Community Land Model Developers Guide

CESM Community Land Model Developer's Guide

The following is intended to:

- Provide guidance to developers who wish to contribute to CLM
- Promote development of easy-to-understand and easy-to-maintain CLM code
- Encourage orderly and timely integration of model improvements into CLM

The procedures and guidelines described here should be considered to be part of a "living document" which will be subject to occasional revisions and updates as procedures evolve or requirements change.

CLM Software development steps:

Initial development may be done outside of the CLM/CESM code repository, but for code modifications to be brought onto the CLM trunk a collaborative process between the developer and the CLM code management team (CMT) will need to take place. Note that all science development needs to be approved by the LMWG (see procedure here) and in some cases other affected working groups. Developers are encouraged to contact the LMWG co-chairs and/or CLM CMT (CLM-CMT@cgd.ucar.edu) at any point during the development cycle to resolve questions regarding CLM science, code development processes, or coding guidelines.

Branch Development

1. Contact LMWG co-chairs to discuss development project plans
   a. Typically, the developer should consult with the LMWG co-chairs if they have a planned development activity so that the co-chairs can help coordinate development across the working group, identify (and help mitigate against) potential for conflicting development projects, and provide feedback on near- and long-term CLM and CESM priorities that may affect the project.
   b. If appropriate, the LMWG co-chairs will recommend a software design consultation with the CLM CMT.
2. Create a branch
   a. Ask the CLM CMT to create a development branch for your project.
3. Make and check in software changes
   a. Make changes to your branch code base and check that code in. The developer will follow the code guidelines outlined here .
   b. Please read and follow the information provided in this document to check in code into your branch; Using SVN to Work with CLM Development Branches.
   c. Note that we strongly recommend that you periodically (even frequently) update your branch to the CLM trunk (see here). A recent update of your branch to the trunk will be required before final integration into CLM.
4. Run the CLM test suite
   a. Run the CLM test suite and resolve failing tests. See documentation here.
   b. If you need additional help using or interpreting the test suite, please contact the CLM CMT.
5. Submit for approval by LMWG and CLM CMT
   a. Provide a scientific summary of the impacts of your development project to the LMWG co-chairs
   b. Identify a reproducible test case for use by the CLM CMT
   c. Provide list of new datasets required and impact on CLM tools (e.g. mksurfdata_map, interpinic, build-namelist)

We strongly encourage that steps 3 and 4 are repeated frequently throughout the lifetime of the development project.

Final integration into CLM trunk (these steps performed by CLM CMT in consultation with developer)

1. Code reviewed by the CLM CMT
   a. New code will undergo a detailed review by the CLM CMT
   b. When revisions to the code are required, the CLM CMT will provide the developer with a list of revisions that should be completed by the developer and then steps 3, 4, and 5 in Branch Development should be repeated.
2. Testing
   a. Add tests for new features (i.e. take 5b from Branch Development and create a new test when appropriate)
   b. Repeat full CLM test suite and CESM test suite when appropriate.
3. Bring code to trunk or main development branch (CLM Tag and Physics Version Naming Conventions)

Design Review

Prior to undertaking new, significant CLM developments, developers are encouraged to engage with local NCAR LMWG members to discuss their prospective work. Informal design discussions can result in valuable early feedback that can often ultimately reduce the time of the development cycle.

Code Base

New CLM developments should originate from a version as close to the most recent CLM trunk tag as is reasonably possible. Using an older, outdated code base increases the complexity of the code review and merging processes, and it increases the probability of errors. Merging new developments based on a highly outdated code base into the most recent CLM trunk code is the responsibility of the developer.

CLM Code Management Team

- CLM-CMT@cgd.ucar.edu
- emails will currently go to: Erik Kluzek, Dr. Bill Sacks, and Benjamin Andre, as well as Dr. David Lawrence and Dr. Mariana Vertenstein