
norESM documentation Documentation

Anne Fouilloux

Mar 31, 2023

CONTENT:

1	Introduction	3
1.1	Obtaining a version of the model	3
1.2	Running / Configuring the model	3
1.3	Develop the model	4
1.4	Archive model results	5
1.5	CMIP5 archive of NorESM results	5
1.6	Share model results	5
1.7	Past and ongoing work	5
1.8	Resources	5
1.9	References	6
2	Newbies guide to running NorESM	7
2.1	Go to scripts directory	7
2.2	Create the case	7
2.3	Configure the case	7
2.4	Build the case	8
2.5	Run the case	8
2.6	Example	8
2.7	Important files	8
2.8	More information	9
3	Advanced configuration (NorESM1)	11
3.1	Understanding compsets	11
3.2	AMIP type simulations	11
3.3	Running with offline aerosol interaction	11
3.4	Use of look-up tables for aerosol optics and activation to cloud droplets	12
3.5	Use of chemistry	12
4	Fluxes crossing boundaries	13
5	Setting up CAM on your own linux PC	15
5.1	Obtain CAM 5.3 source code	15
5.2	Compile the netcdf libraries using gfortran	15
5.3	Compile CAM on your Linux PC	16
5.4	Including cam-oslo code	17
5.5	Debug the model using ddd or gdb	17
5.6	View results	18
5.7	Configure your case	18
6	Model Diagnostic Tools	19
6.1	NorESM Diagnostic Packages	19

6.2	Tracer Budget terms	22
6.3	NCL Model Version Comparison package (Alf K)	24
6.4	Cloud water mass and number analysis (budgets)	25
6.5	Automatic AEROCOM analysis	25
6.6	ESMval CIS JASMIN platform and tools	26
6.7	Post analysis and workup of CAM diagnostics output tables	26
7	Using the issue tracker	27
7.1	Why do we need an issue tracker	27
7.2	Log in and check what is there	27
7.3	Create issues	27
7.4	Priority definition for NorESM	27
7.5	Which issues should we add to different milestones	28
7.6	Working	28
7.7	Connection to version control system	28
8	Test list for NorESM	29
8.1	Restart test	29
8.2	Bit-identical meteorology for “technical only” code changes	29
8.3	Physical tests (early development)	29
9	Obtain a copy of the model (using git)	31
10	Verify that you have the correct checkout	33
11	Modify files	35
12	Get modifications from github	37
13	Send modifications to github	39
14	If you don’t understand and want to get back to svn	41
15	SVN - Best Practice/FAQ	43
16	Branches	45
16.1	What is a branch?	45
16.2	When should I work on a branch?	46
16.3	How should I name the branch?	46
16.4	A note on branch / tag naming	47
16.5	How and when should I merge from trunk to my branch?	47
16.6	How can I merge my branch back into the trunk?	48
16.7	What is a tag?	48
16.8	When should I create a tag?	48
16.9	What tags exist?	49
16.10	How should I name the tag?	49
16.11	Tagging noresm0 and noresm1 branches?	50
16.12	Case 1	50
16.13	Case 2	50
16.14	Case 3	50
16.15	What should be stated in the commit message?	51
16.16	What should be stated in the copy message of a branch?	51
16.17	What should be stated in the copy message of a tag?	52
17	Subversion how-to for NorStore	53
17.1	How it works	53

17.2	Prerequisites	53
17.3	Start/stop server	53
17.4	Creating a new repository	54
17.5	Customizing access rights	54
17.6	Remote access	55
18	Uncertain parameters (which can be discussed) in the aerosol model	57
19	NorStore Tape Storage	59
19.1	Basic use	59
19.2	Advanced use	59
20	NorStore Research Data Archive: Guidelines for ingestion of NorESM output	63
20.1	Metadata recommendations	63
20.2	Metadata example	68
21	Simple online sharing	71
21.1	Share data online	71
21.2	Unshare data	72
21.3	Simplified publishing (unofficial ESG publication)	72
21.4	Standard publishing (official ESG publication)	74
22	Existing projects	75
23	References	77
24	Search	79
	Bibliography	81

NorESM1 is the Norwegian Earth System model used for CMIP5. The model is based on the CCSM framework (http://en.wikipedia.org/wiki/Community_Climate_System_Model). However, NorESM has special features developed by Norwegian researchers.

The purpose of the NorESM1 documentation is to provide a common place for NorESM users and developers to share information.

- What tools are you using?
- Which version should I run for what purpose? etc.

NorESM version 2 documentation is found here: <https://github.com/NorESMhub/NorESM/edit/noresm2>

If you have any questions, create a new issue on github at <https://github.com/NorESMhub/norESM-docs/issues>

INTRODUCTION

NorESM1 is the Norwegian Earth System model used for CMIP5. The model is based on the CCSM framework (http://en.wikipedia.org/wiki/Community_Climate_System_Model). However, NorESM has special features developed by Norwegian researchers.

Main references are:

GMD - Special issue The Norwegian Earth System Model: NorESM; basic development, validation, scientific analyses, and climate scenarios

http://www.geosci-model-dev.net/special_issue20.html [1] [2] [3] [4]

This website contains information shared between NorESM developers and users.

1.1 Obtaining a version of the model

- The development version has been moved to git: Obtain a copy through git clone <https://github.com/metno/noresm.git> You first need to be registered as a noresm user on github (see detailed info in *Obtain a copy of the model (using git)*).

If you are on a normal ubuntu PC and want the source code, you might see that “svn checkout” complains about “gnome keyring”. If you see this problem, the solution is here: <http://askubuntu.com/questions/206604/svn-and-gnome-keyring>

- Need access to other versions: Special access document:

https://docs.google.com/a/met.no/document/d/1G1ezxtBhzDyNWwrKJYWmp8gn402bOWThe_6gN00PDMQ/edit?usp=sharing

1.2 Running / Configuring the model

- *Newbies guide to running NorESM*
- *Advanced configuration (NorESM1)*
- *Fluxes crossing boundaries*

1.3 Develop the model

1.3.1 Setting up at different machines

Most developers compile and run NorESM on hexagon (hexagon.bccs.uib.no). That machine uses the portland group fortran compiler. Most developers develop the code on that machine using “develop/compile/run/analyze print statements” on that machine.

Some experiments have also been done with compiling running CAM on a normal Linux PC in order to use interactive debuggers. (see below)

Setting up CAM on your own linux PC

1.3.2 Issue tracker

Any development should ideally be agreed with the NorESM development team and be properly described in the issue tracker, see the link below

Using the issue tracker

If you have changed the model and want to merge your changes to the trunk, your model has to pass some tests:

1.3.3 Testing

Test list for NorESM

1.3.4 Version control best practices

- **NEW: After switching to git (13th november 2015) the** svn-repository is read-only. Some advice on how to use the new git-repository are available here: *Obtain a copy of the model (using git)*

Some guidelines for modifying NorESM’s subversion repository:

SVN - Best Practice/FAQ

How-to for setting up svn repositories on NorStore: *Subversion how-to for NorStore*

noresm2nc4mpi

- noresm2nc4norstore
- *Model Diagnostic Tools*
- esmvaltool

1.4 Archive model results

Long-term archiving is normally done on NorStore's disk resources (e.g. in /projects/NS2345K/noresm/cases).

To avoid loss of data, another copy should be placed on tape. For instructions, see *NorStore Tape Storage*

Data that builds the basis of publications should be migrated to NorStore's Research Data Archive in order to guarantee preservation and also to offload the project area. For specific NorESM instructions, see *NorStore Research Data Archive: Guidelines for ingestion of NorESM output*

1.5 CMIP5 archive of NorESM results

NorStore Research Data Archive: Guidelines for ingestion of NorESM output

1.6 Share model results

Model output and derived data products can be shared via the Norwegian Earth System Grid data portal <http://noesg.norstore.no> (see *Simple online sharing* for instructions).

1.7 Past and ongoing work

Several simulations have been performed with NorESM. A list of available simulations and runs can be found here. listofruns. The page also contains an overview of planned simulations. A fairly extensive description of the model and to some extent also the CMIP5 runs can be found at <http://pcmdi9.llnl.gov/esgf-web-fe/>

Choose one of the links. Search for NorESM1-M CMIP5 in the search fields. Choose the link model documentation

NorESM is also used in several projects: *Existing projects*

1.8 Resources

See presentation from NorESM workshop November 28'th 2013

- EVA final meeting: CMOR-ization and ESGF publication of NorESM2 simulations - Ingo Bethke
- NICEST workshop for ESGF at NSC: CMOR post-processing of NorESM CMIP5 & CMIP6 output - Ingo Bethke
- ESMValTool setup for NorESM (10 May 2017) - Ingo Bethke
- TaiESM CCliCS workshop in Taipei 2016 - Ingo Bethke pptx, pdf
- HappiEVA kick-off 2016 - Ingo Bethke
- NeIC/Nordic ESM workshop 2015 - CMOR-izing of NorESM output by Ingo Bethke
- Presentation on data management in EVA - EVA meeting, 2014, Bergen
- NorESM Workshop 2014 SU - Alf Grini
- NorESM Workshop 2014 SU - Annica Ekman
- NorESM Workshop 2014 SU - Chandan Basu

- NorESM Workshop 2014 SU - Francesco Pausata
- NorESM Workshop 2014 SU - Juan-Camilo Acosta
- NorESM Workshop 2014 SU - Risto Makkonen
- NorESM Workshop 2014 SU - Trond Iversen
- NorESM Workshop 2014 SU - Vidya Varma
- After NorESM Workshop 2014 SU - Alf Kirkevåg: some AeroTab-slides

1.9 References

NEWBIES GUIDE TO RUNNING NORESM

The purpose of this page is to be able to set up and run the model within 15 minutes. For more advanced configuration, please consult http://www.cesm.ucar.edu/models/ccsm4.0/ccsm_doc/ug.pdf

This guide assumes that you have properly checked out the model to some directory which will call \$NORESM. This directory will contain subdirectories “models” and “scripts”.

2.1 Go to scripts directory

```
cd $NORESM/scripts
```

2.2 Create the case

```
./create_newcase -case ../cases/casename/ -mach machinename -res f19_g16 -compset_↵  
↵compsetname
```

(where casename, machinename and compsetname are user input. Res is imodel resolution. The user can *not* give any resolution since input data are not prepared for any resolution in NorESM.)

To see a list of available composites type

```
./create_newcase -list
```

The simplest case which runs production tagged aerosols and data ocean in a VERY coarse resolution is

```
./create_newcase -case /path/to/where/I/store/the/case -compset NFPTAERO -mach hexagon -  
↵res f10_f10
```

2.3 Configure the case

```
cd $NORESM/cases/casename
```

edit any configuration files (understand env_conf.xml and env_run.xml)

```
./configure -case (in NorESM1 / CAM4)
```

or

```
./cesm_setup (in NorESM2 / CAM5)
```

After these two commands, the case is configured

2.4 Build the case

```
./casename.machinename.build (NorESM1 / CAM4)
./casename.build (NorESM2 / CAM5)
```

2.5 Run the case

```
qsub casename.machinename.run
```

2.6 Example

There are already several pre-defined compsets. They all have long and short names. As an example we can use the compset N_2000_AEROSLO_CN (with short name N2000AERCN). Thus a valid command on the machine *hexagon* is:

```
./create_newcase -case /path/to/my/case/directory/ -mach hexagon -res f19_g16 -compset_
↪ N2000AERCN
```

(will run a “year 2000” CAM4/NorESM1 case with the Oslo-aerosols)

```
./create_newcase -case /path/to/my/case/directory/ -mach hexagon -res f19_g16 -compset_
↪ FAMIPC5
```

(will configure the “default” atmosphere-only simulation of CAM5)

2.7 Important files

The most important files to understand in your case-directory are:

- env_run.xml (model run type, how long time to run etc)
- env_conf.xml (model configuration)

2.8 More information

Go to *Advanced configuration (NorESM1)* for more information

ADVANCED CONFIGURATION (NORESM1)

3.1 Understanding compsets

A compset is a collection of configuration parameters which describe a specific case. NorESM has several pre-defined compsets. The CCSM users guide provides information on configuring your own compsets. The compsets define things like *how many processors will be used in this case, which options go to cam_oslo in this case*

For existing compsets, search for the file config_compsets.xml in your \$NORESM folder. They can also all be printed by going to \$NORESM/scripts and type the command

```
./create_newcase -list
```

3.2 AMIP type simulations

For AMIP type simulations, using the data ocean model with prescribed (observed) SST, use one of the pre-defined compsets with long names starting with “**NF_**”. One, used in the AMIP run for CMIP5, is “NF_1979-2005_AER_AMIP_OBS”, with the short name “NF2005AERAMIPO”. To create a Case with the chosen name AMIPtest for this compset, as an example, write

```
./create_newcase -case ../cases/AMIPtest/ -mach hexagon -res f19_f19 -  
↪compset NF2005AERAMIPO
```

3.3 Running with offline aerosol interaction

In order to limit the number of compsets there are (at present) no compsets available for automatically setting up the model in offline mode (i.e., the meteorology is forced by aerosol optics and CDNC from CAM4 instead of CAM4-Oslo) or for taking out extra AeroCom diagnostics, which requires numerous additional subroutine calls and therefore is quite expensive to run, both with respect to CPU time and memory. To set up the model in offline mode (before compiling), simply find all subroutines which contain the logical variable AEROFFL (e.g.: grep AEROFFL *.F90), and replace all instances of

```
#undef AEROFFL
```

and

```
!#define AEROFFL
```

with

```
#define AEROFFL
```

This is useful for short simulations where we want to look at direct and/or indirect radiative forcing by aerosols, since the meteorology does not change with changing aerosol emissions (e.g. for year 1850 and 2000). Similarly, to set up the model to take out additional aerosol output for use in AeroCom or other studies where there is a need for extensive aerosol diagnostics, find all subroutines which contain the logical variable AEROCOM (e.g.: `grep AEROCOM *.F90`), and then replace all

```
!#define AEROCOM
```

with

```
#define AEROCOM
```

The model may be run with any combination of these options: with AEROFFL only, with AEROCOM only, or with AEROFFL and AEROCOM activated at the same time.

See also presentation from NorESM workshop November 28'th 2013.

3.4 Use of look-up tables for aerosol optics and activation to cloud droplets

See presentation from NorESM workshop November 28'th 2013:

The look-up table code, AeroTab, is now available through subversion under `cam/tools/AeroTab` on `norEsmTrunk`. To obtain a local copy (`myAeroTab`) of the newest version without checking out the whole NorESM model, run

```
svn checkout https://svn.met.no/NorESM/noesm/trunk/noesm/models/atm/cam/tools/AeroTab_
↪myAeroTab
```

Or, if you are interested in the CMIP5 version (with some updates), run instead

```
svn checkout -r 199 https://svn.met.no/NorESM/noesm/trunk/noesm/models/atm/cam/tools/
↪AeroTab myAeroTab
```

3.5 Use of chemistry

See presentation from NorESM workshop November 28'th 2013

FLUXES CROSSING BOUNDARIES

NorESM can simulate several biogeochemical cycles. Here is a list of the fluxes which cross boundaries in the models and how to enable/disable interactions

Component	Source	Receiver	Unit	How to enable
Dust	CLM	CAM	kg/m ² /A	Always calculated by CLM. Picked up and used by atmosphere. Used differently in different aerosol-packages (bulk-aero,MAM, OSLO_AERO)
Dust	CAM	HAM-MOC	kg/m ² /A	Leaves CAM as cam_out%dstwet{n} ,cam_out%dstdry{n}(n=1-4), picked up by HAMMOC in coupler
DMS	HAM-MOC	CAM	kg/m ² /A	HAMMOC writes DMS to the coupler if one sets "CCSM_BGC=CO2_DMSA" in env_run.xml. Picked up and used by CAM-Oslo if namelist-variable dms_source=='ocean_flux'. Only compsets with MICOM%ECO run HAMMOC
CO2	CLM	CAM	??	
CO2	HAM-MOC	CAM	??	
CH4	CLM	CAM	??	
isoprenes/monoterpenes	CLM	CAM	kg/m ² /A	Calculated by MEGAN in CLM by setting "isoprene = isoprene', 'monoterp = myrcene + sabinene +limonene + carene_3 + ocimene_t_b + pinene_b + pinene_a'" Note that any additional emission-file for monoterpenes will be ADDED to the MEGAN emissions
NO (?)	CLM	CAM		
NH3 (?)	HAM-MOC	CAM		
NHx (?)	CAM	CLM+HAMMOC		
NOx (?)	CAM	CLM+HAMMOC		

SETTING UP CAM ON YOUR OWN LINUX PC

CAM is the atmospheric component of NoRESM. It is possible to compile and run CAM on your own Linux PC. This makes it easier to debug simple tests using programs like ddd or gdb. CAM 5.3 compiles with gfortran.

5.1 Obtain CAM 5.3 source code

```
svn checkout https://svn-ccsm-release.cgd.ucar.edu/model_versions/cesm1_2_0
```

You will be asked for a user name and password. You can easily get that from NCAR through http://www.cesm.ucar.edu/models/cesm1.0/register/register_cesm1.0.cgi

(the first time it will use your unix user name.. Just type the wrong password, and then it will prompt you for another user name, then use the one provided by NCAR).

- **To use the development version of CAM5-Oslo, check out NorESM (not CESM) from the noesm repository**

```
svn checkout https://svn.met.no/NorESM/noesm/branches/featureCAM5-OsloDevelopment_  
↪trunk2.0-4 myCamOsloDev
```

5.2 Compile the netcdf libraries using gfortran

1) INSTALL NETCDF (4.2): Follow instructions on <http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-install/Quick-Instructions.html#Quick-Instructions>

I created library directories on /home/alfg/LIBS/.

For example /home/alfg/LIBS/netcdf-4.2.gfortran. These directories are use as “prefix=” in the installation guide..

When installing:

- Use instructions on how to build with hdf5
- Use the instructions about shared libraries. (There are also instructions about static libraries)
- Remember to set the LD_LIBRARY_PATH variable as described. Otherwise it does not work.

==> Some exceptions

(Build order is wrong) http://mail.lists.hdfgroup.org/pipermail/hdf-forum_lists.hdfgroup.org/2013-May/006818.html

(Have to do separately)

```
make
make check
make install
```

Add the following to your `.bashrc` file:

```
export FC=gfortran
export CC=gcc
export CFLAGS=-O0
export CXX=g++
export CXXFLAGS=-O0
```

The netcdf c-library complains about a non-successful compilation because it can not find Doxyfile. This is not important. The compilation needs also the following env-variables: `CPPFLAGS=-I${dir}/include`, `LD_FLAGS=-L${dir}/lib`, where “dir” is directory you use for the “configure” scripts

3) Get the netcdf source code (including the c-headers) from <http://www.unidata.ucar.edu/software/netcdf/docs/getting.html> a) make b) make check c) make install

4) Get fortran netcdf api (<http://www.unidata.ucar.edu/downloads/netcdf/index.jsp>) and follow <http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-fortran-install.html>

5.3 Compile CAM on your Linux PC

When you have successfully obtained netCDF libraries, you need to set some environment variables in order to tell CAM where to find these libraries, add the following to your `.bashrc` file (but change the path to the correct path). Make sure you have the netcdf.mod-file in “MOD_NETCDF”. These environment variables are used by the CAM “configure” script!

```
export INC_NETCDF=/home/alfg/LIBS/netcdf-4.2.gfortran/include
export LIB_NETCDF=/home/alfg/LIBS/netcdf-4.2.gfortran/lib
export MOD_NETCDF=/home/alfg/LIBS/netcdf-4.2.gfortran/include
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$LIB_NETCDF
```

The steps for building and installing CAM are well described at http://www.cesm.ucar.edu/models/cesm1.0/cam/docs/ug5_1/ug.html

You need the following settings in your `.bashrc` file:

```
export camcfg=/path/to/CESM_1_2_0/cesm1_2_0/models/atm/cam/bld
export CSMDATA=/path/to/cam/input/data
```

This command configures CAM (not including any CAM-Oslo code) on my machine. It creates a Makefile used later for compilation.

```
$camcfg/configure -dyn fv -debug -hgrid 10x15 -fc gfortran -nospmd -nosmp -test -fc_type_
↪gnu
```

- As long as `$camcfg` is defined, you can execute this from any directory..

Then you need the following commands **executed from the same directory as you executed “configure”**

```
$camcfg/build-namelist -test -config config_cache.xml
```

```
make
```

The input data have been collected and put at norstore:

```
/projects/NS2345K/noresm/CAM5.3_PCLinux/CAM5.3Input10x15/csm/inputdata
```

You should copy these files to somewhere on your computer. The CSMDATA environment variable should point to the “input-data” directory after the copying. For example I copied it to /disk1/alfg/csm/, so my \$CSMDATA is

```
alfg@pc4400:$ echo $CSMDATA /disk1/alfg/csm/inputdata
```

Finally run the model simply executing the command ./cam

NOTE: In CAM5.5 (upcoming versions) this same procedure is supposed to work with the following command (NOT VERIFIED):

```
$camcfg/configure -fc gfortran -fc_type gnu -debug -nospmd -nosmp -dyn fv -res 10x15 -  
→ice sice -phys cam5
```

5.4 Including cam-oslo code

This assumes you checked out NorESM (and CESM) with subversion

Works with latest version of oslo aerosol development branch

```
$camcfg/configure -dyn fv -debug -hgrid 10x15 -fc gfortran -nospmd -nosmp -test -fc_  
→type gnu -chem trop_mam_oslo
```

There are additional input needed for the CAM-Oslo code. They are available in the same input-directory as the normal CESM-input files. As long as the whole input directory is copied and the CSMDATA-environment-variable is set, you don't have to do anything. The input-directory at norstore is:

```
/projects/NS2345K/noresm/CAM5.3_PCLinux/CAM5.3Input10x15/csm/inputdata
```

5.5 Debug the model using ddd or gdb

Some first test show that the following information is useful:

gdb hangs forever on backtrace command sometimes. Adding //set print frame-arguments none// to your ~/.gdbinit file solves that problem on the expense of less information on “bt” command.

ddd sometimes hangs forever on startup. If that happens you need to remove the ~/.ddd/init file

gdb has problems with printing info about allocatable arrays. Allocatable arrays have to be displayed as “*((real_8 *)my_array + N)@M” where N is number of elements beyond first element and M is number of elements to show. (See bottom of this page (and links therein) <http://stackoverflow.com/questions/11786958/how-to-print-fortran-arrays-in-gdb>)

It is sometimes useful to let the compiler tell you about additional errors. Assuming you use gfortran on your PC: See <https://gcc.gnu.org/onlinedocs/gfortran/Code-Gen-Options.html> for additional options. For example if you want to check just about everything, add “-fcheck=all” to the “FC_FLAGS” in the gfortran section. (Note: Running configure again re-generates the Makefile)

5.6 View results

ncview has trouble visualizing results which are of this coarse resolution (10x15 degrees). Panoply is a better option. Download from http://www.giss.nasa.gov/tools/panoply/download_gen.html

1. Unpack the files
2. Add the folder where you find panoply.sh to your path. In your .bashrc file, add:

```
export PATH=$PATH:/path/to/panoply/dot/sh
```

3. launch program with “panoply.sh netcdfFileName” from any folder

5.7 Configure your case

If you want an other configuration than the standard configuration, you must add a use-case. The use case is an xml-file which is added to the /cesm1_2_0/models/atm/cam/bld/namelist_files/use_cases/ directory.

Below is an example of a use-case which writes some more frequent output (every hour) to history files.

The file says that we have 5 different history files. The first one is the monthly output with max one time step. The other files are output every hour (-1) with max number of samples 30 in each file.

For each of the different files (1-5) we have specified the fields we want to output. For example in the second, we output QREFHT, TREFHTMN etc..

Fields which end in “:I” are instantaneous values as opposed to time averages.

Save the xml-file to the use_cases directory as “my_case.xml” and run the command

```
$camcfg/build-namelist -test -config config_cache.xml -use_case my_case
```

```
<?xml version="1.0"?>
<namelist_defaults>
<mfilt>1,30,30,30,30</mfilt>
<nhtfrq>0,-1,-1,-1,-1</nhtfrq>
<!-- TEM diagnostics output -->
<fincl2>'QREFHT', 'TREFHTMN', 'TREFHTMX', 'TREFHT', 'PRECT', 'PRECC', 'PRECSC', 'PRECSL', 'PSL',
↪ 'T', 'Z3', 'U', 'V', 'PS', 'TS', 'SST', 'PHIS', 'CLDTOT'</fincl2>
<fincl3>'U850:I', 'V850:I', 'T850:I', 'Q850:I', 'OMEGA850:I', 'U:I', 'V:I', 'T:I', 'PS:I', 'PSL:I'
↪ ', 'Q:I', 'PHIS:I'</fincl3>
<fincl4>'PRECT', 'LHFLX', 'SHFLX', 'FLDS', 'FLNS', 'FSNS', 'PRECC', 'PRECSC', 'PRECSL'</fincl4>
<fincl5>'TREFHT:I', 'QREFHT:I', 'TS:I', 'SST:I', 'PS:I'</fincl5>

</namelist_defaults>
```


MODEL DIAGNOSTIC TOOLS

This page links to tools used for the NorESM model evaluation.

6.1 NorESM Diagnostic Packages

Output from the latest NCAR diagnostic pages can be found on nird here:

via the web: http://ns2345k.web.sigma2.no/noresm_diagnostics/

via the filesystem on nird here: /projects/NS2345K/www/noresm_diagnostics/

The diagnostics packages are currently available on NIRD. Each package can be run/configured from the command line using the program `diag_run`:

```
-----  
Program: /projects/NS2345K/noresm_diagnostics/bin/diag_run Version: 5.1  
-----  
  
Short description: A wrapper script for NorESM diagnostic packages.  
  
Basic usage:  
  diag_run -m [model] -c [test case name] -s [test case start yr] -e [test case end yr]  
↪ # Run model-obs diagnostics  
  diag_run -m [model] -c [test case name] -s [test case start yr] -e [test case end yr] -  
↪ c2 [cntl case name] -s2 [cntl case start yr] -e2 [cntl case end yr] # Run model1-  
↪ model2 diagnostics  
  nohup /projects/NS2345K/noresm_diagnostics/bin/diag_run -m [model] -c [test case name] -  
↪ s [test case start yr] -e [test case end yr] &> out & # Run model-obs diagnostics in_  
↪ the background with nohup  
  
Command-line options:  
  -m, --model=MODEL Specify the diagnostics package (REQUIRED).  
    Valid arguments:  
      cam      : atmospheric package (AMWG)  
      clm      : land package (LMWG)  
      cice     : sea-ice package  
      micom    : ocean package  
      hamocc   : biogeochemistry package  
      all     : configure all available packages.
```

(continues on next page)

(continued from previous page)

```

-c, -c1, --case=CASE1, --case1=CASE1      Test case simulation (OPTIONAL).
-s, -s1, --start_yr=SYR1, --start_yr1=SYR1 Start year of test case climatology.
↳(OPTIONAL).
-e, -e1, --end_yr=EYR1, --end_yr1=EYR1    End year of test case climatology.
↳(OPTIONAL).
-c2, --case2=CASE2                        Control case simulation (OPTIONAL).
-s2, --start_yr2=SYR2                    Start year of control case climatology.
↳(OPTIONAL).
-e2, --end_yr2=EYR2                      End year of control case climatology.
↳(OPTIONAL).
-i, -i1, --input-dir=DIR, --input-dir1=DIR Specify the directory where the test case.
↳history files are located (OPTIONAL).
                                          Default is --input-dir=/projects/NS2345K/
↳noesm/cases
-i2, --input-dir2=DIR                    Specify the directory where the control.
↳case history files are located (OPTIONAL).
                                          Default is --input-dir=/projects/NS2345K/
↳noesm/cases
-o, --output-dir=DIR                     Specify the directory where the package(s).
↳the climatology and time-series files should be stored (OPTIONAL).
                                          Default is --output-dir=/projects/NS2345K/
↳noesm_diagnostics/out/$USER
-p, --passive-mode                       Run the script in passive mode: the.
↳diagnostic script
                                          will be configured but not executed.
↳(OPTIONAL).
-t, --type=TYPE                          Specify climatology or time series.
↳diagnostics (OPTIONAL):
                                          valid types are --type=climo and --
↳type=time_series.
                                          Default is to run both. Note that the time
↳series are computed over the entire simulation.
-w, --web-dir=DIR                        Specify the directory where the html should.
↳be published (OPTIONAL).
                                          Default is --web-dir=/projects/NS2345K/www/
↳noesm_diagnostics
--no-atm Run CLM diagnostics without CAM data. Must be used for offline CLM.
↳simulations.
Examples:
diag_run -m all -c N1850_f19_tn11_exp1 -s 21 -e 50 # model-obs diagnostics of.
↳case=N1850_f19_tn11_exp1 (climatology between yrs 21 and 50) for all model components.
diag_run -m cam -c N1850_f19_tn11_exp1 -s 21 -e 50 -w /path/to/my/html # model-obs.
↳diagnostics in CAM, publish the html in /path/to/my/html.
diag_run -m micom -c N1850_f19_tn11_exp1 -t time_series # model-obs time-series.
↳diagnostics in MICOM for all years represented in the model output directory (/
↳projects/NS2345K/noesm/cases/N1850_f19_tn11_exp1/ocn/hist/).
diag_run -m cice -c N1850_f19_tn11_exp1 -s 21 -e 50 -p # configure (but do not run).
↳model-obs diagnostics for CICE.
diag_run -m clm -c N1850_f19_tn11_exp1 -s 21 -e 50 -i /input/directory1 -c2 N1850_
↳f19_tn11_exp2 -s2 21 -e2 50 -i2 /input/directory2 # model1-model2 diagnostics for CLM.
↳with user-specified history file directories

```

(continues on next page)

(continued from previous page)

```
diag_run -m micom -c N1850_f19_tn11_exp1 -s 21 -e 50 -t climo # model-obs_
↳climatology diagnostics (no time series) for MICOM:
diag_run -m cam -o /my/dir # install CAM diagnostics in /my/dir with minimal_
↳configuration.
diag_run -m micom,hamocc -c N18500C_f19_tn11_exp1 -s 21 -e 50 # model-obs_
↳diagnostics for MICOM and HAMOCC.
diag_run -m clm -c N1850_f19_tn11_clmexp1 -s 71 -e 100 --no-atm # model-obs time-
↳series diagnostics for an offline (uncoupled) CLM simulation.
diag_run -m hamocc -c N18500C_f19_tn11_exp1 -s 31 -e 100 -t time_series # model-obs_
↳time-series diagnostics in HAMOCC between yrs 31 and 100.
```

A comprehensive technical summary of `diag_run` (pdf)

Report any problems, comments or suggestions to Yanchun He: yanchun.he@nersc.no

6.1.1 Recent updates

- 29.08.10. Update to v5.1: update NCO/NCL versions to support efficient process of compressed netcdf-4 files.
- 29.06.18. Update to v5.0: new fields to HAMOCC and MICOM diagnostics; minor fixes to other issues.
- 20.04.18. Update to v4.3: added new fields to HAMOCC diagnostics.
- 19.04.18. Update to v4.2: included ability to do time-series diagnostics between two user-specified years.
- 18.04.18. Update to v4.1: improved climatology and time-series calculations in CLM, and introduction of the `--no-atm` option to enable diagnostics for offline CLM simulations.
- 09.04.18. Update to v4.0: included the HAMOCC diagnostics package.
- 23.02.18. Update to v3.1: added monthly MLD, seasonal SST/SSS and annual meridional heat/salinity fluxes to the MICOM diagnostics.
- 17.01.18. Update to v3.0: the first version of MICOM diagnostics has been included.
- 28.11.17. Update to v2.0: included a set of time series plots in CAM diagnostics, along with an html interface, which can be accessed from the index page (sets.htm).

6.1.2 Using `diag_run` with cron

If you want to use `diag_run` with crontab, you first need to load `$HOME/.bash_profile`, i.e.:

```
#-----
# Min Hour Day Month Weekday Command(s)
#-----
50 09 23 11 \* . $HOME/.bash_profile; /projects/NS2345K/noresm_diagnostics/bin/diag_run -
↳m cam,cice -c N18_f19_tn11_080617 -s 21 -e 50 -o /scratch/$USER/noresm_diagnostics2 -w_
↳\
/projects/NS2345K/www/test -t time_series >& /scratch/$USER/cron_out
```

6.1.3 Other tips

It is useful to add `diag_run` as an alias in `$HOME/.bashrc`, so that you do not need to write out the whole path every time you run it: alias

```
diag_run='/projects/NS2345K/noresm_diagnostics/bin/diag_run'
```

6.1.4 NorESM diagnostics on GitHub

The NorESM diagnostics packages and `diag_run` are included in the Git version control repository: <https://github.com/johiak/NoresmDiagnostics>

Aerosol and Chemistry, Clouds and Forcing Diagnostics

In both the default CAM5-aerosol packages (MAM3,MAM7) and the Oslo-aerosol packages, the budget terms can be taken out using a variable in the namelist :

6.1.5 Configuring a run with more aerosol diagnostics in (NorESM2)

```
&phys_ctl_nl  
history_aerosol = .true.  
/  

```

Two more diagnostics are useful:

- Enable estimates multiple calls to radiation which are necessary for effective radiative forcing estimates
- Enable diagnostics for AEROCOM

To enable this, take the file `cam/src/physics/cam_oslo$ vim preprocessorDefinitions.h` and copy it to your `Source-Mods/src.cam` folder

Change both preprocessor definitions to true

```
#define AEROCOM  
#define AEROFFL
```

The AEROCOM-token turns on diagnostics needed for AEROCOM The AEROFFL-token tells the model to do additional radiation-diagnostics for aerosol indirect effect

6.2 Tracer Budget terms

6.2.1 Fields produced in monthly average files when running with budgets activated

Running with budgets activated will produce the following terms in the monthly output files:

Output variable name	Meaning	Comment
SF{Tracer} GS_{Tracer}	Emissions from surface gas phase chemistry	3D-emissions and gas phase washout included in this term
AQ_{Tracer}	aquous chemistry	
{Tracer}_Mixnucl	Activation in clouds and evaporation of cloud droplets	
{Tracer}_DDF	Dry deposition flux (aerosol tracers)	
{Tracer}_SFWET	Wet deposition flux (aerosol tracers)	
{Tracer}_condtend	loss/production in condensation/nuclation	(CAM-Oslo only)
{Tracer}_coagTend	loss/production in coagulation	(CAM-Oslo only)
DF_{Tracer}	dry deposition flux (gas tracers)	output with history_aerosol with CAM-Oslo only
WD_A_{Tracer}	wet deposititon flux (gas tracers)	output with history_aerosol with CAM-Oslo only
{Tracer}_CLXF	3D-emissions (“external forcing”)	output with history_aerosol with CAM-Oslo only
{Tracer}_clcoagTend	loss of tracer due to coagulation with cloud droplets	output with history_aerosol with CAM-Oslo only

Note: Since 3D-emissions and and gas washout rates are included in the term GS_{Tracer} in the mozart chemistry solver, the individual terms can be found like this (example for SO2):

```
ncap2 -0 -s GS_ONLY_SO2=GS_SO2-WD_A_SO2-SO2_CLXF infile.nc outfile.nc
```

More info on SO2 budgets (see /models/atm/cam/tools/diagnostics/ncl/ModIvsModII/ for scripts with info on all tracers):

GS_SO2 contains the SO2 budget terms for all that goes on in the chemistry-routine, which is

1. Gas phase chemistry,
2. Wet deposition, and
3. 3D-emissions.

Gas phase chemistry is both production from DMS (GS_DMS) and loss through OH (GL_OH)

For calculations of net loss, e.g. used to calculate SO2 life-times, we're interested in the loss through OH from the chemistry-term (GL_OH).

```
GS_SO2 = GL_OH + SO2_CLXF - WD_A_SO2 - GS_DMS*64/62
```

or

```
GL_OH = GS_SO2 - SO2_CLXF + WD_A_SO2 + GS_DMS*64/62
```

Estimating chemical loss w.r.t. S (instead of SO2 or DMS), for comparison with CAM4-Oslo numbers:

```
net chemical loss gas phase = (GS_SO2/1.998 - SO2_CLXF + WD_A_SO2)/1.998 + GS_DMS/1.938
net chemical loss = net chemical loss gas phase + AQ_SO2/1.998
```

Finally, total net loss (used to calculate life-time = -load/(net loss), where load = cb_SO2/1.998):

```

net loss =
- WD_A_S02/1.998 ;wet deposition in kg/m2/sec (positive in output file)
- DF_S02/1.998 ;dry deposition in kg/m2/sec (positive in output file)
+ AQ_S02/1.998 ;wet phase production of S04 in kg/m2/sec (negative in output file)
+ (GS_S02 - S02_CLXF + WD_A_S02)/1.998 + GS_DMS/1.938 ; net chemical loss gas phase

```

6.2.2 Looking at the aerosol budgets (CAM-Oslo only)

- Go to the directory models/atm/cam/tools/diagnostics/ncl/budgets
- Change the filename to use in the file budgets.ncl (“myFileName” around line 18). Should be for example yearly average of month-avg file in a run with budgets
- Run the script budgets.sh to create a pdf-file (output.pdf)

6.3 NCL Model Version Comparison package (Alf K)

6.3.1 Making ncl plots of often used aerosol and cloud fields, including ERFs, for two model versions (CAM-Oslo only)

- Make a local copy (on Linux) of the directory models/atm/cam/tools/diagnostics/ncl/ModIvsModII
- Assuming that you have produced output data from 4 simulations: two different model versions, each with PD and PI emissions, and all run with `#define AEROCOM & AEROFFL`:
- **In ModIvsModII.csh (note: read the header info):**
 - edit model info for the first model (shown to the left in the plots): modelI = CAM4-Oslo or modelI = CAM5-Oslo ?
 - provide paths and partial file names of the model data (PD and PI) for Model I (CAM4-Oslo or CAM5-Oslo) and Model II (must be CAM5-Oslo)
 - choose desired plot format (plotf=ps, eps, pdf or png)
- Run the script: `./ModIvsModII.csh`
- **Furthermore, to display the plots in an organized form by use of a web browser (only possible if the chosen plot format is png):**
 - download htm template files from <ftp://ftp.met.no/projects/noresmatm/upload/NorESM2Diagnostics/ModIvsModII/htm-templates/>
 - edit general model info (only) in ModIvsModII.htm, and manually cut and paste the mass budget numbers from the script output into this file
 - copy all png (plots) and htm files to the desired output (common) directory
 - open ModIvsModII.htm in your browser: hyper-links to all other htm files, including plots, are found here
- Example: ftp://ftp.met.no/projects/noresmatm/upload/NorESM2Diagnostics/ModIvsModII/revision610inclSOA-Nudged_1984-12to1985-11_vs_CAM4-Oslo/ModIvsModII.htm

6.4 Cloud water mass and number analysis (budgets)

6.4.1 Configuring a run with more cloud diagnostics in NorESM2

To switch on extra output for cloud diagnostics (mass and number tendencies for liquid water and mass) change the following namelist variable:

```
&phys_ctl_nl history_budget = .true. /
```

A python script for plotting the mass and number budgets for the cloud microphysics can be found under:

models/atm/cam/tools/diagnostics/ncl/cloudBudgets

in the same branch. Copy the script to your local computer or lustre and edit the script to read the correct input file(s) (instructions inside the script). Run the script by typing:

```
python scriptname.py
```

in your terminal.

6.5 Automatic AEROCOM analysis

To prepare output so that it is processed automatically by the aerocom tools, use the script located at **models/atm/cam/tools/aerocom/** in the svn repository. The script prepares files such that the idl aerocom tools prepare plots for the aerocom webinterface: [URL link to NorESM on AeroCom webinterface](#)

The script requires <ModelName>_<ExperimentName> and <Period> as input.

- <Period>: for a climatological average and run choose 9999 , for nudged simulations choose the year of the meteorology
- <ModelName>_<ExperimentName>: is the dataset identifier under which the plots appear on the AeroCom webinterface in the required format NorESM-CAM5_svn{RevisionNumber}_YYMMDD{initials}_Freetext.

Example:

“NorESM-CAM5_svn1094_151201AG_CMIP6endelig”

Initials AG: Alf Grini, AK: Alf Kirkevåg, DO: Dirk Olivie...

Where the date YYMMDD corresponds to the time when the AeroCom data preparation script has been executed.

The script creates files named like

```
aerocom3_<ModelName>_<ExperimentName>_<VariableName>_<VerticalCoordinateType>_<Period>_<Frequency>.nc
```

- <ModelName> eg NorESM-CAM53
- <ExperimentName> svn{RevisionNumber}_YYMMDD{initials}_Freetext
- <VariableName> aerocom variable names
- <VerticalCoordinateType> “Surface”, “Column”, “ModelLevel”, “SurfaceAtStations”, “ModelLevelAtStations”
- <Period> eg “2008”, “2010”, “9999”
- <Frequency> “timeinvariant”, “hourly”, “daily”, “monthly”, “sat1000”, “sat1330”, “sat2200”, “sat0130”

Note that VerticalCoordinateType is dependent on the variable!! It is not a question about “vertical coordinate type used in model simulations”!

The script copies files on norstore into /projects/NS2345K/CAM-Oslo/DO_AEROCOM/<ModelName>_<ExperimentName>/renamed/

6.6 ESMval CIS JASMIN platform and tools

- ESMVALtool <http://www.geosci-model-dev-discuss.net/8/7541/2015/gmdd-8-7541-2015-discussion.html>
- cis tools <http://www.cistools.net>
- JASMIN <http://www.jasmin.ac.uk/services/jasmin-analysis-platform/>

6.7 Post analysis and workup of CAM diagnostics output tables

A tool for post analysis of (multiple) CAM diagnostics ASCII tables can be found in the following repository:

GitHub https://github.com/jgliss/noresm_diag_postproc

To get started, please follow the instructions in repository README (displayed in repository). Currently, the main analysis tool is a jupyter IPython notebook called

analysis_tool.ipynb

(https://github.com/jgliss/noresm_diag_postproc/blob/master/analysis_tool.ipynb)

which includes more detailed instructions about setup and options.

Use the notebook

https://github.com/jgliss/noresm_diag_postproc/blob/master/download_tables.ipynb

to download local copies of result tables using a list of URL's.

Short summary:

The notebook reads multiple diagnostics files (runs) into one long table and creates heatmap plots of *Bias*, *RMSE* and *RMSE relative error* for a subset of variables (rows -> y-axis of heatmap) vs. the individual runs (columns -> xaxis).

NOTE: In the current version, you need to download all tables that you are interested in as csv or ascii into one directory, that is specified in the header of the notebook.

Variable groups can be defined in this config file:

https://github.com/jgliss/noresm_diag_postproc/blob/master/config/var_groups.ini

NOTE: If you add groups to this file in your local copy of the repository, please consider sending the updated to jonasg@met.no or to submit a pull request, so that the remote repository remains up to date.

Troubleshooting

If you run into problems, please raise an issue in the repository or contact jonasg@met.no

USING THE ISSUE TRACKER

[Quick Link to NORESM scrum](#)

7.1 Why do we need an issue tracker

- Oslo and Bergen can easily see what the others are working on
- Better traceability of code changes (an issue can contain a reference to a code change)
- Better work planning
- Better communication between developers
- Help us work as a team, not just a collection of individuals.

7.2 Log in and check what is there

- In github: Go to the “issues” tab. Sort by milestones or labels to see the ones you are interested in

7.3 Create issues

- Go to “create issue”. Note that in NorESM, the issues are different **components**. Make sure you select the right component for your issue.
- Also add other information to the issue as label (can be e.g. be a project-name). Adding Multiple labels is OK.

7.4 Priority definition for NorESM

- Blocker: We need to solve this immediately. Some project can not be delivered because of this problem. Problem blocks other people from working.
- Critical: Should be solved as quickly as possible. Major problem with product functionality.
- Major: This is the default priority
- Minor: Nice to do this, but not really necessary
- Trivial: Fix this when you have the time
- Not prioritized: We don't need to do this

7.5 Which issues should we add to different milestones

- Find out together with your team which issues are most important
- Add the issue to the appropriate milestone

7.6 Working

- When you want to start working on something you should always do something which is **included in a milestone**. Those are the tasks that the team has defined as most important.
- Go to the task and choose “assign” and “assign to me”.

7.7 Connection to version control system

- Mention the task when you commit the fix. For example:

```
git commit -m "metno/noresm#346: I did something clever"
```

will link the changeset to the right issue in github.

TEST LIST FOR NORESM

This page contains a list of tests which need to pass before any changes can be ported to the trunk. By definition, the trunk always passes these tests. Any person who finds that the trunk does not pass these tests should immediately send an email to noresm-ncc(at)met.no

Any development version which does **not** pass the tests **can only exist on branches**.

8.1 Restart test

- Start a 2 month branch run, create monthly average output
- Start the same model run, but define it as two single months with restart in between.
- Verify that the last monthly average is equal in both runs.

8.2 Bit-identical meteorology for “technical only” code changes

This test is applicable for code which is not supposed to change the model physics (e.g. writing out extra diagnostics, cleaning up code without changing the functionality..)

- Run the code without the new changes
- Run the code with the new changes
- Verify that temperature is equal in both runs (ncdiff file1.nc file2.nc file3.nc) should give zero values in file3.nc

8.3 Physical tests (early development)

These tests are applicable for early development of the model. At later stages, closer to important deliveries, other (and stricter) tests may apply.

Compare your new result with a result from the previous version of the trunk code.

If any of the global average of the following variables change with more than a limit X, then you must first pass through a discussion with the other developers. Send an e-mail to noresm-ncc(at)met.no before merging your changes to trunk:

- Aerosol optical depth (AOD_VIS, X=10%)
- Cloud droplet number concentration (CDNC, X=10%)
- Temperature (X = 2K)
- Liquid water path (LWP, X=10%)

- Total aerosol number concentration (N_AER, X=10%)
- Total precipitation (PRECT, X=10%)

OBTAIN A COPY OF THE MODEL (USING GIT)

1. **Create a github user:** You can create the github user yourself.

Go to <https://github.com/join> and create a user (Make user-name which is easy to understand, for example FirstnameLastname. You can attach several email-addresses to the same user.)

2. Visit this page: <https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup>
3. Send email to oyvind.seland@met.no to get the right permissions for the new github user (The email must contain who you are and the github username).
4. When you have the right permissions, you can obtain the code.
5. `git clone https://githubUserName@github.com/metno/noresm.git`

The last point will create a new directory called “noresm” in the place you checked out the model. Go to that directory before executing any git-commands.

If you get error messages, verify that you can open the page <https://github.com/metno/noresm> in a web-browser. If you can not, you are probably not a github-user or not member of the noresm group on github.

- Also do the following on all machines where you use git:
 - **Make sure you have a version of git >= 2.0** (add the line “module load git” to your .bashrc files on hexagon, vilje)
 - **git config -global push.default simple** (Will edit your ~/.gitconfig file to a safer way to share your modifications, see <http://stackoverflow.com/questions/13148066/warning-push-default-is-unset-its-implicit-value-is-changing-in-git-2-0>)

Note that with git, the main branch is no longer called “trunk”, it is called “master”!

VERIFY THAT YOU HAVE THE CORRECT CHECKOUT

When you have cloned the model, check that you have gotten what you wanted!

Check that your favourite branch is available using the command

```
git branch --all
```

(You should see the branch “master” on top with a star next to it. This is the branch you get by default. The other branches are listed below with remotes/origin/branchName, but you can not work on them until you check them out, see below)

To check out (locally) your favourite branch and to start working on it, write

```
git checkout -b myBranchName origin/myBranchName
```

(Note that myBranchName must be one of the branches listed by the above command)

If you don't use the “-b” option, you will get something which is not correct. Make sure you are tracking a remote branch. You can write

```
git branch -vv
```

to see which remote branch you are tracking. The output will be something like:

```
myCheckedOutBranchName 1a08184 [origin/myCheckedOutBranchname] ✓  
↔ LatestCommitMessageOnBranch
```

Note that once a branch has been checked out using the -b option, you can switch between any of your checked out branches using the command git checkout aCheckedOutBranchName

Note that in git, switching to a new branch change the files in your working directory. Git will warn you if you have any modified files before switching to a new branch. This is different from how svn works.

MODIFY FILES

Modify the code (for example a file named myChangedFile.F90) and send back to your local repository through

```
git add myChangedFile.F90  
git commit -m "aMessage"
```

The message should link to the issue on github, so if you fix issue number 100 by this code change, you would probably write something like

```
git commit -am "Did part of the work to resolve metno/noresm#100"
```

Verify, using the tool “gitk” that the changes make sense.

GET MODIFICATIONS FROM GITHUB

```
git pull
```

To be absolutely sure about branch names etc, you can do

```
git pull remoteName remoteBranchName:myLocalBranchName
```

which if your are picking up changes the master-branch would translate to

```
git pull origin master:master
```


SEND MODIFICATIONS TO GITHUB

This command assumes that your changes go to the remote branch named like your branch (which is most of the times the case) `git push`

You can also do (to be completely sure):

```
git push remoteName myLocalBranchName:remoteBranchName
```

which if your are changing the master-branch would translate to

```
git push origin master:master
```

(The above command means push my changes to the remote named “origin” from my local branch named master to the remote branch named master. If you are changing another branch than master, you must obviously not write “master”.)

IF YOU DON'T UNDERSTAND AND WANT TO GET BACK TO SVN

<http://www.git-tower.com/blog/git-for-subversion-users-cheat-sheet/>

SVN - BEST PRACTICE/FAQ

We encourage developers to use branches when developing the model. Do you find version control, branching and merging a bit difficult, try the NorESM svn branch/merge tutorial to understand branching and merging. In 30 minutes you can become an svn wizard

Follow this link to find the tutorial:

[svntutorial](#)

- **Note:** As of November 13th 2015, NorESM uses git as version control system. The rules and guidelines for merging/branching, tags and branch names are still valid in the new system

BRANCHES

16.1 What is a branch?

A branch is your own version of the repository. Sometimes it is nice to work on a feature “in private” without having to relate to other developers’ codes. And sometimes you want to work on something which takes some time before it is finished.

In these cases you can create a *branch*. You can work on it on your own or in a team of colleagues. When you are happy with the the content of the branch, you can merge it back to the *trunk*.

The *trunk* is considered a safe repository. The code in the trunk should always have passed a set of tests to make sure it is stable.(<https://wiki.met.no/noresm/testlist>)

Branches let you take advantage of the svn features, but without having to worry every day that your code passes the tests.

Create a branch with (<http://svnbook.red-bean.com/en/1.7/svn.branchmerge.using.html>)

```
svn copy http://svn.example.com/repos/calc/trunk http://svn.example.com/repos/calc/  
↪branches/my-calc-branch -m "Creating a private branch of /calc/trunk."
```

or specifically for the NorESM repository with

```
svn copy https://svn.met.no/NorESM/noresm/tags/trunk2.0-1 https://svn.met.no/NorESM/  
↪noresm/branches/privateMYPORJECT_trunk2.0-1 -m "Creating a project branch of tags/  
↪trunk2.0-1."
```

Then check out the branch using `svn checkout $BRANCHURL nameOfBranchOnMyPC`

In git: First create your new branch locally and then make the remote aware of the new branch like so:

```
git checkout -b my_branch_name  
git push -u origin my_branch_name
```

..and make sure your `.gitconfig`-file is configured for doing a merge (for example):

```
[merge]  
  
tool = vimdiff  
  
[diff]  
  
tool = vimdiff
```

16.2 When should I work on a branch?

In most cases the answer is “always”. However look at it this way:

If you work directly on the trunk you have to check that your code passes the tests every time you change the code. This is cumbersome and takes a lot of time. However if you are on a branch, you can work peacefully together with your small team taking full advantage of the version control system. In the end, when you and your teammates are happy with the changes, you perform the tests and merge back to trunk.

You should obviously test your code also before you commit to a branch, but for a branch the test does not include running and analyzing long simulations.

16.3 How should I name the branch?

The NorESM branches have the following ******naming convention :

{PurposeOfLife}_{ParentTagName}.

This means that a branch name should state **why** it is created and **where** it is created from.

PurposeOfLife is a text string (allowed characters are a-z, A-Z, 0-9 and “.”) that must start with *feature*, *release*, *project* or *private*, where

- a *feature* branch is a temporary branch created to work on a complex change without interfering with the stability of /trunk (or another parent branch). Feature branches are always reintegrated. See also <http://svnbook.red-bean.com/en/1.7/svn.branchmerge.commonpatterns.html>
- a *release* branch is created if a version with frozen functionality is desired. The development on the release branch itself is limited to bug fixes, addition of forcing scenarios and other minor changes. Release branches are not reintegrated.
- a *project* branch is similar to a release branch. For some projects a specific version with small changes might be required to do specific model runs. It is important that all members of the project use the same code. If the project involves major development tasks, the project team should consider creating a *feature* branch instead. A *project* branch is likely not reintegrated.
- a *private* branch can be created if a project requires a strongly tailored version of the model, that typically is maintained by only one person. Commits to a private branch should be agreed on with the branch creator. Private branches are not necessarily reintegrated.

ParentTagName is the name of an existing noresm tag.

Examples of valid branch names are:

- featureIceActivation_trunk2.0-1: development of ice activation feature, created from tag trunk2.0-1)
- release2.0.0_trunk2.0-19 (release branch created from tag trunk2.0-19)
- release2.0.1_release2.0.0-15 (release branch created from tag release2.0.0-15)
- privateHiatusStudy_release2.0.1-3 (private branch for Ingo’s hiatus study, created from tag release2.0.1-3)
- projectEXPECT_cmip5-r143-1 (branch created for use in project named EXPECT, created from tag cmip5-r143-1)

Note that in this scheme a new branch is never branched off directly from trunk or another branch. **A tag MUST be created before creating a branch** (see section on tags below).

Note that purposeOfLife of release-branches contains version numbers. “release2.0.0” and “release2.0.1” are “purposeOfLife”. When creating release-branches, please agree on numbering with <Mats.Bentsen@uni.no>. **The version numbers in release-branches** “purposeOfLife” should not be confused with “increasingVersionNumber” used to make tag-names unique.

16.4 A note on branch / tag naming

16.4.1 Naming of feature branches and associated tags

Consider the following example: A small group decides to have a branch of their own. The group defines their branch’s purpose of life as “featureLandSurfaceModeling”.

They can now branch off from trunk and create the branch *featureLandSurfaceModeling_trunk2.0-1*. They will work happily on their branch until one day they have completed their feature. In the mean time they have tagged their branch a couple of times as *featureLandSurfaceModeling-1*, *featureLandSurfaceModeling-2*.

After the feature is completed, the branch is merged back to trunk, and following svn recommendations, the branch should now be considered dead.

The team still want their branch on which they cooperate well. They should now re-generate their branch from trunk, for example *featureLandSurfaceModeling_trunk2.0-40*. This is OK. The repository should now have two branches, but by inspection of branch names it is easy to see that *featureLandSurfaceModeling_trunk2.0-40* is the recent one and the other is a dead end.

While developing on *featureLandSurfaceModeling_trunk2.0-40*, the team decides to tag their model twice, creating the tags *featureLandSurfaceModeling-3* and *featureLandSurfaceModeling-4*.

The repository now has four tags: *featureLandSurfaceModeling-1*, *featureLandSurfaceModeling-2*, *featureLandSurfaceModeling-3* and *featureLandSurfaceModeling-4*. They originate from two different branches. However this is OK in the NorESM naming convention scheme!

16.4.2 Naming of release branches and associated tags

The PurposeOfLife string of a release branch should begin with “release” followed by .. (e.g., *release2.0.1*).

The numbers and are inherited from the tag from which the branch is created. The number is set to 0 in the special case that the parent is a trunk tag (e.g., *release2.0.0_trunk2.0-19*) and augmented if the release branch is created from an existing release branch tag (e.g., *release2.0.1_release2.0.0-5*).

For more information on and see section “How should I name the tag”.

16.5 How and when should I merge from trunk to my branch?

You can decide this for yourself. The main hypothesis is that the trunk is always stable and working, so you will not harm your branch by merging from trunk. If you are working on something which you will finally merge back to the trunk, you can merge often. Then the final merge will be easier.

Just use `svn merge ^/noresm/trunk` (The “^” means “the URL of the repository’s root directory” in newer versions of svn. In older versions you need to give full URL)

16.6 How can I merge my branch back into the trunk?

You can merge to trunk after your changed code has passed a list of tests. The tests are available here: <https://wiki.met.no/noresm/testlist>

If your code does not pass the tests, you can **not** merge your code back to the trunk

Note that in svn, **you can only merge ONE time from your branch to the trunk**, or you risk making a mess of the system! (See <http://svnbook.red-bean.com/en/1.7/svn.branchmerge.basicmerging.html> ==> “Reintegrating a branch”, note the statement **Once a –reintegrate merge is done from branch to trunk, the branch is no longer usable for further work**”).

N.B. There is a workaround to this, described in <http://svnbook.red-bean.com/en/1.7/svn.branchmerge.advanced.html#svn.branchmerge.advanced.reintegratetwice>

The merge command (from trunk) will be something like (<http://svnbook.red-bean.com/en/1.7/svn.branchmerge.basicmerging.html>)

```
svn merge --reintegrate $BRANCHURL
```

Using git, just use `git merge branchNameIWantToMergeWith`

16.7 What is a tag?

A tag is a version of the model which should be considered “frozen”. It makes sense to make a new “tag” for a production system, for example a specific version which should be used for many runs.

In svn notation, there is no difference between a branch and a tag. It is just a question of naming convention. Recommended convention is that tags are branches saved under the “tags” subdirectory and a branch is saved under the “branches” directory.

- **In NorESM tags are considered frozen, and people are not allowed** to do development on tagged versions.

Create the tag with a command like (<http://svnbook.red-bean.com/en/1.6/svn.branchmerge.tags.html>)

```
svn copy http://svn.example.com/repos/calc/trunk http://svn.example.com/repos/calc/tags/
↪release-1.0 -m "Tagging the 1.0 release of the 'calc' project."
```

or specifically **for** the NorESM repository **with**

```
svn copy https://svn.met.no/NorESM/noresm/trunk https://svn.met.no/NorESM/noresm/tags/
↪trunk2.0-2 -m "Creating new tag of trunk."
```

16.8 When should I create a tag?

You should create a tag in the following cases:

- When you want to create a branch! **Always tag the model first and then create the branch from the tag.** (If a tag has already been made at the point where you want to branch off, you don’t need to create a new tag!!)
- A “released” version, a version which has been properly tested and which we recommend other users to run
- A version which has been used for some specific paper (maybe you want to do more runs after referee comments)

16.9 What tags exist?

All tags are in <https://svn.met.no/viewvc/noresm/noresm/tags/>. Have a look there to find out which tags exist before you create a new tag. In order not to break the naming convention scheme, you need to know which tags exist already!

16.10 How should I name the tag?

Tags created from trunk have the naming convention:

```
trunk{MainModelVersion}.{MinorModelVersion}-{IncreasingVersionNumber}
```

Tags created from branches have the naming convention:

```
{BranchPurposeOfLife}-{IncreasingVersionNumber}
```

Every time a release branch is created from trunk, then `MinorModelVersion` is increased and `IncreasingVersionNumber` reset to 1. If no new release branch is created, then `MinorModelVersion` stays the same and `IncreasingVersionNumber` is increased.

- **Important: If a user wants to create a tag from a freshly** created branch, then `IncreasingVersionNumber` should be set to 0 (e.g., *featureMicomDevelopment-0*). Be aware, however, that creating a tag from a freshly created branch results in tag duplication, e.g., if the branch is *featureMicomDevelopment_trunk2.0-19* then *featureMicomDevelopment-0* will be identical to *trunk2.0-19*.

`MainModelVersion` and `MinorModelVersion` are global counters while `IncreasingVersionNumber` is local to the trunk or a specific branch.

Examples are:

- trunk2.0-1
- trunk2.0-2
- featureIceActivation-1
- release2.0.0-1
- release2.0.0-2
- release2.0.1-1 (associated branch created from tag release2.0.0-2)
- privateHiatusStudy-1 (owned by Ingo)
- privateKatlaStudy-1 (owned by Øyvind)

Note how this is consistent with the branch naming scheme. You actually need to create a tag in order to give your branch a proper name!

16.11 Tagging noresm0 and noresm1 branches?

NorESM0 and NorESM1 had a quite random naming convention for branches where branch names involved “noresm” and “revision numbers”. For example a NorESM1 branch name is “noresm-ver1_cmip5-r112/”. When tagging these, we need to know what is “purposeOfLife” of this branch. People tend to talk about these branches as the “112-version” or the branch “noresm-ver1_cmip5-r143/” as the “143 version”.

Therefore, even though it is confusing it is proposed here to use “cmip5-r112” or “cmip5-r143” as purposeOfLife for these branches. So tags created from these branches would be called “cmip5-r112-1”, “cmip5-r112-2”, “cmip5-r143-1”, “cmip5-r143-2” etc.

- **purposeOfLife of NorESM1 branches is therefore whatever follows** after “noresm-verX-” in the branch name

16.12 Case 1

A user has performed a control simulation using a tagged NorESM version. He/she wants to use this control simulation as baseline for several new implementations. How should the user create/update working branches without having to worry about changes in trunk that can affect the result?

This is straight forward. The user creates several branches based on the original tag. Following the naming convention scheme, they all have a different “PurposeOfLife” but they have the same ParentTagName. The combination is a unique branch name for all the new implementations.

Importantly, “merge from trunk” - which is required to be able to reintegrate the branch into trunk - has to be postponed until after evaluation experiments for the new implementations have been performed.

16.13 Case 2

A user group wants to develop one model component without constantly having to worry about changes in other model components.

A solution is to bundle new implementations for a specific component by creating a super branch (e.g., *featureMicomDevelopment_trunk2.0-19*).

The super branch can then be used to create ordinary feature branches for the individual implementations (e.g., *featureNewMixingScheme_featureMicomDevelopment-0*).

Reintegration into trunk is done in two steps: First, feature branches from the individual implementations are reintegrated into the super branch. Last, the super branch is reintegrated into trunk.

16.14 Case 3

- A tagged NorESM version has been used for the production of a certain simulation. Part of the simulations has to be rerun with extended diagnostic capability (e.g., with U10 output). How can I commit the extended diagnostic capability to svn?*

One can imagine that something like this happened:

- The originally tagged version was made from a branch called “release2.0.0_trunk2.0-45”
- The experiment was run with “release2.0.0-3”

When the update is needed there are two possibilities:

This is the normal (and simplest) case:

Update the “release2.0.0_trunk2.0-45” branch with the changes needed. After the development is done on “release2.0.0_trunk2.0-45” brand, tag this as “release2.0.0-4” and do the experiments.

In the mean time, someone has done a lot of bugfixing in “release2.0.0_trunk2.0-45”, so the latest version of “release2.0.0_trunk2.0-45” is significantly different from “release2.0.0-3” and you are afraid including the bugfixes will change the original results. You can not include all the bugfixes before doing the extra simulations so using the latest version of “release2.0.0_trunk2.0-45” is not an option.

In this case, create a new branch: “release2.0.1_release2.0.0-3”, update it and tag it “release2.0.1-1”. **Note that in this case, you have created a new branch with a different “purposeOfLife”: “release2.0.1” is different from “release2.0.0”.**

===== SVN messages =====

16.15 What should be stated in the commit message?

- **Always state the JIRA issue associated with the work! This makes** the code changes pop up in JIRA!: For example:

```
svn commit -m "NE-100: These are the fixes needed to solve this issue"
```

(see <https://wiki.met.no/noresm/usingtheissuetracker>)

The commit message should include a concise description of the committed changes.

Furthermore, it should indicate whether the changes preserve bit-reproducibility. If this information is omitted, then one should assume that bit-identity is broken.

Examples 1:

```
Added new compset COMPSETNAME, bit identical compared to revision r
```

Examples 2:

```
Added new diagnostics in cloud.F90, bit identical compared to revision r
```

16.16 What should be stated in the copy message of a branch?

A more elaborated “purpose of life” than indicated by the branch name can be provide as svn message when creating the branch. If a corresponding jira issue exists, then the text provided to the issue track can be reused here.

16.17 What should be stated in the copy message of a tag?

Ideally, the commit message of a tag should provide a complete change history relative to the last tag. This information can be easily extracted from the svn viewer.

Example 1:

```
Change history: trunk-2.0 -> trunk-2.1 (r190 -> r198)
```

```
Revision 198
```

```
changes made in r198...
```

```
Revision 193
```

```
changes made in r193...
```

```
Revision 190
```

```
changes made in r190...
```

Example 2:

```
Change history: trunk-2.3 -> featureMicomDevelopment-1 (r200 -> r205)
```

```
Revision 205
```

```
changes made in r205 to branch featureMicomDevelopment...
```

```
Revision 202
```

```
changes made in r202 to branch featureMicomDevelopment...
```

```
Revision 200
```

```
changes made in r200 to branch featureMicomDevelopment...
```

SUBVERSION HOW-TO FOR NORSTORE

17.1 How it works

NorStore uses the `svnserve` server solution to take local svn repositories online.

On the NorStore node `noresg.norstore.no`, a `svnmerge` demon is running as a background process. The demon serves repositories that are stored in NorStore's project work space in `/projects/NS2345K/svn`. The `svn` repositories each receive an URL of form

```
svn://noresg.norstore.no/<repository name>
```

17.2 Prerequisites

To create and manage a repository, you need a user account at NorStore which is member of the `ns2345k` project.

The creator of a repository has the full flexibility to grant remote read/write access to external users. Once the repository is create, the use of the repository does not require a NorStore account. The repository creator can define `svn` users (consisting of a user-name with corresponding password) which in general have no relation to NorStore user accounts.

To start and stop the `svn` server, your NorStore user must in addition have access to `noresg.norstore.no`. Currently, this group contains following people: Alf Grini, Ingo Bethke, Thierry Toutain and Martin King.

17.3 Start/stop server

Log on to `noresg.norstore.no` via `ssh`.

To take all `svn` repositories online, do

```
svnserve -d -r /projects/NS2345K/svn --log-file /projects/NS2345K/svn/svnserve.log --pid-  
↪file /projects/NS2345K/svn/svnserve.pid
```

An `svnserve` demon has now been started as a background process on `noresg.norstore.no`. The process id is logged in `/projects/NS2345K/svn/svnserve.pid`.

To take all repositories offline again, do

```
kill `cat /projects/NS2345K/svn/svnserve.pid`
```

17.4 Creating a new repository

Log on to *norstore.uio.no* via ssh.

Change directory to */projects/NS2345K/svn*.

Create a new svn repository with

```
svnadmin create testrepo
```

where *testrepo* is to be replaced with the repository name.

The new repository is now set up in */projects/NS2345K/svn/testrepo*

17.5 Customizing access rights

Edit *testrepo/conf/svnserve.conf* for general customisation of access rights. Important: Make sure to remove all leading blanks when activating an option.

The default is read/write access for authenticated users and no access for anonymous.

To limited access to read for authenticated users, change

```
# auth-access = write
```

to

```
auth-access = read
```

To grant anonymous read, change

```
# anon-access = none
```

to

```
anon-access = read
```

The users of the repository are defined in *testrepo/conf/passwd* with user name and password, e.g.

```
[users]
  harry = harryssecret
  sally = sallyssecret
  guestuser = friendly
```

The user customisation is activated in *svnserve.conf* by uncommenting

```
# password-db = passwd
```

to

```
password-db = passwd
```

Further fine tuning of access rights can be done in *testrepo/conf/authz*. E.g.,

```
[/]  
harry = rw  
guestuser = r
```

gives *harry* read/write access but limits the access of *guestuser* to read-only. The *authz* customisation is activated in *svnserve.conf* by uncommenting

```
# authz-db = authz
```

to

```
authz-db = authz
```

17.6 Remote access

After taking the repository online, the URL of the repository *testrepo* is `svn://noresg.norstore.no/testrepo`

To checkout the repository, do

```
svn co svn://noresg.norstore.no/testrepo
```

Change directory to *testrepo*

```
cd testrepo
```

Create a dummy file and mark it for adding

```
echo test > README  
svn add README
```

Commit the repository

```
svn commit -m "my commit message"
```


**UNCERTAIN PARAMETERS (WHICH CAN BE DISCUSSED) IN THE
AEROSOL MODEL**

Parameter	Meaning	Where in code
sticking coefficients	How easy is it for H ₂ SO ₄ to condense on aerosol modes	condtend.F90 (and AeroTab for the look-up tables)
size distribution/total dust emission	Tune on fraction of dust going to different modes (back to aerocom estimates??): how much is emitted in DST_A2 relative to the DST_A3 tracer	oslo_dust_intr.F90 oslo_dust_intr.F90
om_to_oc	How much more “OM” do we get when “OC” is emitted	mo_srf_emission.F90, mo_extfrc.F90
below cloud scavenging coeffs	Below cloud scavenging coefficients	aerosoldef.F90
size of “life-cycle species”	what is the effective size of “condensate”, “coagulate”	aerosoldef.F90

NORSTORE TAPE STORAGE

19.1 Basic use

Instructions for basic use of the tape resources are found on NorStore's [homepage](#).

19.2 Advanced use

A collection of high-level tape tools is available in `/projects/NS2345K/tools`.

19.2.1 noresm2tape

`noresm2tape` copies the output of a NorESM case from NorStore's disk area to NorStore's tape resource. The output folder should be organised in the standard CCSM way (e.g., atmospheric output is expected in 'atm/hist'). If that is not the case, please use the command `disk2tape` instead.

- **IMPORTANT:**

It is highly recommended to run the script in background using the `nohup` command, e.g.,

```
nohup /projects/NS2345K/tools/noresm2tape &
```

Make sure not to forget the “&” at the end of the line. After submitting the script with `nohup`, it is safe to log out. You will receive a notification email when the transfer to tape is completed. ``

Run `/projects/NS2345K/tools/noresm2tape -h` to print detailed instructions:

```
Usage: noresm2tape
Example: noresm2tape /scratch/ingo/mycase /tape/NS2345K/cases/mycase auto replica
Purpose: Copies NorESM case output from disk to tape.
Description: <path on disk> and <path on tape> must be full path names (see example).
             <chunk size> must be one of: auto, all, 1, 10, 100, or 1000 (numbers
             ↪ indicate simulation years per chunk)
             <replication flag> must be either 'replica' - resulting in two tape copies -
             ↪ or 'noreplica'.

             The output folder <path on tape> will be created - with parent
             directories if necessary - and must not exist before running this script.
```

(continues on next page)

(continued from previous page)

The **input** data **is** staged **in** form of tar-chunks **in** /scratch/ingo/noresm2tape. The tar-chunks are removed **from** **scratch** after successful transfer to tape.

Checksums **for all input** files are computed **and** stored **in** a separate checksum file. The checksums are used to verify the tar-chunks.

19.2.2 disk2tape

disk2tape copies a folder from NorStore's disk area to NorStore's tape resource. In contrast to noresm2tape, the data folder can have any structure and content.

IMPORTANT: It is highly recommended to run the script in background using the nohup command, e.g.,

```
nohup /projects/NS2345K/tools/disk2tape &
```

Make sure not to forget the "&" at the end of the line. After submitting the script with nohup, it is safe to log out. You will receive a notification email when the transfer to tape is completed.

Run

```
/projects/NS2345K/tools/disk2tape -h
```

to print detailed instructions:

```
Usage: disk2tape [none|gzip|bzip2] [replicate]
```

```
Example: disk2tape /projects/NS2345K/hirlam /tape/NS2345K/hirlam
```

Purpose: Copies a folder from disk to tape.

Description: <path on disk> must be the absolute path to the input folder.
<path on tape> will be created - with parent directories if necessary - and must not exist before running this script.

gzip or bzip2 compression will be performed if the third argument is set to "gzip" or "bzip2", respectively. The default is no compression.

If the fourth argument is set to "replicate" then a second copy of the data will be stored on another physical tape medium.

The path of the additional copy differs from the path of the first copy by that /tape is replaced with /replica.

The input data is staged in form of tar-chunks in /scratch/\$USER/`basename \$0`. The tar-chunks are removed from scratch after success transfer to tape.

The size of a single tar-chunk is about 15Gb if no compression is applied. If compression is applied, the size of the tar-chunks can be up to 60Gb as a compression factor 4 is assumed.

(continues on next page)

(continued from previous page)

Checksums for all input files are computed and stored in a separate checksum file. The checksums are used to verify the tar-chunks.

19.2.3 tape2disk

tape2disk copies a folder that had been put on tape with either noesm2tape or disk2tape. All tar-archives found in the top-level folder are unpacked while tar-files stored within tar-files are not inflated.

Run

```
/projects/NS2345K/tools/tape2disk -h
```

to print detailed instructions:

```
Usage: /projects/NS2345K/tools/tape2disk <path on tape> <path on disk>
```

```
Example: /projects/NS2345K/tools/tape2disk /tape/NS2345K/hirlam /scratch/ingo/hirlam
```

```
Purpose: Retrieves a folder from tape and unpacks all tar-archives contained in it.
```

```
Description: <path on disk> will be created if it does not exist.
```

19.2.4 listontape

listontape lists all files that are archived in a tape folder.

Run

```
/projects/NS2345K/tools/listontape -h
```

to print detailed instructions:

```
Usage: /projects/NS2345K/tools/listontape <path to data-directory on tape>
```

```
or
```

```
  /projects/NS2345K/tools/listontape <path to checksum file on tape>
```

```
Example:
```

```
  /projects/NS2345K/tools/listontape /tape/NS2345K/hirham_nobackup/BCM
```

```
or
```

```
  /projects/NS2345K/tools/listontape /tape/NS2345K/hirham_nobackup/BCM/BCM.md5.tar
```

```
Purpose: Lists the names of files that are archived in a given directory on tape.
```

```
Description: The script relies on the existence of a checksum file, with extension md5.tar, that is stored in the archive directory.
```

```
  If exactly one checksum file exists then directory path is sufficient as input argument. However, if two or more checksum file exist then the absolute path to the checksum file has to be provided.
```


NORSTORE RESEARCH DATA ARCHIVE: GUIDELINES FOR INGESTION OF NORESM OUTPUT

The recommendations presented on this page are based on discussion between Alok, Mats and Ingo. Please feel free to comment and modify them.

The access point to NorStore's Research Data Archive is <http://archive.norstore.no>

For testing purposes, the alternative site <http://archive-test.norstore.uio.no> should be used.

20.1 Metadata recommendations

20.1.1 BibliographicCitation

Specify "<http://cmip-pcmdi.llnl.gov/cmip5/terms.html>" as value.

20.1.2 Coverage

Please follow DCMI recommendations when specifying coverage.

Box (<http://dublincore.org/documents/dcmi-box>)

If coverage is global, use:

- northlimit=90
- southlimit=-90
- westlimit=-180
- eastlimit=180
- units=signed decimal degrees

(leave rest unspecified)

Period (<http://dublincore.org/documents/dcmi-period> <http://www.w3.org/TR/NOTE-datetime>)

Specification of start- and end-year required (optionally, month and day):

- start=YYYY[-MM-DD]
- end=YYYY[-MM-DD]
- scheme=W3C-DTF

The value of *scheme* should be always set to W3C-DTF. The value of *name* can be left blank.

20.1.3 Description

Recommended content in description:

- long version of title, e.g. “Norwegian Earth System Model version 1 (medium resolution) output prepared for the CMIP5 pre-industrial control experiment.”
- citation information (see below)
- reference to CMIP5 experimental design (http://cmip-pcmdi.llnl.gov/cmip5/docs/Taylor_CMIP5_design.pdf)
- forcing agents (link to <http://search.es-doc.org>) (**INGO: skip?**)
- initialisation (parent experiment, branch time) (**INGO: skip?**)

NorESM1-M citation:

- **Bentsen, M., Bethke, I., Debernard, J. B., Iversen, T., Kirkevåg, A., Seland, Ø., Drange, H., Roelandt, C., Seierstad, I. A., Hoose, C., and Kristjánsson, J. E.:** The Norwegian Earth System Model, NorESM1-M – Part 1: Description and basic evaluation of the physical climate, *Geosci. Model Dev.*, 6, 687-720, doi:10.5194/gmd-6-687-2013, 2013.

NorESM1-ME citation:

- **Tjiputra, J. F., Roelandt, C., Bentsen, M., Lawrence, D. M., Lorentzen, T., Schwinger, J., Seland, Ø., and Heinze, C.:** Evaluation of the carbon cycle components in the Norwegian Earth System Model (NorESM), *Geosci. Model Dev.*, 6, 301-325, doi:10.5194/gmd-6-301-2013, 2013.

Mapping of CMIP experiment long/short names

10- or 30-year run initialized in year XXXX	decadalXXXX
volcano-free hindcast initialized in year XXXX	noVolcXXXX
prediction with 2010 volcano	volcIn2010
pre-industrial control	piControl
historical	historical
historical extension	historicalExt
other historical forcing	historicalMisc
mid-Holocene	midHolocene
last glacial maximum	lgm
last millennium	past1000
RCP4.5	rcp45
RCP8.5	rcp85
RCP2.6	rcp26
RCP6	rcp60
ESM pre-industrial control	esmControl

(continues on next page)

(continued from previous page)

ESM historical	esmHistorical
ESM RCP8.5	esmrcp85
ESM fixed climate 1	esmFixClim1
ESM fixed climate 2	esmFixClim2
ESM feedback 1	esmFdbk1
ESM feedback 2	esmFdbk2
1 percent per year CO2	1pctCO2
abrupt 4xCO2	abrupt4xCO2
natural-only	historicalNat
GHG-only	historicalGHG
AMIP	amip
2030 time-slice	sst2030
control SST climatology	sstClim
CO2 forcing	sstClim4xCO2
all aerosol forcing	sstClimAerosol
sulfate aerosol forcing	sstClimSulfate
4xCO2 AMIP	amip4xCO2
AMIP plus patterned anomaly	amipFuture
aqua planet control	aquaControl
4xCO2 aqua planet	aqua4xCO2
aqua planet plus 4K anomaly	aqua4K
AMIP plus 4K anomaly	amip4K

Boundary conditions

piControl:

- Prescribed atmospheric concentrations of pre-industrial well mixed gas: Carbon Dioxide
- Unperturbed Pre-Industrial Land Use
- Prescribed concentrations **or** emissions of pre-industrial natural aerosols
- Prescribed concentrations **or** emissions of pre-industrial natural aerosol precursors
- Prescribed atmospheric concentration of pre-industrial short lived (reactive) gas.↵
↵species
- Prescribed concentrations **or** emissions of pre-industrial short lived (reactive) aerosol.↵
↵species
- Prescribed atmospheric concentrations of pre-industrial well mixed gases: excluding CO2

20.1.4 Label

If the dataset matches a NorESM case then specify the case name as label, e.g., NAER1850CNOC_f19_g16_06.

20.1.5 Rights holder

If in doubt, specify

- Organization: *Norwegian Climate Centre*
- Organization acronym: *NCC*
- Organization web-page: *gfi.uib.no/EarthClim*
- Contact email: *noresm-ncc@met.no* (**INGO: NEED TO FIND BETTER SOLUTION FOR EMAIL ADDRESS**)

20.1.6 Title

We recommend to use following formula:

<MODEL ACRONYM> <PROJECT> <OFFICIAL EXPERIMENT ACRONYM> (<experiment number in Taylor_
→document>) r<realisation number>[<realisation letter>] <output type>

Examples:

- NorESM1-M CMIP5 historicalExt (3.2) r1 raw output
- NorESM1-ME CMIP5 rcp85 (4.2) r1a raw output
- NorESM1-ME CMIP5 rcp85 (4.2) r1 cmor-processed output

<output type> should be a brief and general description of the type of output, e.g. raw output or cmor-processed output. Information on data format etc will be specified elsewhere.

<realisation letter> can be used if an experiment is composed of several cases – e.g., NorESM1-ME CMIP5 rcp85 r1 is associated with the cases NRCP85AERCNOF_f19_g16_01 and NRCP85AERCNOF_f19_g16_02 – and one intends to create a dataset for each cases.

Mapping between CMIP5 experiment acronyms and NorESM1-M cases:

piControl	r1 N1850AERCNOF_f19_g16_CTRL_02
1pctCO2	r1 N1850RMAERCNOF_f19_g16_02
historical	r1 N20TRAERCNOF_01
esmControl	r1 N1850AERBPRP_f19_g16_02
esmHistorical	r1 N20TRAERCNOCBPRP_f19_g16_01
esmrcp85	r1 NRCP85AERBPRP_f19_g16_03
esmFdbk1	r1 N1850RMAERCNOF_f19_g16_RAD_02
esmFixClim1	r1 N1850RMAERCNOF_f19_g16_BGC_02
rcp26	r1 NRCP26AERCNOF_f19_g16_01 (2006-2060), NRCP26AERCNOF_f19_g16_02 (2061-→2101)
rcp45	r1 NRCP45AERCNOF_f19_g16_02
rcp6	r1 NRCP60AERCNOF_f19_g16_01 (2006-2050), NRCP60AERCNOF_f19_g16_02 (2051-→2101)
rcp85	r1 NRCP85AERCNOF_f19_g16_01 (2006-2044), NRCP85AERCNOF_f19_g16_02 (2045-→2100)

Mapping between CMIP5 experiment acronyms and NorESM1-ME cases names:


```

piControl      r1 N1850AERCNOC_f19_g16_CTRL_02
lpctCO2        r1 N1850RMAERCNOC_f19_g16_02
historical     r1 N20TRAERCNOC_01
esmControl     r1 N1850AERBPRP_f19_g16_02
esmHistorical  r1 N20TRAERCNOCBPRP_f19_g16_01
esmrcp85       r1 NRCP85AERBPRP_f19_g16_03
esmFdbk1       r1 N1850RMAERCNOC_f19_g16_RAD_02
esmFixClim1    r1 N1850RMAERCNOC_f19_g16_BGC_02
rcp26          r1 NRCP26AERCNOC_f19_g16_01 (2006-2060), NRCP26AERCNOC_f19_g16_02 (2061-
↪2101)
rcp45          r1 NRCP45AERCNOC_f19_g16_02
rcp6           r1 NRCP60AERCNOC_f19_g16_01 (2006-2050), NRCP60AERCNOC_f19_g16_02 (2051-
↪2101)
rcp85         r1 NRCP85AERCNOC_f19_g16_01 (2006-2044), NRCP85AERCNOC_f19_g16_02 (2045-
↪2100)

```

20.1.7 Project

For CMIP5 output, specify *Integrated Earth System Approach to Explore Natural Variability and Climate Sensitivity (EarthClim)*

20.1.8 Conforms to

If in doubt, specify “*Climate and Forecast (CF) metadata conventions*”

20.1.9 Provenance

In case the output has been compressed, specify “*gzip compression of restart files and conversion of history output to compressed NetCDF-4 format*” and state the time stamp of the last compressed restart file.

Please add additional provenance entries in case further manipulations have been performed on the output.

20.2 Metadata example

[Link to this section.](#)

Parameter	Value	Comment
Title	NorESM1-M CMIP5 historicalExt (3.2) r2 raw output	
Created on	29/Oct/2011	Ingo: time stamp of the last restart folder (i.e, time experiment was completed)
Category	Simulation	
State	Raw	
Domain	Natural Sciences	
Field	Earth Sciences	
Creator	Norwegian Climate Center (NCC)	
Contributor	Alok Kumar Gupta (Alok.Gupta@uni.no)	Ingo: NorStore defines the contributor as the person who puts in the metadata
Data Manager	Alok Kumar Gupta (Alok.Gupta@uni.no)	
Rights Holder	Norwegian Climate Center (NCC) (Ingo.Bethke@uni.no)	Ingo: need to find better solution for the email address
Access Rights	Public	Ingo: our processed CMIP5 is public, so we might as well make the raw data public
Label	NRCP85AERCN_f19_g16_02	Ingo: decided to use the experiment “case name” as label
BibliographicCitation	http://www.geosci-model-dev.net/6/687/2013/gmd-6-687-2013.html	Ingo: the <i>BibliographicCitation</i> value has to be an URL. We decided to add additional citation information to Description
Project	Integrated Earth System Approach to Explore Natural Variability and Climate Sensitivity (EarthClim)	Ingo: specify time stamp of last compressed restart file
Conforms to	Climate and Forecast (CF) metadata conventions	
Provenance	gzip compression of restart files and conversion of history output to compressed NetCDF-4 format	
Coverage	Box: Southlimit=-90, Northlimit=90, Westlimit=-180, Eastlimit=180, Uplimit=20000, Downlimit=-9000, Units=signed decimal degrees, Zunits=m; Period: start=2006-01-01, end=2012-12-31, scheme=W3C-DTF	Ingo: use of DCMI standard makes it easy for external servers to interpret the coverage information

Description

Norwegian Earth System Model version 1 (medium resolution) output prepared for the CMIP5 historical extension experiment with forcing scenario RCP8.5.

Citation: Bentsen, M., Bethke, I., Debernard, J. B., Iversen, T., Kirkevåg, A., Seland, Ø., Drange, H., Roelandt, C., Seierstad, I. A., Hoose, C., and Kristjánsson, J. E.: The Norwegian Earth System Model, NorESM1-M - Part 1: Description and basic evaluation of the physical climate, *Geosci. Model Dev.*, 6, 687-720, doi:10.5194/gmd-6-687-2013, 2013.

Technical details: Production machine: Cray XT3 in Bergen (hexagon)

Model source: https://svn.met.no/viewvc/noresm/noresm/branches/noresm-ver1_cmip5-r112
Model revision number: 112 Model components: atmosphere=CAM4;
ocean=MICOM; land=CLM; sea ice=CICE Horizontal resolution:
atmosphere/land=1.9x2.5 degree; ocean/sea ice=~1 degree Output
frequency: monthly + daily + 6-hourly + 3-hourly as requested by CMIP5

Experiment type: fully coupled Initialisation: branched from CMIP5 historical simulation r2 (N20TRAERCN_f19_g16_02) at 2006-01-01 Changing forcing agents: prescribed GHG concentration changes; aerosol emissions for SO4, POM and BC (see Kirkevåg et al. 2013) Tuning parameters changed relative to the host model CAM4: rhminl=0.90 (0.91 in CAM4) reduced RH threshold for formation of low stratiform clouds; critrtp=5.0 mm/day (0.5 mm/day in CAM4) maximum precipitation rate for suppression of autoconversion of cloud water; r3lc=14 um (10 um in CAM4) critical mean droplet volume radius for onset of autoconversion Other comments: -

External references:

http://cmip-pcmdi.llnl.gov/cmip5/docs/Taylor_CMIP5_design.pdf
(experimental design) <http://search.es-doc.org> (model system, boundary conditions, experiments, etc) <http://noresg.norstore.no> (Norwegian ESGF portal with post-processed CMIP5 data)
http://www.geosci-model-dev.net/special_issue20.html (NorESM special issue)
<http://www.cristin.no/as/WebObjects/cristin.woa/wo/18.Profil.29.25.2.3.3.7>
(link to national publication database)

SIMPLE ONLINE SHARING

21.1 Share data online

For simple data sharing, use the script `**/projects/NS2345K/tools/wwwpublish**`: Usage:

```
`` wwwpublish ***** ``  
`` or ``  
`` wwwpublish ***** ``
```

Example:

```
`` wwwpublish /projects/NS2345K/noresm/thredds/CORE2/AMOC CORE2-AMOC ``  
`` or ``  
`` wwwpublish /projects/NS2345K/noresm/noresm2cmor.tbz noresm2cmor ``
```

Purpose:

```
`` Publish data online at \ `` http://ns2345k.web.nird.no <http://ns2345k.web.nird.no>`_`` ``
```

Description:

```
`` The input path must be absolute and start with /projects/NS2345K.``
```

```
`` Only regular files are considered. Symbolic links are skipped.``  
`` ``
```

```
`` All date files must have unique names as the folder structure is ``  
`` flattened during publication. ``
```

```
`` Reading permissions of the data files have to be open to everyone. ``
```

```
`` However, reading permissions of the folders containing the files ``  
`` can be closed. ``  
`` ``
```

```
`` ***** should be one word and not contain any special ``  
`` characters other than “-” and “_”. ``
```

The script provides following:

```
`` *local path to a folder that contains hard-links to the data files ``
`` *URL that points to the online-catalogue where the shared files are exposed ``
`` *download-URL to a wget-script that can be used to download the entire dataset (note that execution permis-
sions have to be changed prior to running the script) ``
```

NOTE: The sharing method uses hard-links. This has the advantages that no additional disk space is used and that the sharing happens instantaneous. A removal of the hard-links will unshare the data, but will not affect the original files.

21.2 Unshare data

If you don't want to share a dataset anymore, use the script `**/projects/NS2345K/tools/wwwunpublish**`: Usage: `wwwunpublish`

Example: `wwwunpublish CORE2-AMOC`

Purpose: Unpublish a dataset from NorStore's www-server

NOTE: Unsharing data will not affect the original data.

Data sharing via the Norwegian ESG data portal

21.2.1 IMPORTANT: The portal is currently offline for maintenance and its operation is expected to be resumed sometime in October. For an alternative data sharing method see previous section "Simple data sharing".

```
{{ :noresm:noresg.png?nolink&512 |}}
```

21.3 Simplified publishing (unofficial ESG publication)

All users of the NorStore project ns2345k have the possibility to share their data via NorStore's ESG data portal <http://noresg.norstore.no>.

Requirements are:

```
`` * the data has to reside in /projects/NS2345K (same as /norstore_osl/projects/NS2345K)``
`` * all files must be regular (i.e., no symbolic links) ``
`` * file names must all be unique (folder hierarchy is flattened during publication)``
`` * reading permissions of data files must be open to everyone (does not apply to their parent folders)``
```

Note that the data sharing solution uses hard links to avoid additional use of disk resources. The original data is not touched/modified.

21.3.1 Publish a simple dataset to NorStore's ESG data portal

For simple data sharing with ESG support, use the script `**/projects/NS2345K/tools/esgpublish**`: Usage: `esgpublish [unlisted]`

```
`` or``
`` esgpublish \ \ `` ``\ \ `` [unlisted]
```

Example: `esgpublish /projects/NS2345K/noresm/thredds/CORE2/AMOC CORE2-AMOC`

```
`` or``
`` esgpublish /projects/NS2345K/noresm/noresm2cmor.tbz noresm2cmor unlisted ``
```

Purpose: Publish data via NorStore's online data portal <http://noresg.norstore.no>

Description: The input path must be absolute and start with `/projects/NS2345K` or

```
`` /norstore_osl/projects/NS2345K. ``
```

```
`` Only regular files (i.e., no symbolic links) are considered. ``
`` ``
```

```
`` All date files must have unique names as the folder structure is ``
`` flattened during publication. ``
```

```
`` Reading permissions of the data files have to be open to everyone, ``
`` but reading permissions of the folders containing the files can be ``
`` closed. ``
```

```
`` ``
`` **** should be one word and not contain any special ``
`` characters other than "-" and "_". ``
`` ``
```

```
`` If a third argument is specified and set to "unlisted" then the ``
`` dataset will be served through the file server but not published ``
`` to the portal. ``
```

21.3.2 Remove a simple dataset from NorStore's ESG data portal

To unpublish a dataset that previously has been published with `esgpublish`, use the script `**/projects/NS2345K/tools/esgunpublish**`: Usage: `esgunpublish`

Example: `esgunpublish CORE2-AMOC`

Purpose: Unpublish a dataset from NorStore's online data portal <http://noresg.norstore.no>

21.4 Standard publishing (official ESG publication)

Several projects (e.g, CMIP, GeoMIP, etc) require special post-processing and ESG publishing procedures.

Please contact Ingo Bethke ([ingo.bethke\[at\]uni.no](mailto:ingo.bethke@uni.no)) or Alf Grini ([alfg\[at\]met.no](mailto:alfg[at]met.no)) for assistance.

EXISTING PROJECTS

expect

CHAPTER
TWENTYTHREE

REFERENCES

SEARCH

- search

BIBLIOGRAPHY

- [1] M. Bentsen, I. Bethke, J. B. Debernard, T. Iversen, A. Kirkevåg, Ø. Seland, H. Drange, C. Roelandt, I. A. Seierstad, C. Hoose, and J. E. Kristjánsson. The norwegian earth system model, noresm1-m – part 1: description and basic evaluation of the physical climate. *Geoscientific Model Development*, 6(3):687–720, 2013. URL: <https://www.geosci-model-dev.net/6/687/2013/>, doi:10.5194/gmd-6-687-2013.
- [2] T. Iversen, M. Bentsen, I. Bethke, J. B. Debernard, A. Kirkevåg, Ø. Seland, H. Drange, J. E. Kristjánsson, I. Medhaug, M. Sand, and I. A. Seierstad. The norwegian earth system model, noresm1-m – part 2: climate response and scenario projections. *Geoscientific Model Development*, 6(2):389–415, 2013. URL: <https://www.geosci-model-dev.net/6/389/2013/>, doi:10.5194/gmd-6-389-2013.
- [3] A. Kirkevåg, T. Iversen, Ø. Seland, C. Hoose, J. E. Kristjánsson, H. Struthers, A. M. L. Ekman, S. Ghan, J. Griesfeller, E. D. Nilsson, and M. Schulz. Aerosol–climate interactions in the norwegian earth system model – noresm1-m. *Geoscientific Model Development*, 6(1):207–244, 2013. URL: <https://www.geosci-model-dev.net/6/207/2013/>, doi:10.5194/gmd-6-207-2013.
- [4] J. F. Tjiputra, C. Roelandt, M. Bentsen, D. M. Lawrence, T. Lorentzen, J. Schwinger, Ø. Seland, and C. Heinze. Evaluation of the carbon cycle components in the norwegian earth system model (noresm). *Geoscientific Model Development*, 6(2):301–325, 2013. URL: <https://www.geosci-model-dev.net/6/301/2013/>, doi:10.5194/gmd-6-301-2013.
- [1] M. Bentsen, I. Bethke, J. B. Debernard, T. Iversen, A. Kirkevåg, Ø. Seland, H. Drange, C. Roelandt, I. A. Seierstad, C. Hoose, and J. E. Kristjánsson. The norwegian earth system model, noresm1-m – part 1: description and basic evaluation of the physical climate. *Geoscientific Model Development*, 6(3):687–720, 2013. URL: <https://www.geosci-model-dev.net/6/687/2013/>, doi:10.5194/gmd-6-687-2013.
- [2] T. Iversen, M. Bentsen, I. Bethke, J. B. Debernard, A. Kirkevåg, Ø. Seland, H. Drange, J. E. Kristjánsson, I. Medhaug, M. Sand, and I. A. Seierstad. The norwegian earth system model, noresm1-m – part 2: climate response and scenario projections. *Geoscientific Model Development*, 6(2):389–415, 2013. URL: <https://www.geosci-model-dev.net/6/389/2013/>, doi:10.5194/gmd-6-389-2013.
- [3] A. Kirkevåg, T. Iversen, Ø. Seland, C. Hoose, J. E. Kristjánsson, H. Struthers, A. M. L. Ekman, S. Ghan, J. Griesfeller, E. D. Nilsson, and M. Schulz. Aerosol–climate interactions in the norwegian earth system model – noresm1-m. *Geoscientific Model Development*, 6(1):207–244, 2013. URL: <https://www.geosci-model-dev.net/6/207/2013/>, doi:10.5194/gmd-6-207-2013.
- [4] J. F. Tjiputra, C. Roelandt, M. Bentsen, D. M. Lawrence, T. Lorentzen, J. Schwinger, Ø. Seland, and C. Heinze. Evaluation of the carbon cycle components in the norwegian earth system model (noresm). *Geoscientific Model Development*, 6(2):301–325, 2013. URL: <https://www.geosci-model-dev.net/6/301/2013/>, doi:10.5194/gmd-6-301-2013.
- [5] A. Kirkevåg, A. Grini, D. Olivíé, Ø. Seland, K. Alterskjær, M. Hummel, I. H. H. Karset, A. Lewinschal, X. Liu, R. Makkonen, I. Bethke, J. Griesfeller, M. Schulz, and T. Iversen. A production-tagged aerosol module for earth system models, osloaero5.3 – extensions and updates for cam5.3-oslo. *Geoscientific Model Development*

opment, 11(10):3945–3982, 2018. URL: <https://www.geosci-model-dev.net/11/3945/2018/>, doi:10.5194/gmd-11-3945-2018.