Australian Government
Bureau of Meteorology

The Centre for Australian Weather and Climate Research
A partnership between CSIRO and the Bureau of Meteorology

CSIRO

ACCESS Post-Processor Version 1.0

Peter Uhe, Timothy Hume, Mark Collier

CAWCR Technical Report No. 058

November 2012

www.cawcr.gov.au

ACCESS Post-Processor Version 1.0

Peter Uhe[1], Timothy Hume[2], Mark Collier[1]
.

[1]The Centre for Australian Weather and Climate Research, *CSIRO Marine and Atmospheric Research*
[2]The Centre for Australian Weather and Climate Research, *Australian Bureau of Meteorology*

CAWCR Technical Report No. 058

November 2012

National Library of Australia Cataloguing-in-Publication entry : (pbk)

Author:	Peter Uhe, Timothy Hume, Mark Collier.
Title:	ACCESS Post-Processor Version 1.0.
ISBN:	978-1-922173-24-9 (Electronic Resource PDF)
Series:	CAWCR technical report, no. 058.
Notes:	Includes bibliographical references and index.

Enquiries should be addressed to:

Peter Uhe
CSIRO Marine and Atmospheric Research
Private Bag No 1
ASPENDALE VIC 3195

Peter.Uhe@csiro.au

Copyright and Disclaimer

Contents

List of Tables

ABSTRACT

An important component of the submission of the ACCESS coupled model data to the Coupled Model Intercomparison Project Phase 5 (CMIP5) is publishing a comprehensive set of the CMIP5 requested data and ensuring the data meets the stringent CMIP5 format and quality requirements. CMIP5 specifies a standard for model output where each parameter is stored in a single NetCDF file and includes additional meta-data. These requirements are designed to make analysis of the data as straight forward as possible.

This document describes the ACCESS Post-Processor (APP), and how it is used. The APP has been designed to automate the production of ACCESS CMIP5 data, using a database that contains information about each of the requested parameters for any experiment. Parameters are processed via a “wrapper” tool which selects entries in the database then processes them through the APP. The database approach allows simple and flexible user control of the processing.

The ACCESS model is able to output data that is not requested for CMIP5 but is of scientific interest to researchers. The APP is also able to process these data into the CMIP5 format.

1 INTRODUCTION

The Australian Community Climate and Earth System Simulator (ACCESS) coupled model (Bi et al, 2012), is comprised of the Met Office Unified Model (UM; Collins, 2008), the MOM4 ocean model (Griffies et al., 2004), the CICE4.1 sea-ice model (Hunke and Lipscomb, 2010), coupled using the OASIS coupler (Redler et. at. 2010). The ocean and sea-ice component (ACCESS-OM) and coupling strategy are described in Bi and Marsland, (2010).

There are two different versions of ACCESS used in CMIP5- ACCESS1.0 and ACCESS1.3. The main difference relevant to post-processing is that ACCESS1.0 uses the UM's default land scheme MOSES, while in ACCESS1.3, this is replaced by the Community Atmosphere Biosphere Land Exchange (CABLE) model (Kowalczyk et. al. 2006).

One of the main roles of the ACCESS model is to produce a submission to CMIP5 (Taylor et al, 2011b), for use in the IPCC Fifth Assessment Report (AR5). CMIP5 has a prescribed set of experiments and output parameters requested for those experiments. It also requires that submitted data are in NetCDF format, with one parameter per file, and meet particular metadata conventions (Taylor et al, 2011a). These requirements ease the process of comparing the CMIP5 models.

The APP uses the Climate Model Output Rewriter tool (CMOR2; Doutriaux et al. 2012) to convert ACCESS model output to the CMIP5 format. CMOR has several computer language interfaces. It was decided to use the Python interface because it compliments current ways of local data analysis and is the interface most regularly tested by the CMOR developers.

The APP system is split into three main components. Firstly there is an SQLite database that is used to drive the other components of the APP. The database contains information that is required to generate the CMIP5 data sets and to describe the experiments to be post-processed. Secondly there is the APP core script, which takes information collected from the database. Using this information, it converts ACCESS output to CMIP5 files using CMOR2. Finally, the APP Wrapper is located between the database and APP. It automates selecting database entries and passing this information to the APP core script to produce the corresponding files.

In addition to directly converting variables from the model output into CMIP5 format, the APP is able to produce diagnostic parameters that are calculated from one or more parameters in the raw model output. These derivations are defined in configuration files for the database, and can use external functions allowing completely general calculations to be performed. It is also possible to define new CMOR tables containing variables that are not requested for CMIP5 (unsolicited variables) and then configure the APP system to process these variables into CMIP5 format. The steps to produce unsolicited variables using the APP are described in section 4.1.

Another feature of the APP system is that it may be run for incomplete experiments, allowing the diagnostics to be used for evaluation of the model during development.

This document describes the APP, and is a guide on its use.

Section 2 covers the installation of the APP from the subversion repository and the dependencies that are required to run the APP. It then describes how to configure environment variables and the location of ancillary files.

Section 3 describes the use of scripts to convert the UM output to NetCDF format.

Section 4 is a quick start guide on running the APP.

Section 5 describes the components of the APP. The main database tables are also described in this section.

Section 6 gives an overview of how variables are specified in the “Champions files”.

2 INSTALLATION

The APP consists of Python scripts that are available from a subversion (SVN) repository. The subversion repository is currently located on the National Computing Infrastructure National Facility (NCI NF) access_tools subversion repository.

2.1 Subversion

To checkout the subversion repository first you will need to gain the correct permissions. Email Peter Uhe or Martin Dix to arrange this (you will need an existing account with NCI NF).

Then run the unix command:

```
svn checkout https://trac.nci.org.au/svn/access_tools/post_processor/trunk/
```

If you are using Windows you can use an application such as TortoiseSVN to check out the SVN repository.

2.2 Dependencies

To run the APP on the NCI NF machines dcc or vayu, you will need to load the following modules:

python/2.6, netcdf/3.6.3, python/2.6-matplotlib, ncl/6.0.0, hdf5/1.8.7

These modules are loaded with the system command “module load”. e.g. module load python/2.6. Equivalent packages will need to be installed on other machines where the APP is to be employed.

In addition to this, CDAT is required. Because of issues with the standard CDAT installation on the NCI NF machines, custom CDAT executables have been compiled. The CDAT_LOCATION environment variable is used to point to the CDAT library.

To run the APP on a machine other than at NCI NF, note that the following python libraries are pre-installed on NCI NF and may need to be installed.

Scientific.IO.Netcdf, cmor, matplotlib

2.3 Setting Environment Variables

Environment variables are used to enhance the portability of the APP between different machines. These need to be set for the APP to function correctly.

APP_OUTPATH:
Location that the APP output files are written to. e.g. /short/p66/pfu599/

CDAT_LOCATION:
Location of the Python CDAT library. Required if CDAT is not in the Python path
For dcc use: /home/599/pfu599/lib/python/cdat_lite-6.0rc2-py2.6-linux-x86_64.egg

Optional Environment variables:
APP_CHAMPIONS_DIR:
Location of directory containing champions files (see Section 5.3.3). Default: trunk/database/champions

APP_EXPERIMENTS_TABLE:
Name of csv file that specifies the setup of experiments. This file must be in the folder trunk/database/. Defaults to experiments.csv

APP_DATABASE:
Location of database file. Defaults to trunk/database/app.db

2.4 Ancillary Files

The ancillary files should be put in the directory: ${APP_OUTPATH}/CMIP5/ancillary_files, where $APP_OUTPATH is the environment variable defined in Section 2.3. If an ancillary file is changed, then the code in app_funcs.py will need to reflect the changed filename or variable names within the file.

To set this up, you can copy the ancillary files from
dcc:/ /projects/p66/pfu599/CMIP5/ancillary_files/

(or request these files from Peter Uhe if you lack permissions).

If the model configuration changes different files will need to be generated to replace the current ancillary files. The functions that use the ancillary files may also need to be modified to handle this.

The following functions in app_funcs use ancillary files to specify information about the access grid, or time invariant quantities:

getBasinMask(), getOrog(), get_vertices(), oceanFrac(), iceTransport(), areacella(), getHybridLevels() (this doesn't use ancillary files, but the hybrid levels are hard coded into the function.)

3 PRE (POST)-PROCESSING

Before running the APP on ACCESS model output, the output needs to be arranged to follow the directory structure described in Section 6.1, which has atmospheric (atm), ocean (ocn) and sea-ice (ice) diagnostics in separate folders.

In addition, the raw UM output must first be converted into NetCDF format. The file names expected by the APP are as follows, so the um2netcdf scripts have been configured to produce output in this form:

Monthly:	<experiment_directory>/atm/netCDF/*_mon.nc
Daily:	<experiment_directory>/atm/netCDF/*_dai.nc
6 hourly:	<experiment_directory>/atm/netCDF/*_6h.nc
3 hourly:	<experiment_directory>/atm/netCDF/*_3h.nc

A future version of the UM may allow for the direct writing of NetCDF files alleviating the need for this processing step. The following section describes the scripts used to perform this conversion.

3.1 um2netcdf.py

This script is found at: svn:trunk/um2netcdf/um2netcdf.py,

The script um2netcdf.py reads in a single UM data file, and outputs a NetCDF file with the corresponding data. The UM output are binary files in which each variable has a unique identifier called a stash code. The um2netcdf script uses the file trunk/um2netcdf/stashvar.py to map the stash codes to the variables. It also can add basic metadata that may be missing in the UM file. If a new output variable is added that is not listed in stashvar.py, this may need to be updated for the output to be converted to NetCDF format correctly.

Running the um2netcdf.py requires the CDAT_LOCATION environment variable to be set. There have been problems using standard CDAT libraries on NCI NF computers; this allows us to specify the location of libraries that are known to work correctly with all output.

3.2 runUm2netcdf.py

This script is found at: svn:trunk/um2netcdf/runUm2netcdf.py, It is a wrapper script calling um2netcdf.py for post-processing multiple files in a directory.

It can read environment variables to determine the files to be converted:
ifile= Input files (Wildcards are used to specify multiple files)
syear=Start year (extracted from the file name)
eyear=End year (extracted from the file name)

The runUm2netcdf.py script recognises the type frequency of data from the identifiers used in the file name. The conventions used by the ACCESS coupled model are:

pa → _mon.nc

pe → _dai.nc
pi → _6h.nc
pj → _3h.nc

The script has been set up to be submitted as a batch job to PBS. An example submission script is at svn:trunk/um2netcdf/run.sh. Note that this script relies on the naming conventions used by the ACCESS coupled model and will need to be modified to work for different output.

4 RUNNING THE APP

The main steps in running the APP for an ACCESS experiment, once the APP has been correctly installed are:

1. Run the um2netcdf tool and make sure the model output conforms to the required directory structure.

2. Set-up the 'experiments' table for the configuration of the model run (see Section 5.3.1). An example file is svn:trunk/database/experiments.csv

3. Populate the database by running the file: trunk/database/database_manager.py

4. Run the app_wrapper (see Section 5.2), specifying which parameters to process (either by modifying the file, or by using environment variables). This can be submitted to the queue using a batch script such as trunk/run.sh

5. For publishing of the data locally or publicly via the Earth System Grid (ESG), additional steps are required beyond the running of the APP.

   - Running drs_tool, to change the directory structure to comply with DRS (Taylor, 2011a). (Appendix A).

   - Run the QC tool (Quality Control level 2 check) to report warnings/errors regarding problems found in the CMIP5 output. (Appendix B).

4.1 Modifications / Customisation of the APP

Alongside Step 2, the Champions tables, and the grids table may need to be modified to reflect changes to the output parameters or diagnostics, or if calculations of different diagnostics are required (see Section 6).

Unsolicited data can be produced by defining new CMOR tables. e.g. trunk/cmip5-cmor-tables/Tables/CMIP5_AmonExtras. The simplest way to produce these variables is to follow the format in one of the official CMIP5 tables. Note that the line 'product: output' must be modified to 'product: unsolicited' for these extra CMIP5 tables. APP champions tables then need to be produced for the unsolicited data in the same way as for CMIP5 requested output (see section 6). The one modification is that the column 'implemented in post-processor' must have 'unsolicited'.

For CMIP5, some output parameters are not required for all experiments, or for all time periods of specific simulations. The database_manager only populates the file_master database table with entries corresponding to files within the requested time ranges. To change what times are specified, modifications can be made to database_manager.py before running Step 3 (explained in Section 5.4).

4.2 Post Processor Output:

The output data is written to the directory:

${APP_OUTPATH}/CMIP5/output

following the directory structure that CMOR uses, e.g.

<$APP_OUTPATH>/CMIP5/output/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/tas/r1i1p1

The plots generated by the app_wrapper are written to:

${APP_OUTPATH}/CMIP5/plots/<local_experiment_name>/<CMIP5_Table>

The output log files that are generated when submitting a PBS job via the trunk/run.sh script are written to:

<$APP_OUTPATH>/CMIP5/job_output/<local_experiment_name>/<CMIP5_Table>

5 COMPONENTS OF THE APP

5.1 APP Core

The core of the Access Post Processing system is app.py (Access Post-Processor Python script). This is a Python script that may either be called from the command line or by another Python program. It takes a number of command line arguments (or a dictionary of these arguments when called as a Python routine) specifying the data needed to post-process a specific variable and output a single CMIP5 file.

It is possible to use the APP without using the database and APP Wrapper, but this is not recommended. There are a large number of input arguments, and while not all of them are necessary for every application, the APP Wrapper provides a more reliable and convenient interface.

app.py uses the CMOR2 library to write out the NetCDF files. This uses the CMOR CMIP5 variable tables to fill in a large proportion of the output file's attributes automatically. Details of attributes that are specific to the set up of the ACCESS coupled model or a particular experiment need to be passed to CMOR and are given as arguments when running app.py.

Main steps in the app.py:

1. Set up global attributes of the output file
2. Find all the input files, checking their time values and make a list of files that fall within the time range of the output file.
3. For the output variable, find the axes information for the relevant dimensions. Fill in any missing information that is required (time and space bounds, correct coordinate values, units, etc.)
    - For cases where the output variable has different dimensions to the variable used in the input files, modify the dimensions as appropriate (remove axis, create new axis, change values). Some of these operations are specified using the "axes_modifier" argument (see Section 6.3).
4. Write out the values of the variable from each of the input files to the output file.
    - In some cases there will need to be calculations performed using multiple input variables to give the value of the output variable. This is handled by calling functions in the file app_funcs.py, the function being specified by the input argument "calculation", set up in the Champions table (see Section 5.3.3).

5.2 APP Wrapper

The file app_wrapper.py is a wrapper script for the APP, which selects entries in the database and then calls app.py to initiate processing.

It can be called with environment variables set:

N: Maximum number of database rows (entries) to process.

- If this is not set, the number of rows to process is specified by the variable 'n' in the function main()

Table (CMIP5 table) and Exp (local_experiment name) are required together.

- When called with these environment variables set, the APP Wrapper processes up to N rows that have status='unprocessed' and meet the criteria that, cmip_table= 'Table' and local_experiment='Exp' in the 'file_master' table of the database.

  – If 'Table' and 'Exp' aren't set, a default command to select rows of the file_master database table will be executed. This is specified by the variable 'select' in the function main(). An example of the select string is:

select = ' frequency ==\'mon\' and experiment_id==\'historical\' and status==\'unprocessed\' '

This will process up to n files of monthly data from the historical run that hasn't already been processed.

After it selects the database rows, the APP Wrapper updates the status of these rows to 'processing'. Since the APP Wrapper by default selects only rows with status='unprocessed', this acts as a lock on those database rows.

After calling app.py, the app Wrapper updates the database, depending on the outcome of the processing. For each output file created, a diagnostic plot is generated of the relevant parameter it represents.

Flags (variables in main() which can be modified):

overRideFiles: specifies whether to skip processing if the file exists already (default=False).

plot: specifies whether to plot after processing or not (default=True).

5.3 Database

The database consists of three configuration tables, which are used to populate the 'file_master' table which drives the APP. There is an 'experiment' table, containing all the experiment configuration information. There is a 'grids' table that contains mapping information between dimensions and grid points. This is also used to determine the file block size (maximum number of years to have in one file). Lastly there is a 'champions' table which represent the information about producing CMIP5 parameters from the ACCESS model output.

Each of these tables is generated by reading in comma-separated files. The champions table is generated from a set of champions files, separated into files by CMIP5 table (Taylor 2011).

5.3.1 Experiment Configuration Table

The experiments database table provides information to process the experiment and metadata about each experiment (Table 1). Each CMIP5 experiment ensemble member (Taylor et al. 2011b), requires a separate entry in the experiments table.

The start and end years restricts the time range of the post processing which may not be the full extent of the model output.

The file that the database reads to set up the experiments configuration table can be specified by setting the environment variable $APP_EXPERIMENTS_TABLE before running the database_manager.py. The file $APP_EXPERIMENTS_TABLE is located relative to the database folder e.g. 'experiments_example.csv' will point to the file trunk/database/experiments_example.csv

Table 1. Experiments Database Table Fields.

Field	**Description**
experiment_id	experiment id e.g. piControl, historical
experiment_directory	base directory of model output
start_year	start year of model run
end_year	end year of model run
realization	realisation number of the experiment
initialization	initialisation number of the experiment
physics_version	physics version number of the experiment
local_experiment name	A name used locally to identify the experiment
model name	model name. e.g. ACCESS
forcing	string describing experiment forcings present

parent_id	experiment ID of parent
parent_rip	RIP of parent
branch time	time in parent experiment that the current experiment was branched
land model	which land model was used: CABLE or MOSES
Notes	any additional notes

Note that the experiment_id and forcing strings are restricted to using prescribed values as defined in Taylor2011a.

5.3.2 Grids Table

The main purpose of the grids table is to specify the amount of data in each file (maximum number of years). This is determined from the spatial dimensions of the parameter and the frequency of the data. This chunking of parameters into multiple files is required to prevent files becoming too large. The 'max years' is chosen to minimize the number of files while keeping the size within a reasonable limit. The grids table also provides the information needed to produce an estimate of the file sizes.

Table 2. Grids Database Table Fields.

Column	**Description**
frequency	frequency: mon, day, 6hr, 3hr, monclim
dimensions	string describing dimensions (Table 3)
gridpoints	number of spacial grid points (includes points for pseudo levels, basins etc)
max years	The number of years one file will hold for these parameters
Notes	notes
file_size (MB)	(not used by the app or database), useful for determining maximum years

Table 3. Dimension Identifiers.

dimension_id	**description**
Scalar	scalar time series
2Datmos	Atmos grid- 145*192
2Docean	300*360 ocean grid
3Docean	Ocean grid with 50 vertical levels

3Dm	Atmos grid with 38 (model) height levels
3Dp17	atmos grid with 17 pressure levels
oline	dimension for different ocean straits
2Db_rho	Ocean grid: Lat+ rho depth(80 levels) +3 basins
2Db 1Db	Ocean grid: Lat+ depth +3 basins Ocean grid: Lat+3 basins
3Dtile9	Atmos grid with 9 tiles for moses
3Dsoil4	Atmos grid with 4 soil levels for moses
3Dtile17	Atmos grid with 17 tiles for cable
3Dsoil6	Atmos grid with 6 soil levels for cable
3Dp8	Atmos grid with 8 pressure levels
3Dp3	Atmos grid with 3 pressure levels

5.3.3 Champions Table:

“Champions” is the name of the database table used to provide the relevant data needed to process each variable. This name was given for historical reasons to the files produced under direction of the champions group (see acknoledgements). This table can be thought of as a configuration table of CMIP5 variables and their calculations.

Each CMIP5 table has one (or more) Champions files associated with it (see Table 4). In the cases where variables in the same CMIP5 table are requested for different time periods in experiments, they can be put into different champions files. The database table has an additional column champions_group which specifies which file the variables has come from. For example day_all contains fields from the CMIP5 table 'day', that are requested for all times, but parameters in day_limited are only requested for certain times of each experiment (because they are lower priority or take up more disk space).

Different champions groups are also used for parameters which require different calculations between the CABLE and MOSES versions of ACCESS. The Lmon CMIP5 table is split up between Lmon_both, Lmon_cable, and Lmon_moses. Lmon_both contains parameters requiring the same treatment in both CABLE and MOSES. Lmon_cable and Lmon_moses contain parameters that are handled differently for the two land schemes.

Table 4. List of Champions Files.

CMIP5 table	Champions file (champions group has '.csv' removed)
3hr	3hr_limited.csv
3hr	3hr_limited_cable.csv
3hr 6hrLev	3hr_limited_moses.csv 6hrLev_limited.csv
6hrPlev	6hrPlev_limited.csv
Amon	Amon.csv
Amon	Amon_limited.csv
LImon	LImon_both.csv
LImon	Limon_cable.csv
LImon	Limon_moses.csv
Lmon	Lmon_both.csv
Lmon	Lmon_cable.csv
Lmon	Lmon_moses.csv
OImon	Oimon.csv
Oclim	Oclim.csv
Omon	Omon.csv
Aero	aero.csv
Day	day_all.csv
Day	day_limited.csv
Day	day_limited_cable.csv
Day	day_limited_moses.csv
Fx	fx.csv
Fx	fx_moses.csv
Fx	fx_cable.csv

For experiments with different diagnostics, these Champions files can be modified. The environment variable $APP_CHAMPIONS_DIR specifies the directory containing the Champions files, allowing different sets of Champions files to be used (for example, different champions files may be used for processing output from the Australian Community Ocean Model, AusCOM, Bi et al 2010). The format to specify variables using the Champions tables is explained in Section 6.

5.3.4 File Master Table

The File Master table is a table that encapsulates the data required to generate the CMIP5 files. One row in this table corresponds to one CMIP5 file and each column corresponds to one of the arguments that are passed to app.

Some of the variables in the File Master table are not used by the app.py, but may be used by the app_wrapper to specify a specific set of entries to process. E.g. Entries with status=unprocessed can be selected but status is not passed to app.py.

Table 5. File Master Database Table Fields.

Field	**Description**
experiment_id	Experiment I.D. e.g. piControl, historical etc.
model_id realization	Model I.D. e.g. ACCESS1-0, ACCESS1.3 Realisation number of ensemble
initialization_method	Initialization method number of ensemble
physics_version	Physics version number of ensemble
infile	String of input files
outpath	Base of output directory. e.g. /projects/p66/pfu599/
file_name	Expected name of output file
vin	List of input variables
vcmip	Name of CMIP5 variable to process
cmip_table	CMIP5 table of variable.
realm	Realm of variable (not used by app.py)
frequency	Frequency of variable sampling
tstart	Start time of output file (year)
tend tref	End time of output file (year) Not used, (the reference time is determined from input files)
priority	Used for information about which variables to publish (not used by app.py).
status	String specifying the processing status of the variable. e.g. unprocessed, processed. (not used by app.py)
file_size	Estimated size of output file
dimensions	Dimension identifier, see Table 3 (not used by app.py)
local_experiment	Local experiment identifier (not used by app.py)
calculation	String used to calculate vcmip from vin

axes_modifier	String to specify modifications in the axes between vin and vcmip
in_units	Override the output unit to a specified string
positive	Override the positive attribute in the output.

5.4 Database Manager

The APP database is populated in the database_manager.py script.

It reads in the Champions files, and experiment table and grids table and writes them into tables in the database app.db. It then uses these tables to create a list of entries in the database, each entry corresponding to one file of CMIP5 output.

To determine the list of files, the database_manager takes the experiment start, end times and other experiment data, as well as the "chunking size" of variables on different grids (prescribed in the grids table for each dimension_id). It loops over each variable in the champions database table and populates the file_master database table with entries corresponding to files within the time periods requested for CMIP5 (see standard output document; Taylor, 2011).

5.4.1 Modifying database_manager.py

The database_manager.py script has a function populate(), which governs population of the file_master table with data from the other tables. It doesn't create entries for all variables and for all times though, as some variables that may take up a large amount of disk space are only requested to be produced for certain times in CMIP5. The Champions files have been split up into files with different "champions_groups", which may be requested for particular times.

The functions to add rows to the file_master use either:

populateRows(rows,opts,cursor):

-Takes list of rows in Champions database table, loops over each row, loops over all times in an experiment and add rows into the file_master table.

populateLimited(rows,opts,startLim,endLim,cursor):

-Same as populateRows, except only adds rows into file_master for times between startLim and endLim.

By adding in SQlite commands to select entries of the database followed by a call to populateRows/populateLimited, you can modify which parameters are produced for what times.

The following functions (called within function populate()) are examples that select database rows from the champions database table, for different variables and call one of the above functions to populate the file_master database table.

populate_unlimited(cursor,opts)

-populates parameters requested for all times (calls populateRows())

populate_day_limited(cursor,opts)

-populates parameters for 'day' CMIP5 table, requested for limited set of times (calls populateLimited())

There are other functions similar to populate_day_limited to populate the database for other CMIP5 tables (populate_6hrLev, populate_6hrPlev, populate_Oclim,populate_3hr)

5.5 APP Functions

The file app_funcs.py contains a set of functions which are used by other parts of the APP. The functions in app_funcs.py serve a few different purposes. The first set of functions specify grid information that may be read from ancillary files. Other functions specify the calculations of specific parameters where the derivation is not a simple calculation that can be expressed in a single line. There are a few other functions used by app.py or app_wrapper such as the plotting function.

6 SPECIFYING VARIABLES IN CHAMPIONS FILES

The following section describes the purpose of different columns in the champions configuration files. The champions configuration files are comma-separated files that determine the configuration of the post-processor for each variable. The fields in the champions files are passed into the champions database table, with the addition of the cmip_table and champions_group, which are determined from the name of the champions file (e.g. cmip_table=Lmon and champions_group=Lmon_moses for variables in the file Lmon_moses.csv).

Table 6. Champions Configuration File.

Column	**Description**
cmip variable	Variable name in CMIP5 table
definable in access	If the variable can/will be produced
implemented in post processor	Has this variable been implemented by the post processor
dimensions	String describing the dimensions of the variable output
access variable name	List of variables in ACCESS coupled model output used to produce the CMIP variable (list is separated by spaces)
access file location	String describing the path to files containing the access variables
realm	modelling realm for the cmip variable
calculation	string of python code to calculate the output data
override units	string providing the correct units of the input data after the calculation
axes override	string describing modifications to variable dimension axes
Positive	string describing the positive attribute of the variable ('up' or 'down')
Notes	extra notes about the variable

The columns: 'definable' and 'implemented in post-processor' are not used by the post-processor itself but are useful information which can be used when populating the database or selecting elements to process in the app_wrapper.

6.1 ACCESS File Location

The ACCESS file location column specifies the path to the model output files, relative to the “experiment_directory” in the experiments table.
Wildcards are used to encapsulate multiple files (for different times),
E.g. “/atm/netCDF/*_3h.nc” will expand the input file list for all the 3 hourly atmospheric files in the experiment directory. It will point to the files:
(<experiment_directory>/atm/netCDF/*_3h.nc)

The APP is set up to use the following file layout:

UM output:

Monthly:	/atm/netCDF/*_mon.nc
Daily:	/atm/netCDF/*_dai.nc
6 hourly:	/atm/netCDF/*_6h.nc
3 hourly:	/atm/netCDF/*_3h.nc

Sea Ice:

Monthly:	/ice/iceh.????-??.nc
Daily:	/ice/iceh_day.????-??.nc

Ocean:

Monthly:	/ocn/ocean_month.nc-*
Monthly scalar:	/ocn/ocean_scalar.nc*
Daily:	/ocn/ocean_daily.nc-*

It also assumes that the files are given ordered timestamps in the file names.

6.2 Derivations

Derivations are made using the list of access variables, available as variable 'var' which is a list of the data arrays matching “access_variable_name”, in the input file. The array of time values is also available as variable 'times'.

A simple example is a summation of two variables. The CMIP5 variable ‘clw’ is derived from the sum of ‘clw1’ and ‘clw2’ in the ACCESS coupled model output.

The definition clw=clw1+clw2 is represented in the following way in the Champions table:

Table 7. Calculation of clw.

CMIP5 variable	clw
ACCESS model variable name	clw1 clw2
Calculation	var[0]+var[1]

Calculations can also call functions that can perform arbitrary calculations and return the output data. An example of this is calculating the monthly average of the daily minimum temperatures.

Table 8. Calculation of tasmin.

CMIP5 variable	tasmin
ACCESS variable name	tasmin
ACCESS file location	/atm/netCDF/*_dai.nc
Calculation	monthAve(var[0],times)
axes modifier	day2mon

Things to note:

The function monthAve is a function in the file app_funcs.py which is imported by app.py

The ACCESS file location: /atm/netCDF/*_dai.nc is used to select the daily sampled tasmin.

The variable 'times' contains the time information of the files and is used to determine values to average over for one month.

The axes modifier 'day2mon' is used to modify the time axis from having values every day to having values every month (see Section 6.3).

6.3 Axes Modifiers.

The axes modifier is a string defining commands to modify axes. This is useful when calculations are done such as choosing a particular level from (i.e. the surface) or averaging over a dimension.

Possible commands

dropX ,dropY, dropZ, dropLev: remove axis (dropLev is used to remove axis for land surface tiles or other pseudo levels),

monClim: (monthly climatological averages),

time1: signifies time snapshots rather than time averages

day2mon: convert time values from daily to monthly

basin: add axes for ocean basins

oline: add axis for ocean lines

mosesTiles, cableTiles: specify which vegetation tiles are being used

firsttime: create a time axis that only uses the first time value

surfaceLevel: Take only the first (surface) level of the atmosphere grid

Multiple commands may be used by writing them as a list.

ACKNOWLEDGMENTS

Thank you to everyone who has contributed to the development of the APP.

Champions (configuration of champions files and determining calculations of CMIP5 variables): Harun Rashid, Julie Noonan, Lauren Stevens, Petteri Uotila, Simon Marsland.

Thanks to Holger Wolff and Harun Rashid for their help with the UM stash output.

Thanks to Harun Rashid and Martin Dix's contribution to the development of the um2netcdf scripts.

This work has been undertaken as part of the Australian Climate Change Science Program, funded jointly by the Department of Climate Change and Energy Efficiency, the Bureau of Meteorology and CSIRO.

This work was supported by the NCI National Facility at the ANU.

REFERENCES

Bi, D. and Marsland, S. 2010. Australian Climate Ocean Model (AusCOM) Users Guide, *CAWCR Technical Report*, No. 27, 74pp.

Bi, D., Dix, M., Marsland, S., O'Farrell, S., Rashid, H., Uotila, P., Hirst, T., Kowalczyk, E., Golebiewski, M., Sullivan, A., Yan, H., Franklin, C., Sun, Z., Vohralik, P., Watterson, I., Zhou, X., Collier, M., Ma, Y., Noonan, J., Stevens, L., Uhe, P., Harris, C., Griffies, S. and K. Puri, 2012. The ACCESS Coupled Model: Description and Control Climate, Submitted to the Aust. Met. And Ocean. Journal.

Collins, W.J., Bellouin, N., Doutriaux-Boucher, M., Gedney, N., Hinton, T., Jones, C.D., Liddicoat, S., Martin, G., O'Connor, F., Rae, J., Senior, C., Totterdell, I., Woodward, S., Reichler, T. and Kim, J.2008. Evaluation of the HadGEM2 model. *Met Office Hadley Centre Technical Note no. HCTN 74*, available from Met Office, FitzRoy Road, Exeter EX1 3PB http://www.metoffice.gov.uk/publications/HCTN/index.html

Doutriaux, C. and Taylor, K.E. 2012. Climate Model Output Rewriter (CMOR) Version 2.0. Available from http://cmip-pcmdi.llnl.gov/cmip5/docs/cmor2_users_guide_KET3_clean.doc

Griffies, S.M., Harrison, M.J., Pacanowski, R.C. and Rosati, A. 2004. A Technical Guide to MOM4, *GFDL Ocean Group Technical Report No. 5*, Princeton, NJ:: NOAA/Geophysical Fluid Dynamics Laboratory, 342 pp.

Hunke, E.C. and Lipscomb, W.H. 2010. *CICE: the Los Alamos Sea Ice Model Documentation and Software User's Manual Version 4.1*, 76pp. Available from http://oceans11.lanl.gov/trac/CICE/attachment/wiki/WikiStart/cicedoc.pdf

Kowalczyk, E.A., Wang, Y.P., Law, R.M., Davies, H.L., McGregor J.L. and Abramowitz, G. 2006, The CSIRO Atmosphere Biosphere Land Exchange (CABLE) Model for use in climate models and as an offline model, CSIRO Marine and Atmospheric Research Paper 013

Redler, R., Valcke, S. and Ritzdorf, H. 2010. *OASIS4* – a coupling software for next generation earth system modelling, *Geosci. Model Dev.*, 3, 87–104.

Taylor, K.E. 2011. *Standard Output Document*. 167pp. Available from http://cmip-pcmdi.llnl.gov/cmip5/docs/standard_output.pdf.

Taylor, K.E., Balaji, V., Hankin, S., Jukes, M., Lawrence, B. and Pascoe, S. 2011a. *CMIP5 Data Reference Syntax (DRS) and Controlled Vocabularies*. 14 pp. Available from http://cmip-pcmdi.llnl.gov/cmip5/docs/cmip5_data_reference_ syntax.pdf.

Taylor, K.E., Stouffer, R.J. and Meehl, G.A. 2011b. *A Summary of the CMIP5 Experiment Design*. 33pp. Available from
http://cmip-pcmdi.llnl.gov/cmip5/docs/Taylor_CMIP5_design.pdf.

APPENDIX A: DRS TOOL

The drs_tool is a tool that reads in a directory of CMIP5 files generated by CMOR and renames the files and directory structure to meet the DRS naming conventions. This is done as a last step in the post-processing before publishing.

Website and installation:

See http://esgf.org/esgf-drslib-site/ for documentation and information on installing drslib (drs_tool is part of drslib)

Running on dcc:

Use the following commands to load required modules.

```
$module use /projects/r87/public/modulefiles
$module load python/2.7.1 python-drslib
$module load python-setuptools
```

Then run the drs_tool list command. This lets you check to see if it is going to work before trying to convert to DRS format.

Example:

```
$drs_tool list -v 20111102 -R /short/p66/pfu599/drs_output/drs_output_11_29 -I /short/p66/pfu599/CMIP5/output/ cmip5.output1.CAWCR.ACCESS1-0.%.%.%
```

When you want to convert, run the same command, replacing 'list' with 'upgrade'.

This command will move all the CMIP5 files in /short/p66/pfu599/CMIP5/output/ into DRS compliant format in the folder /short/p66/pfu599/drs_output/drs_output_11_29.

note: -v 20111102 forces the version number (This allows us to match the file structure version number with the file attributes).

A script to only copy specific variables into the DRS output folder is located at trunk/move_to_published.py.

APPENDIX B: QC CHECKING

Quality control sets is important to verify the quality of the datasets to be published. There are three stages of quality control
(see https://redmine.dkrz.de/collaboration/projects/cmip5-qc/wiki for more detail):

- QC Level 1 (QC L1)

  QC L1 requires the files to meet the CF (Climate and Forecast) metadata conventions and some extra CMIP5 metadata requirements. Output formatted by CMOR should automatically satisfy these requirements.

- QC Level 2 (QC L2)

  QC L2 is passed by running a tool
  (see http://proj.badc.rl.ac.uk/go-essp/wiki/CMIP5/QualityControl#QualityControl)
  which checks metadata and runs consistency checks on the data. The results are compiled in a central database. The QC L2 check is run alongside publishing the data to the ESG.

- QC Level 3

  Assigned to files after further analysis and cross-checking of the data and metatdata. These datasets are assigned DOI's and can be cited.