Developer Manual for the Community Multiscale Air Quality (CMAQ) Modeling System

Version 5.2 (2017 Release)

Prepared in cooperation with the
Community Modeling and Analysis System
Institute for the Environment
University of North Carolina at Chapel Hill
Chapel Hill, NC

Disclaimer

The information in this Developer Manual has been funded wholly or in part by the United States Environmental Protection Agency. The draft version of this document has not been subjected to the Agency's peer and administrative review, nor has it been approved for publication as an EPA document. The draft document is currently being edited and reviewed by the Community Modeling and Analysis System Center; this content has not yet been reviewed or approved by the EPA. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

Introduction

This document is intended to describe general development practices within the CMAQ Modeling Community. The information contained should be read prior to starting a project within the CMAQ framework. Instructions can be used by EPA developers, CMAS-Center developers, or external developers. Notes specific to external developers are made where relevant.

Contributions

To contribute source code to the CMAQ Public repository, create a fork of the EPA CMAQ External GitHub repository. Prospective developers should send the following information to the EPA and CMAS-Center code maintainers.

- What is your GitHub user name? (can be acquired at github)
- Can your work be completed in a fork of the release repository?
  - If no, please explain the reason.
- What science module do you intend to develop?
- What work do you intend to contribute to CMAQ?
- Have you reviewed the development strategy including code consistency, benchmarking, configuration testing, compiler testing, model output validation, documentation and merging?
- Are you able to provide ongoing support and technical guidance for your proposed contribution?

Repository Descriptions

The CMAQ project contains several repositories. Some of the repositories are private for development efforts while some are public for users and releases. These repositories are described below.

The main CMAQ repositories are described in this section

Internal EPA Private CMAQ-Dev Repository

This will be where CMAQ development by EPA employees occurs. This is a private repository for development efforts by employees of EPA.

External EPA Public CMAQ-Release Repository

The main CMAQ release repository is located here. Users should refer to this repository for bug fixes, issues, and major releases for CMAQ.

https://github.com/USEPA/CMAQ

Users who wish to implement a new feature, and have that feature merged into the CMAQ repository should follow the instructions on code requirements and repository layout as an CMAQ developer would. Forks should be made of this release repository. Developers will use GitHub commands to add, commit, and push code to their forked repository. Once their forked version of the repository code has undergone checks for code consistency, benchmark testing, and model output validation, and documentation including release notes, they may submit a pull request to the CMAQ-Release maintainer. New features will have to undergo a code review before any merge onto the release repository. Instructions on using it to add a feature are in this (insert link) section.

External CMAS Public CMAQ-Release Repository

A fork of the main CMAQ-Release release repository that is used for testing and documenation by the CMAS Center.

Repository Layouts

The CMAQ-Dev and CMAQ-Release repository directories for the base model are all laid out as follows.

- CCTM
  - docs
    - Releaase_Notes
    - User_Manual
    - Developer_Manual
  - scripts
  - src
    - ICL
    - MECHS
    - PARIO
    - STENEX
    - areo
      - aero6
    - biog
    - beis3
    - cloud
      - acm_ae6
      - acm_ae6_kmt

- acm_ae6_mp
- acm_ae6i_kmti

* couple
  - gencoor
  - gencoor_wrf
* depv
  - m3dry
* driver
  - wrf
  - yamo
* emis
  - emis
* gas
  - ebi_cb05e51_ae6_aq
  - ebi_cb05e51_ae6nvPOA_aq
  - ebi_cb05eh51_ae6_aq
  - ebi_cb05mp51_ae6_aq
  - ebi_cb05tucl_ae6_aq
  - ebi_cb05tump_ae6_aq
  - ebi_cb6r3_ae6_aq
  - ebi_cb6r3_ae6nvPOA_aq
  - ebi_racm2_ae6_aq
  - ebi_saprc07tb_ae6_aq
  - ebi_saprc07tc_ae6_aq
  - ebi_saprc07tc_ae6nvPOA_aq
  - ebi_saprc07tic_ae6i_aq
  - ebi_saprc07tic_ae6invPOA_aq
  - ros3
  - smvgear
* grid
  - cartesian
* hadv
  - yamo
* hdiff
  - multiscale
* init
  - yamo
* par
  - mpi
  - par_noop
* phot
  - inline
  - table
* plrise

        - smoke
    * procan
      - pa
    * pv_o3
    * spcs
    * twoway
    * util
      - util
    * vadv
      - wrf
      - yamo
    * vdiff
      - acm2
- POST
  - appendwrf
  - bldoverlay
  - block_extract
  - combine
  - hr2day
  - merge_aqs_species
  - sitecmp
  - sitecmp_dailyo3
  - writesite
- PREP
  - agdust
  - bcon
  - icon
  - jproc
  - mcip
- Tutorials
- UTIL

Development Life-cycle

EPA target schedule for CMAQ includes a final stable release of the base model that is available 6 months prior to the availability of the instrumented versions of the code.

Prior to the release version, the model is released to the public as a development version that is not intended for regulatory or research application use.

The purpose of releasing the development version to the public is to give community members a deadline by which to submit their code for inclusion into the base model and to encourage the community to assist with configuration testing, compiler testing, benchmarking and verification of model output to help improve the code, scripts and documentation prior to a final release.

The 6 month advance allows time for the CMAQ base model to be fully tested and vetted prior to beginning updates to the instrumented versions that are dependent on and incrementally added to the base model package.

- Versioning
- Current Development Release
  - CMAQ 5.2
  - Alpha
  - Beta release
  - Gamma release
- Current Stable Release
  - CMAQ 5.1
    - CMAQv5.1 Base Model
    - Two-Way WRF34-CMAQ5.02
- Maintenance release
  - CMAQ 5.0.2
  - CMAQv5.0.2 Base Model
  - Two-Way WRF34-CMAQ5.02
  - Integrated Source Apportionment Method (CMAQ-ISAM)
  - Direct Decoupled Method (CMAQ-DDM)
  - Sulfur Tracking Method (CMAQ-STM)
  - Volatility Basis Set (CMAQ-VBS)
  - Advanced Plume Treatment (CMAQ-APT)

CMAQ Code Introduction

General Code Introduction

Parallelization Strategy

Contributions Life-cycle

The repositories are all hosted on github. The typical life-cycle of a project is as follows:

1. Create a design document for the project.
2. Visit appropriate repository website.
3. Create a fork of the repository.
4. Locally clone the newly created fork.
5. Create a branch within the fork, for the new feature or bug x.
6. Develop branch.
7. Push complete branch to remote fork.
8. Submit a pull request to merge branch on fork to ??

Project contributions don't have to follow this example verbatim, but this at least gives a general overview of the process. Some details related to this life-cycle will be described in the following sections.

Design Documents

Design documents are recommended for projects that contribute signficant changes to either an existing science module or to create a new capability within CMAQ. They should clearly describe the justication for the project, and the changes and impacts of the project.

Core maintainers reserve the right to deny a pull request that lacks a design document.

Developers can refer to xxx and xxx for examples of design documents that have been previously completed and approved.

Fork CMAQ-Release Code

- Obtain an account on GitHub
- visit EPA/CMAQ page
  https://github.com/USEPA/CMAQ
- in the upper right hand corner of the page is a button
  - hover over the Fork button, and you should see a Tooltip that says "Fork your own copy of EPA/CMAQ to your account
- visit your github account
  https://github.com
  - on the right are two lists
    - Repositories you contribute to
    - Your repositories
  - click on the CMAQ repository under your repositories to view your forked CMAQ-Release version
- Tips for managing your fork
  - if your fork is behind the EPA's you can bring it up to date using the following instructions
    https://help.github.com/articles/syncing-a-fork/

Locally Clone the Newly Created Fork

Follow these instructions:

- On GitHub, navigate to the main page of the repository.
- Under the repository name, click Clone or download.
- In the Clone with HTTPs section, click to copy the clone URL for the repository.
- Open a Terminal on your computer
- Change the current working directory to the location where you want the cloned directory to be made
- Type git clone, and then paste the URL you copied in Step 2.

  ```
  git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
  ```

Create a new Branch

- Create a new branch on your local machine and switch to this branch

  git checkout -b [name_of_your_new_branch]
- Change to this branch (make it your working branch)

  git checkout [name_of_your_new_branch]

Branch Strategy

As a developer, most work will be completed in a branch. This branch can be one of several types of branches. To give an overview of the current branching strategy, please see this link.

The name of a branch should be descriptive and tell where the feature addition will be. For example, if the branch is intended to implement multiple blocks, the majority of its work would take place in framework. A good name for the branch would then be framework/multiple blocks.

- review this language..
  Feature branches should only be created from the develop branch. This allows a cleaner work flow when planning releases. Any features that have been merged onto develop can be staged for release. This means, if you submit a pull request for a feature to be merged onto the CMAQ version of develop, you are giving consent for it to be included in the next coordinated release.

Release Branches

(need to review this text from MPAS developer document and compare the MPAS method to the method currently used by EPA, and then make edits here.)

One of the variable branch types that developers will see in the shared repository is a release branch. This release branch is used to prepare the code for a new release, and comes with some guidelines of use. As per the branch document listed in section XX, release branches should branch from develop. Once they are complete, they are merged into both develop and master. After they are merged into master, that commit is tagged with a new version number, and then pushed into the CMAQ-Release repository for a public release. A feature branch can be thought of as a feature freeze. Once the release branch is created, no new features can make it into the next release. The only commits that can be appended to a release branch are clean up and bug fix commits.

Due to the requirement that anything merged into develop can be released at any point in time, a group of developers can decide they want to begin the release process at any point in time. The guidelines for the release process are as follows, however the exibility of these guidelines is at the discretion of the group performing the release.

From any point in time, the soonest date that can be targeted for release is 1 month from the current date.

Once a release date is selected, it should be anounced to the CMAQ developers list to alert all core developers.

There is a minimum lag time of 2 weeks between the announcement of a release date, and the creation of a release branch. This is to allow developers to finish any projects that are close to being complete, that they would like to be part of the next release.

After the 2 weeks are up, a release branch can be created from the tip of the develop branch. The first commit to a release branch should be updating the version numbers. The maximum time a release branch is allowed to exist is 2 weeks. After the release branch has been in existence for 2 weeks, it is assumed all core maintainer groups are finished with any edits to prepare for release. At this point, the release branch is complete. The release branch is then merged into master and develop. The commit on master is tagged with the new version number. The master branch is then pushed to the CMAQ-Release repository. These guidelines lay out a minimum amount of time a release cycle should consume. However, the core group that intends to release should announce
the targeted date of release as soon as they have one selected.

Version Numbers

(review and edit!)

CMAQ will have a two number versioning system for release version, with th an alphanumeric string denoting the development version.
For example, 5.2. In this case, the first digit and second digets refers to an official release version.
5.2 alpha, 5.2 beta, 5.2 gamma refers to intermediate bug fixes of the development version.

Referring to the branching strategy, coordinated releases occur when a release branch is created of off develop and merged back into
develop and master.
While an intermediate bug fix happens when a branch is created from a master and merged back into master and develop.

The minor version (second digit) of CMAQ increments when a bug fix branch is merged into master. These merges never include feature additions to any of the cores or shared framework.

The major version (first digit) of CMAQ increments when a release branch is merged into master. A release branch includes all of the features that were present on the develop branch when it was created. The release branch persists for some period of time to allow core developers to "clean up" any issues they have prior to release. After the grace period, the release branch is merged into master and develop and the major version of CMAQ increments by 1, while the minor version is reset to 0.

Development Strategy

As developers of CMAQ, we attempt to make the code look as uniform as we can across the entire code-base. In order to enforce this, there are a set of guidelines developers should follow.

Code Consistency

(review and edit!)

- Each science module has a name and abbreviation, and for each method within each science module there is a defined abbreviation. For example:
- The science module named horizontal advection has the abreviation hadv and the available method is abbreviated as yamo.
- The science module named vertical advection has the abreviation vadv, and the available methods are abbreviated as wrf and yamo.
- All subroutines should be named in a manner which prevents namespace conflicts.
- Subroutine names should all be upper case, with underscores in place of spaces
- Variable names should be upper case, with underscores in place of spaces).
- In general, variable names should be self-descriptive (e.g. NCELLS rather than N).

Benchmark Dataset

The U.S. EPA Calnex 12km domain July 2, 2011 testing dataset is provided with the CMAQ-Dev Release. This dataset is distributed with CMAQv5.2gamma to use for benchmarking the model installation.

Testing

- Developers need to test using multiple compilers, multiple processor configurations, and singel processor configuration runs for single day to verify answers match the previous stable release, and/or that the answers are computationally and physically reasonable.
- Developers need to share results of tests and request review of the documentation prior to a merge.
- Developers of CMAQ code will need to request review from the CMAQ maintainers by submitting a pull request, and requesting a merge.

Two classes of tests:

- Compiler tests used the default benchmark configuration with different compilers and MPI configurations.

- Configuration tests used the Portland Group 15.7 OpenMPI compiler to generate executables that exercise different scientific configurations of the release software.

Compiler flags:

- PGI: -Mfixed -O3 -Mextend
- GCC: -ffixed-form -ffixed-line-length-132 -O3 -funroll-loops -finit-character=32
- Intel: -fixed -132 -O3 -override-limits -fno-alias -mp1 -fp-model precise -fp-model source -shared-intel -openmp
- In the Intel Basic Test: -fixed -132 -O3 -openmp
- In the NoOpt Tests: -O0 with extend source and fixed line length flags

Compilation Testing Manifest Table (Example)

Scenario	Compiler	netCDF	I/O API	MPI_YN(#P)	CMAQv5.1 Timing(HH:MM:SS)	CMAQ New Project Timing(HH:MM:SS)	Notes
Gfortran Serial	Gfortran version 4.8.1	4.3.3	3.1(11/15)N	N/A	8:19:51	7:35:30	UNC module gcc/4.8.1
Gfortran MVAPICH2	Gfortran version 4.8.1	4.3.2	3.1(11/15)Y (16)	mvapich2-1.7	0:45:55	0:42:40	
Intel Serial	Intel Fortran version 16.2.0	4.3.2	3.1(11/15)N	N/A	6:01:42	5:10:16	UNC module intel/16.2
Intel OpenMPI (EPA Config)	Intel Fortran v15.0.0	4.3.2	3.1(11/15)Y (16)	openMPI-1.42	0:34:27	UNC module openmpi_intel/15.0	
Intel OpenMPI	Intel Fortran v16.2.0	4.3.2	3.1(11/15)Y (16)	openMPI-1.4.2	0:35:29	UNC module openmpi_intel/16.2	
Intel MVAPICH2	Intel Fortran v16.2.0	4.3.2	3.1(11/15)Y (16)	mvapich2-1.7	0:36:34	UNC module mvapich2_intel/16.2	
Portland Serial	PG Fortran v16.1	4.3.2	3.1(11/15)N	N/A	7:33:36	6:26:31	UNC module pgi/16.1
Portland OpenMPI	PGI Fortran v15.7	4.3.2	3.1(11/15)Y (16)	openMPI-1.4.2	0:40:20	0:36:16	UNC module openmpi_pgi/15.7

Configuration Testing Manifest Table (Example)

Scenario	Description	Mechanism	Notes	Timing(16PE)H:MM:SS
Benchmark Case	Online emissions processing, inline photolysis, inline lightning from MCIP RC, no windblown dust, surface HONO, bidirectional NH3 and Hg, no potential vorticity scaling	cb05e51_ae6_aq	Done; LTNGNO InLine, LTNGPARM = N, LOG_START = 2.0	0:40:20
Halogen Chemistry	Same as Benchmark case with halogen chemistry enabled	cb05eh51_ae6_aq	Done. Turned off the diagnostic file logging.	0:47:40
No Bidi	Same as Benchmark with Hg and NH3 BiDi deactivated	cb05e51_ae6_aq	Done. set CTM_HGBIDI = N; set ABFLUX = N	0:37:21
Process Analysis	Benchmark case with IPR and IRR	ros3	Done. Switch to Rosenbrock solver because EBI solver not supported by PA module; ran with inline process analysis	0:54:10
MOSAIC	Benchmark case with MOSAIC and additional stomatal flux files activated	cb05e51_ae6_aq	Done. set CTM_MOSAIC = Y; set CTM_FST = Y	0:44:02
New Mechanism Test	Benchmark case with toluene and chlorine chemistry	cb05tucl_ae6_aq	Done.	0:40:30
Potential vorticity UTLS exchange	Benchmark case with scaling for STE	cb05e51_ae6_aq	Uncomment potvortO3 in build. Need PV variable in METCRO3D file	0:38:03
Dust	Benchmark case with dust, including new MODIS FP input	cb05e51_ae6_aq	Done. setenv CTM_WB_DUST Y; setenv CTM_ERODE_AGLAND Y; setenv CTM_WBDUST_BELD BELD3	0:38:28
Hourly NLDN	Benchmark with lightning NOx calculated using hourly bNLDN strikes	cb05e51_ae6_aq	Done; LTNGNO InLine, LTNGPARM = Y, USE_NLDN Y	0:40:18
POA Sensitivity	Benchmark with new POA mechanism	cb05e51_ae6nvPOA_aq	Done	0:34:42

Verification of Model output

m3diff

- see min, max, mean differences between two different model runs

VERDI

- absolute difference plots for multiple variables, timesteps, layers (see spatial differences)

1:1 Scatter Plots

-

Documentation

- Subroutines and modules should be appropriately documented. CMAQ code use git tags to facilitate creation of Release Notes.
- Provide an example of how the EPA creates their Release Notes

Merging Code

Submit changes to your forked repository on Github

- Use git to commit code and documentation to your fork
  - View a list of files that have been modified in your local repository

    ```
    git status
    ```
  - Add files that have been modified, to be traced in your local git repository

    ```
    git add new_module_file
    ```
  - Commit files that have been added to your local git repository

    ```
    git commit -m "new module edits"
    ```
  - Transfer the files from your local repository to your forked repository

    ```
    git push
    ```

Pull requests

- use GitHub Website to view your CMAQ-Release Fork
- go to the branch that you have committed code, example: 5.2gamma
- Submit a pull request

Copyright Information

Code Maintainers

Maintainers are developers with write access to the main developers repository. Code maintainers will be responsible for pull requests into CMAQ.

Current CMAQ github maintainers are as follows:

- Bill Hudzell, EPA Private CMAQ-Development Repository
- Ben Murphy, EPA Public CMAQ-Release Repository
- Zac Adelman, CMAS-Center Public CMAQ-Release Repository