nc`; `SOSSTSST` will be sent to atmoa4.

Different operations are performed by the PSMILe model interface on the coupling or I/O fields such as statistics, time accumulation time averaging, as specified in the SMIOC files. The exchanges of the coupling fields between atmoa4 and lanoa4 (and vice-versa) are direct, involving possibly some repartitioning if their parallel partitioning are different. As atmoa4 and oceoa4 do not have the same grid, their exchanges of coupling fields go through the Transformer (not illustrated on figure 5) where a linear interpolation is performed. The different coupling and I/O periods are also specified in the different SMIOC files.

TOYOA4 also illustrates the use of a coupling restart file for field `COSENHFL` for which a positive lag of 2 is defined. The first time TOYOA4 is run, the variable `run` should be set to `start` in `run_toyoa4`. In that case, the file `scc.xml.start` is copied in `scc.xml` and used, TOYOA4 is run for 3 days starting January 1st 2000, and the first field `COSENHFL` received by lanoa4 comes from the restart file `COSENHFL_toyatm_atmos_rst.2000-01-01T00_00_00.nc`; at the end of the run, the restart file for the next run, `COSENHFL_toyatm_atmos_rst.2000-01-04T00_00_00.nc`, is created by the last call to prism_put for `COSENHFL` in atmoa4. A next run of 3 days starting January 4th 2000 can then be run by changing `run=restart` in `run_toyoa4` and executing `run_toyoa4` again.

1 Appendix

1.1 Migrating from the previous directory structure to the new one

OASIS4 is currently being developed using the directory structure described in section 2.1 on CERFACS CVS server `alter` but was previously using another directory structure on the `bedano` CVS server in CSCS (Switzerland). The modifications that were included in OASIS4 to migrate from the old directory structure to the new one are described here.

The sources that were previously in directory `PRISM_Cpl/source/driver` are now found in `prism/src/mod/oasis4/src`. Sources that were in `PRISM_Cpl/source/del` and `/io` are now mainly in `prism/src/lib/psmile_oa4` with some files in `prism/src/lib/common_oa4`. Files from the old directory `PRISM_Cpl/source/xml` are now in `prism/src/lib/common_oa4` with only one file in `/psmile_oa4`. All files from the old `PRISM_Cpl/source/mpp_io` are in the new directory `prism/src/lib/mpp_io`; those sources were slightly modified and are now common for OASIS3 and OASIS4. Files from `PRISM_Cpl/include` are now in `prism/src/lib/common_oa4` and `psmile_oa4`. Table 1 lists all sources form the new `prism/src/lib/common_oa4` directory and their origin in the old directory structure.

Files in prism/src/lib/common_oa4/src/	*Original directory PRISM_Cpl/*
prism_constants.F90	include/
psmile_common.F90	*new file*
psmile_abort.F90	source/del/
psmile_char2buf.F90	source/del/
psmile_error_common.F90	*new file*
psmile_flushstd.c	source/del/
psmile_int2char.F90	source/del/
psmile_redirstdout.c	source/del/
psmile_scc.F90	source/xml/
psmile_smioc.F90	source/xml/
sasa_c_f90.c	source/xml/
sasa_c_xml.c	source/xml/
Files in prism/src/lib/common_oa4/include/	*Original directory PRISM_Cpl/*
PSMILe_f2c.h	include/
prism.inc	include/
psmile.inc	include/
sasa_c_f90.h	source/xml/
sasa_c_xml.h	source/xml/

Table 1: Contents of the new directory `prism/src/lib/common_oa4`

The next section describes in details how the OASIS4 sources from the previous directory structure were modified in order to include them in the new directory structure conforming to the PRISM standard.

1.2 Modifications of OASIS4 sources

Tables 2, 3, 4, 5 list files respectively in directories `prism/src/mod/oasis4`, `prism/src/lib/common_oa4`, `/psmile_oa4`, and `/mpp_io` that were changed compared to their previous version in the old directory structure. Some of the modifications are explained in more detail here below.

Table 6 and 7 details the new localisation of the sources that were respectively in `PRISM_Cpl/source/xml` `PRISM_Cpl/include/` directories.

Original file name	New file name(if changed)	Action
prismdrv.F90	prismdrv.F90 prismdrv_constants.F90	Contained 2 modules; split into 2 files; Changed *USE psmile* to *USE psmile_common*
prismdrv_trs_finalize.F90		Removed
prismdrv_def_mpi_comm.F90 prismdrv_finalize.F90 prismdrv_get_smioc_file_name.F90 prismdrv_init.F90 prismdrv_init_appl.F90 prismdrv_set_scc_info.F90 prismdrv_trs_bcast2trs.F90 prismdrv_trs_get_epio_handle.F90 prismdrv_trs_interp.F90 prismdrv_trs_loop.F90 prismdrv_trs_mind_dble.F90 prismdrv_trs_mind_int.F90 prismdrv_trs_mind_real.F90 prismdrv_trs_set_neighbors3d.F90 prismdrv_trs_set_src_epio_dble.F90 prismdrv_trs_set_src_epio_real.F90 prismdrv_trs_set_tgt_epio_dble.F90 prismdrv_trs_set_tgt_epio_real.F90 prismdrv_trs_set_triple_links.F90 prismdrv_trs_target_dble.F90 prismdrv_trs_target_int.F90 prismdrv_trs_target_real.F90	prismtrs_bcast2trs.F90 prismtrs_get_epio_handle.F90 prismtrs_interp.F90 prismtrs_loop.F90 prismtrs_mind_dble.F90 prismtrs_mind_int.F90 prismtrs_mind_real.F90 prismtrs_set_neighbors3d.F90 prismtrs_set_src_epio_dble.F90 prismtrs_set_src_epio_real.F90 prismtrs_set_tgt_epio_dble.F90 prismtrs_set_tgt_epio_real.F90 prismtrs_set_triple_links.F90 prismtrs_target_dble.F90 prismtrs_target_int.F90 prismtrs_target_real.F90	Changed *USE psmile* to *USE psmile_common*; Changed *CALL psmile_error* to *CALL psmile_error_common*.
prismdrv_main.F90 prismdrv_trs_apply_grads.F90 prismdrv_trs_apply_weights.F90 prismdrv_trs_bicubic_grad_2d.F90 prismdrv_trs_bicubic_weight_2d.F90 prismdrv_trs_bilinear_weight_2d.F90 prismdrv_trs_bilinear_weight_2d1d.F90 prismdrv_trs_distwght_weight_2d.F90 prismdrv_trs_distwght_weight_2d1d.F90 prismdrv_trs_distwght_weight_3d.F90 prismdrv_trs_gauswght_weight_2d.F90 prismdrv_trs_gauswght_weight_3d.F90 prismdrv_trs_get_trans_rank.F90 prismdrv_trs_linear_weight_for_2d1d.F90 prismdrv_trs_main.F90 prismdrv_trs_set_epio_trans.F90 prismdrv_trs_trilinear_weight.F90 psmile_spawn_child.F90	prismtrs_apply_grads.F90 prismtrs_apply_weights.F90 prismtrs_bicubic_grad_2d.F90 prismtrs_bicubic_weight_2d.F90 prismtrs_bilinear_weight_2d.F90 prismtrs_bilinear_weight_2d1d.F90 prismtrs_distwght_weight_2d.F90 prismtrs_distwght_weight_2d1d.F90 prismtrs_distwght_weight_3d.F90 prismtrs_gauswght_weight_2d.F90 prismtrs_gauswght_weight_3d.F90 prismtrs_get_trans_rank.F90 prismtrs_linear_weight_for_2d1d.F90 prismtrs_main.F90 prismtrs_set_epio_trans.F90 prismtrs_trilinear_weight.F90	Changed *USE psmile* to *USE psmile_common*
prismdrv_finalize_smioc_struct.F90 prismdrv_init_smioc_struct.F90 prismdrv_set_smioc_info.F90		Changed *CALL psmile_error* to *CALL psmile_error_common*

Table 2: Modified files in prism/src/mod/oasis4/

Original file name	New file name(if changed)	Action
prismf.F90	(psmile_oa4/src/prism.F90) prism_constants.F90	Contained 2 modules and was therefore split into 2 files. prism.F90 was moved to prism/src/lib/psmile_oa4 library
psmile.F90	psmile_common.F90 (psmile_oa4/src/psmile.F90)	See the text below
psmile_scc.F90 psmile_smioc.F90 psmile_char2buf.F90 psmile_int2char.F90		Changed *USE psmile* to *USE psmile_common*
psmile_abort.F90		Changed *USE psmile* to *USE psmile_common*
	psmile_error_common.F90	Created based on psmile_error.F90

Table 3: Modified files in prism/src/lib/common_oa4/

One module per file with corresponding name

In the PRISM standards, only one module per file is allowed and the file name must correspond to the module name.

According to this rule, the following modifications were done:

- Routine `PRISM_Cpl/include/prismf.F90` which contained two modules, `prism` and `prism_constants` was split in two files, `prism/src/lib/psmile_oa4/prism.F90` and `prism/src/lib/common_oa4/prism_constants.F90`.

- Files `mpp_domains_mod.F90`, `mpp_io_mod.F90`, and `mpp_mod.F90` previously in `PRISM_Cpl/source/mpp_io` were renamed `mpp_domains_mod_oa.F90`, `mpp_io_mod_oa.F90`, and `mpp_mod_oa.F90` in `prism/src/lib/mpp_io/src`.

- The file `PRISM_Cpl/source/io/psmile_io_get_attr.F90` was renamed `psmile_io_get.F90` in `prism/src/lib/psmile_oa4/src`.

- The file `PRISM_Cpl/source/driver/prismdrv.F90` was split into `prismdrv.F90` and `prismdrv_constants.F90` in `prism/src/mod/oasis4/src`.

Module PRISM_Cpl/include/psmile.F90

The module in `PRISM_Cpl/include/psmile.F90` was originally used by both the OASIS4 Driver/-Transformer executable and the PSMILe model interface . This module was using the mpp_io library which was then necessarily linked to the OASIS4 Driver/Transformer executable. As this library is in fact not needed by OASIS4 Driver/Transformer executable, the old file `psmile.F90` was split into `prism/src/lib/psmile_oa4/src/psmile.F90`, which contains the part from the original module used only in *libpsmile_oa4.a*, and into `prism/src/lib/common_oa4/src/psmile_common.F90`, which contains the remaining part from the original module and does not use the mpp_io library.

The module *psmile_constants* which was included in the original `psmile.F90` was removed and its contents was added into the new `prism/src/lib/common_oa4/src/psmile_common.F90`.

Routine PRISM_Cpl/source/del/psmile_error.F90

The routine `PRISM_Cpl/source/del/psmile_error.F90` was divided into `prism/src/lib/common_oa4/src/psmile_error_common.F90` and `prism/src/lib/psmile_oa4/src/psmile_error.F90`.

The routine was originally called both in the OASIS4 Driver/Transformer and by the PSMILe model interface. A part of this routine was using structures defined in module `psmile`. The new file

Original file name	New file name(if changed)	Action
PRISM_Cpl/include/psmile.F90	psmile.F90	Contains remaining parts of original module *psmile* that are not in module psmile_common.F90. Added *USE psmile_common*.
mpi-calendar.F90	prism_calendar.F90	Changed name to corresponding module name. Removed code wrapped in cpp key __test_mpi_calendar: program calendar-test, module prism and module psmile.
psmile_io_get_attr.F90	psmile_io_get.F90	Changed file name to corresponding module name
psmile_reduce.F90		Removed code wrapped in cpp key test_reduce: program reduce and module psmile
psmile_neigh_nearx_sub_irr_dble.F90 psmile_neigh_nearx_sub_irr_real.F90 psmile_neigh_nearx_sub_reg_dble.F90 psmile_neigh_nearx_sub_reg_real.F90 psmile_info_trs_loc_3d_reg_dble.F90 psmile_info_trs_loc_3d_reg_real.F90 psmile_info_trs_loc_irreg2_dble.F90 psmile_info_trs_loc_irreg2_real.F90 psmile_info_trs_loc_gauss2_dble.F90 psmile_info_trs_loc_gauss2_real.F90 psmile_info_trs_locs_3d_dble.F90 psmile_info_trs_locs_3d_real.F90 prism_init.F90 psmile_def_mpi_compcomm.F90		Line break; line becomes to long due to preprocessor substitution of __FILE__
psmile_bsend.c psmile_clock.F90		Changed syntax for inclusion of mpif.h
PRISM_Cpl/source/xml/ psmile_smioc_init.F90	psmile_smioc_init.F90	Moved from source/xml, added *USE psmile*
psmile_error.F90		Modified and added call psmile_error_common
psmile_mg_prev_final_3d_xxxx.F90 prism_set_angle_xxxx.F90 prism_set_scalefactor_xxxx.F90 prism_set_scfact_xxxx.F90 prism_set_subgrid_xxxx.F90		Removed

Table 4: Modified files in prism/src/lib/psmile_oa4/

Original file name	New file name(if changed)	Action
mpp_io_mod.F90 mpp_domains_mod.F90 mpp_mod.F90	mpp_io_mod_oa.F90 mpp_domains_mod_oa.F90 mpp_mod_oa.F90	Changed file names

Table 5: Modified files in prism/src/lib/mpp_io/src/

Original file name	New file location
psmile_smioc_finalize.F90 psmile_smioc_init.F90	prism/src/lib/psmile_oa4/src/
psmile_scc.F90 psmile_smioc.F90 sasa_c_f90.c sasa_c_xml.c	prism/src/lib/common_oa4/src/
sasa_c_f90.h sasa_c_xml.h	prism/src/lib/common_oa4/include/

Table 6: Table of new locations for all files in PRISM_Cpl/source/xml/

Original file name	New file location
ipsl-calendar.F90	No longer used
mpi-calendar.F90	prism/src/lib/psmile_oa4/src/prism_calendar.F90
prismf.F90	Split into prism/src/lib/common_oa4/src/prism.F90 and prism/src/lib/common_oa4/src/prism_constants.F90
psmile.F90	Split into prism/src/lib/psmile_oa4/src/psmile.F90 and prism/src/lib/common_oa4/src/psmile_common.F90
prism.inc PSMILe_f2c.h psmile.inc	prism/src/lib/common_oa4/include/

Table 7: Table of new locations for all files in PRISM_Cpl/include/

`psmile_error_common.F90` contains the parts independent of the module `psmile` and the remaining parts are left in `psmile_error.F90`.

Module prism_calendar and test programs

In the old directory `PRISM_Cpl/include/`, two files, `mpi-calendar.F90` and `ipsl-calendar.F90` were containing a module *prism_calendar*. In the new structure `mpi-calendar.F90` was renamed `prism_calendar.F90` in `prism/src/lib/psmile_oa4/src` and the file `ipsl-calendar.F90` was removed.

In `psmile_reduce.F90` and `prism_calendar.F90` (originally `mpi-calendar.F90`) there were test programs and extra modules wrapped in the CPP keys, which were removed to avoid problems with duplicated modules for the search of prerequisites which does not take CPP keys into account.

Preprocessor include instructions

The syntax for inclusion of system files in the SCE has to be a preprocessor include. For example, the mpif.h and mpi.h system files have to included using the syntax `#include <mpif.h>` and `#include <mpi.h>`. Following this rule ensures that system files are not automatically added system files as prerequisites in the Makefiles when using the SCE tool Append_dependencies. The files `psmile_clock.F90`, `psmile_common.F90`, `psmile_bsend.c` were modified to follow this rule.

When compiling it is important to make sure the correct mpif.h is used. The mpif90 compiler does not on all platforms necessarily take in count the MPI-environment for the preprocessing. This problem is circumvented in the SCE by explicitly adding the full path to the compiler command.

Other modifications

- In some files, the variable __FILE__, which corresponds to the full path and filename, became to long after preprocessing and so the line exceeded the maximum number of characters. A line break was inserted in these cases (see table 4).

- The file `mod_kinds_model.F90` in `PRISM_Cpl/source/mpp_io` was renamed `mod_kinds_mpp.F90` in `prism/src/lib/mpp_io/src` to avoid conflict with another routine named `mod_kinds_model.F90` in OASIS3.

- The routine `PRISM_Cpl/source/mpp_io/mppnccombine.nc` was moved to `prism/src/mod/oasis4/util/mppnccombine`.

- All files starting with prefix `prismdrv_trs_` in `PRISM_Cpl/source/driver` were renamed with prefix `prismtrs_` in `prism/src/mod/oasis4/src` to be coherent with the routine name inside the file.

- The files not originally compiled or used were removed. These files are : prismdrv_trs_finalize.F90, psmile_mg_prev_final_3d_xxxx.F90, prism_set_angle_xxxx.F90, prism_set_scalefactor_xxxx.F90, prism_set_scfact_xxxx.F90, prism_set_subgrid_xxxx.F90.

Bibliography

[1] http://www.gfdl.gov/~ vb/mpp_io.html

[2] Legutke, S., V. Gayler, 2005: The PRISM Standard Compile Environment Handbook. PRISM Report Series No 4.