CCSM Research Tools: CLM4.0 User's Guide Documentation

Erik Kluzek
NCAR

Jun-21-2010

$URL: https://svn-ccsm-models.cgd.ucar.edu/clm2/branch_tags/ccsm4_0_rel_tags/ccsm4_0_rel_06_clm4_0_00/models/
$

CCSM Research Tools: CLM4.0 User's Guide Documentation

by Erik Kluzek

The user's guide to CLM4.0 which is the active land surface model component of CCSM4.0. The purpose of this guide is to instruct both the novice and experienced user, as well as CLM developers in the use of CLM4 for land-surface climate modeling.

Dedication

Dedicated to the Land Model Working Group, winners of the 2008 CCSM Distinguished Achievement Award. May you continue to collaborate together well, and continue to drive the science of land surface modeling forward with your diligent and persistent efforts.

Table of Contents

List of Examples

Acknowledgments

I want to acknowledge all of the people that helped review or edit the model documentation: David Lawrence, Samuel Levis, Keith Oleson, and Sean Swenson. Thank you for your help in catching errors, and making the document more understandable and readable. Our readers thank you as well, as now it is much easier for them to digest. Any mistakes, or errors are all mine. If you run across one of those errors, please let us know, by following the Section called *Reporting bugs in CLM4* in *Other resources to get help from*. I also want to thank Sheri Mickelson, for her work in doing perturbation analysis on bluefire and intrepid, and allowing us to use that data in this document.

Introduction

The Community Land Model (CLM) is the latest in a series of global land models developed by the CCSM Land Model Working Group (LMWG) and maintained at the National Center for Atmospheric Research (NCAR). This guide is intended to instruct both the novice and experienced user on running CLM.

The novice user should read Chapter 1 in detail before beginning work, while the expert user should read *What is new with CLM4 since CLM3.5?* and *Quickstart to using CLM4* chapters, and then use the more detailed chapters as reference. Before novice users go onto more technical problems covered in Chapter 2, Chapter 3, Chapter 4, or Chapter 5 they should know the material covered in Chapter 1 and be able to replicate some of the examples given there.

All users should read the *How to Use This Document* and *Other resources to get help from* sections to understand the document conventions and the various ways of getting help on using CLM4. Users should also read the *What is scientifically validated and functional in CLM4?* chapter to see if their planned use of the model is something that has been scientifically validated and well tested. Users that are NOT using NCAR machines or our list of well tested machines should also read the *What are the UNIX utilities required to use CLM?* chapter to make sure they have all the required UNIX utilities on the system they want to do their work.

Introduction to the CLM4 User's Guide

What is in here anyway?

Here in the introduction we first give a simple guide to understand the document conventions in *How to Use This Document*. The next section *What is new with CLM4 since CLM3.5?* describes the differences between CLM4.0 and CLM3.5, both from a scientific as well as a software engineering point of view. It also talks about differences in the configuration, namelist, and history fields. The next section *Quickstart to using CLM4* is for users that are already experts in using CLM and gives a quickstart guide to the bare details on how to use CLM4. The next *What is scientifically validated and functional in CLM4?* tells you about what has been extensively tested and scientifically validated (and maybe more importantly) what has NOT. *What are the UNIX utilities required to use CLM?* lists the UNIX utilities required to use CLM4.0 and is important if you are running on non-NCAR machines, generic local machines, or machines NOT as well tested by us at NCAR. Next we have *Important Notes and Best Practices for Usage of CLM4* to detail some of the best practices for using CLM4 for science. The last introductory chapter is *Other resources to get help from* which lists different resources for getting help with CCSM4.0 and CLM4.

Chapter 1 goes into detail on how to setup and run simulations with CLM4 and especially how to customize cases. Details of **configure** modes and **build-namelist** options as well as namelist options are given in this chapter.

Chapter 2 gives instructions on the CLM4.0 tools for creating input datasets for use by CLM, for the expert user. There's an overview of what each tool does, and some general notes on how to build the FORTRAN tools. Then each tool is described in detail along with different ways in which the tool might be used. A final section on how to customize datasets for observational sites for very savvy expert users is given as the last section of this chapter.

As a followup to the tools chapter, Chapter 3 tells how to add files to the XML database for **build-namelist** to use. This is important if you want to use the XML database to automatically select user-created input files that you have created when you setup new cases with CLM.

In Chapter 4, again for the expert user, we give details on how to do some particularly difficult special cases. For example, we give the protocol for spinning up both the CLMCN model and CLM with dynamic vegetation active (CNDV). We also review how to validate a port to a new machine using the Perturbation error growth technique. Lastly we tell the user how to use the DATM model to send historical CO_2 data to CLM.

Finally, Chapter 5 outlines how to do single-point or regional simulations using CLM4. This is useful to either compare CLM simulations with point observational stations, such as tower sites, or to do quick simulations with CLM for example to test a new parameterization. There are a couple different ways given on how to perform single-point simulations which range from simple PTS_MODE to more complex where you create all your own datasets, tying into Chapter 2 and also Chapter 3 to add the files into the **build-namelist** XML database.

In the appendices we talk about some issues that are useful for advanced users and developers of CLM. In Appendix A we give some basic background to the CLM developer on how to edit the `models/lnd/clm/bld/clm.cpl7.template`. This is a very difficult exercise and we don't recommend it for any, but the most advanced users of CLM who are also experts in UNIX and UNIX scripting.

In Appendix B we go over how to run the script **runinit_ibm.csh"** that will interpolate standard resolution initial condition dataset to several other resolutions at once. It also runs CLM to create template files as well as doing the interpolation using **interpinic**. In general this is only something that a developer would want to do. Most users will only want to interpolate for a few specific resolutions.

In Appendix C we go over the automated testing scripts for validating that the CLM is working correctly. The test scripts run many different configurations and options with CLM making sure that they work, as well as doing automated testing to verify restarts are working correctly, and testing at many different resolutions. In general this is an activity important only for a developer of CLM, but could also be used by users who are doing extensive code modifications and want to ensure that the model continues to work correctly.

Important Notes and Best Practices for Usage of CLM4

- When running with CN, it is critical to begin with initial conditions hat are provided with the release or to spin the model up following the CN spinup procedure before conducting scientific runs (see the Section called *Spinning up the biogeochemistry Carbon-Nitrogen Model (CN spinup)* in Chapter 4. Simulations without a proper spinup will effectively be starting from an unvegetated world.
- Initial condition files are provided for fully coupled BCN and offline ICN cases for 1850 and 2000 at 1deg and 2deg resolutions. The 1850 initial condition files are in 'reasonable' equilibrium. The 2000 initial condition files represent the model state for the year 2000, and have been taken from transient simulations. Therefore, by design the year 2000 initial condition files do not represent an equilibrium state. Note also that spinning the 2000 initial conditions out to equilibrium will not reflect the best estimate of the real carbon/nitrogen state for the year 2000.
- Users can generate initial condition files at different resolutions by using the CLM tool **interpinic** to interpolate from one of the provided resolutions to the resolution of interest. Interpolated initial condition files may no longer be in 'reasonable' equilibrium.
- Aerosol deposition is a required input to CLM4. Simulations without aerosol deposition will exhibit unreasonably high snow albedos.

How to Use This Document

Conventions used in the document for code and commands

This section provides the details in using CLM with the CCSM modeling system. Links to descriptions and definitions have been provided in the code below. We use the same conventions used in the CCSM documentation as outlined below.

```
Throughout the document this style is used to indicate shell
commands and options, fragments of code, namelist variables, etc.
Where examples from an interactive shell session are presented, lines
starting with > indicate the shell prompt.  A backslash "\" at the end
of a line means the line continues onto the next one (as it does in
standard UNIX shell).  Note that $EDITOR" is used to refer to the
text editor of your choice. $EDITOR is a standard UNIX environment
variable and should be set on most UNIX systems. Comment lines are
signaled with a "#" sign, which is the standard UNIX comment sign as well.
$CSMDATA is used to denote the path to the inputdata directory for
your CCSM data.

> This is a shell prompt with commands \
that continues to the following line.
> $EDITOR filename # means you are using a text editor to edit "filename"
# This is a comment line
```

What is new with CLM4 since CLM3.5?

Since CLM3.5 there have been advances in both the science and the software infrastructure. There are also new configure and namelist options as well as new history fields. In this section we will describe each of these changes in turn.

What is new with CLM4 Science?

The following aspects are changes to the science in CLM since CLM3.5.

Biogeophysics and Hydrology

Changes to CLM4 beyond CLM3.5 (Oleson et al., 2008a; Stockli et al., 2008) include updates throughout the model. The hydrology scheme has been modified with a revised numerical solution of the Richards equation (Zeng and Decker, 2009; Decker and Zeng, 2009); a revised soil evaporation parameterization that removes the soil resistance term introduced in CLM3.5 and replaces it with a so-called B formulation, as well as accounts for the role of litter and within- canopy stability (Sakaguchi and Zeng, 2009). CLM4 also includes a representation of the thermal and hydraulic properties of organic soil that operates in conjunction with the mineral soil properties (Lawrence and Slater, 2008). The ground column has been extended to ~50-m depth by adding five additional hydrologically inactive ground layers (making a total of 15 ground layers, 10 soil layers and 5 bedrock layers; Lawrence et al., 2008). An urban landunit and associated urban canyon model (CLMU) has been added which permits the study of urban climate and urban heat island effects (Oleson et al., 2008b).

Snow Model

The snow model is significantly modified via incorporation of SNICAR (SNow and Ice Aerosol Radiation) which represents the effect of aerosol deposition (e.g. black and organic carbon and dust) on albedo, introduces a grain-size dependent snow aging parameterization, and permits vertically resolved snowpack heating (Flanner and Zender, 2005; Flanner and Zender, 2006; Flanner et al., 2007). The new snow model also includes a new density-dependent snow cover fraction parameterization (Niu and Yang, 2007), a revised snow burial fraction over short vegetation (Wang and Zeng, 2009) and corrections to snow compaction (Lawrence and Slater, 2009).

Surface Datasets

The PFT distribution is as in Lawrence and Chase (2007) except that a new cropping dataset is used (Ramankutty et al., 2008) and a grass PFT restriction has been put in place to reduce a high grass PFT bias in forested regions by replacing the herbaceous fraction with low trees rather than grass. Grass and crop PFT optical properties have been adjusted according to values presented in Asner et al. (1998), resulting in significantly reduced albedo biases. Soil colors have been re-derived according to the new PFT distribution.

Biogeochemistry

The model is extended with a carbon-nitrogen biogeochemical model (Thornton et al., 2007; Thornton et al., 2009; Randerson et al., 2009) which is referred to as CLMCN. CN is based on the terrestrial biogeochemistry Biome-BGC model with prognostic carbon and nitrogen cycle (Thornton et al., 2002; Thornton and Rosenbloom, 2005). CLMCN is prognostic with respect to carbon and nitrogen state variables in the vegetation, litter, and soil organic matter. Vegetation phenology and canopy heights are also prognostic. A detailed description of the biogeochemical component can be found in Thornton et al. (2007). Note that CLM4 can be run with either prescribed satellite phenology (CLMSP) or with prognostic phenology provided by the carbon- nitrogen cycle model (CLMCN). Additionally, a transient land cover and land use change, including wood harvest, capability has been introduced that enables the evaluation of the impact of historic and future land cover and land use change on energy, water, and momentum fluxes as well as carbon and nitrogen fluxes. The dynamic global vegetation model in CLM3.0 has been revised such that the carbon dynamics (e.g. productivity, decomposition, phenology, allocation, etc.) are controlled by CN and only the dynamic vegetation biogeography (competition) aspect of the CLM3.0 DGVM is retained. The biogenic volatile organic compounds model (BVOC) that was available in CLM3.0 has been replaced with the MEGAN BVOC model (Heald et al. 2008).

Miscellaneous Changes

Several other minor changes have been incorporated including a change to the atmospheric reference height so that it is the height above zo+d for all surface types. The convergence of canopy roughness length zo and displacement height d to bare soil values as the above-ground biomass, or the sum of leaf and stem area indices, goes to zero is ensured (Zeng and Wang, 2007). Several corrections have been made to the way the offline forcing data is interpreted. The main change is a vastly improved and smooth diurnal cycle of incoming solar radiation that conserves the total incoming solar radiation from the forcing dataset. Additionally, in offline mode rather than partitioning incoming solar radiation into a constant 70%/30% direct vs diffuse split, it is partitioned according to empirical equations that are a function of total solar radiation. Finally, to improve global energy conservation in fully coupled simulations, runoff is split into separate liquid and ice water streams that are passed separately to the ocean. Input to the ice water comes from excess snowfall in snow-capped regions.

Summary of Science Changes

Taken together, these augmentations to CLM3.5 result in improved soil moisture dynamics that lead to higher soil moisture variability and drier soils. Excessively wet and unvarying soil moisture was recognized as a deficiency in CLM3.5 (Oleson et al. 2008a, Decker and Zeng, 2009). The revised model also simulates, on average, higher snow cover, cooler soil temperatures in organic-rich soils, greater global river discharge, lower albedos over forests and grasslands, and higher transition-season albedos in snow covered regions, all of which are improvements compared to CLM3.5.

What is new with CLM4 Software Infrastructure?

The following aspects are changes to the software infrastructure in CLM since CLM3.5.

Update to cpl7 and scripts.
Remove offline and cpl6 modes.
Remove support for CASA model.
Update to datm8 atmospheric data model.
Add gx3v7 land mask for T31 and fv-4x5 horizontal resolutions.
Add gx1v6 land mask for f05, f09, and f19 horizontal resolutions.
Add tx1v1 land mask and 1.9x2.5_tx1v1 horizontal resolution.
Add in 2.5x3.33 horizontal resolution.
Add in T62 horizontal resolution so can run at same resolution as input datm data.
Allow first history tape to be 1D.
Add ability to use own version of input datasets with CLM_USRDAT_NAME variable.
Add a script to extract out regional datasets.
New **build-namelist** system with XML file describing all namelist items.
Add glacier_mec use-case and stub glacier model.
Add NCL script to time-interpolate between 1850 and 2000 for fndepdat dataset, for fndepdyn version.
Make default of maxpatch_pft=numpft+1 instead of 4.
Only output static 3D fields on first h0 history file to save space.
Add new fields for VOC (Volatile Organic Compounds) on surface datasets, needed for the new MEGAN VOC model.
Add irrigation area to mksurfdata tool (NOT used in CLM yet).
Add multiple elevation class option for glaciers in mksurfdata tool (NOT used in CLM yet).
Add ascale field to land model in support of model running on it's own grid.

What are The New Configuration Options?

Describe any changes made to build system:

Change directory structure to match CCSM.
Add BGP target.
Add choice between ESMF and MCT frameworks.
Start removing #ifdef and directives that supported Cray-X1 Phoenix as now decommissioned.
Make default of maxpatch_pft=numpft+1 instead of 4 for all configurations.
By default turn on CLAMP when either CN or CASA is enabled
New SNICAR_FRC, CARBON_AERO, and C13 CPP ifdef tokens.

New options added to **configure**: More information on options to clm **configure** are given in the Section called *More information on the CLM **configure** script* in Chapter 1.

Option: -comp_intf <name>>
Description: Component interface to use (ESMF or MCT) (default MCT)

Option: -nofire
Description: Turn off wildfires for bgc setting of CN (default includes fire for CN)

Option: -pio <name>
Description: Switch enables building with Parallel I/O library. [on | off] (default is on)

Option: -snicar_frc <name>
Description: Turn on SNICAR radiative forcing calculation. [on | off] (default is off)
More information on options to clm **configure** are given in the Section called *More information on the CLM **configure** script* in Chapter 1.

What are The New Namelist Options?

build-namelist now checks the validity of your namelist you generate by looking at data in the namelist_definition.xml file. In order to add new namelist items you need to change the code and also edit this file (e.g. a namelist option required for your research project that is not currently an option in CLM). To view information on the namelist view the file: `models/lnd/clm/bld/namelist_files/namelist_definition.xml` in a browser and you'll see the names, type, description and valid_values for all namelist variables.

Changes to **build-namelist**:

Allow simulation year entered to include ranges of years (i.e. 1850-2000)
Remove cam_hist_case option.
Make sure options ONLY used for stand-alone testing have a "drv_" or "datm_" prefix in them and list these options all tog

New option to **build-namelist**:

```
   -clm_usr_name "name" Dataset resolution/descriptor for personal datasets.
                        Default: not used
                        Example: 1x1pt_boulderCO_c090722 to describe location,
                                 number of pts, and date files created
```

New list options to **build-namelist**

```
      build-namelist -res list          # List valid resolutions
      build-namelist -mask list         # List valid land-masks
      build-namelist -sim_year list     # List valid simulation years and simulation year ranges
      build-namelist -clm_demand list   # List namelist variables including those you could
                                    # demand to be set.
      build-namelist -use_case list     # List valid use-cases
```

New use-cases for **build-namelist**:

```
    1850_control = Conditions to simulate 1850 land-use
    2000_control = Conditions to simulate 2000 land-use
 20thC_transient = Simulate transient land-use, aerosol and Nitrogen deposition
                   from 1850 to 2005
```

New namelist items:

```
    urban_hac = OFF, ON or ON_WASTEHEAT   (default OFF)     Flag for urban Heating
                                                        and Air-Conditioning
                OFF          = Building internal temperature is un-regulated.
                ON           = Building internal temperature is bounded to reasonable range.
                ON_WASTEHEAT = Building internal temperature is bounded and resultant waste
                               heat is given off.
    urban_traffic = .true. or .false.     Flag to include additional multiplicative
                                          factor of urban traffic to sensible heat flux.
                                          (default .false.)
    fsnowoptics  = filename    file for snow/aerosol optical properties (required)
    fsnowaging   = filename    file for snow aging parameters (required)
    faerdep      = filename    file of aerosol deposition (required)
```

More information on the **build-namelist** options are given in the Section called *Definition of Namelist items and their default values* in Chapter 1.

More information on the **build-namelist** options are given in in the Section called *Definition of Namelist items and their default values* in Chapter 1.

What are The New History Fields?

New history variables: (note watt vs. W in units, 26 vs. 76)

Name: BCDEP
Long-name: total BC deposition (dry+wet) from atmosphere
Units: kg/m^2/s

Name: BIOGENCO
Long-name: biogenic CO flux
Units: uGC/M2/H

Name: C13_PRODUCT_CLOSS
Long-name: C13 total carbon loss from wood product pools
Units: gC13/m^2/s

Name: DSTDEP
Long-name: total dust deposition (dry+wet) from atmosphere
Units: kg/m^2/s

Name: EFLX_DYNBAL
Long-name: dynamic land cover change conversion energy flux
Units: W/m^2

Name: FGR12
Long-name: heat flux between soil layers 1 and 2
Units: watt/m^2

Name: FSAT
Long-name: fractional area with water table at surface
Units: unitless

Name: FSH_NODYNLNDUSE
Long-name: sensible heat flux not including correction for land use change
Units: watt/m^2

Name: GC_HEAT1
Long-name: initial gridcell total heat content
Units: J/m^2

Name: GC_HEAT2
Long-name: post land cover change total heat content
Units: J/m^2
Active/Inactive: inactive

Name: GC_ICE1
Long-name: initial gridcell total ice content

Units: mm/s

Name: GC_ICE2
Long-name: post land cover change total ice content
Units: mm/s
Active/Inactive: inactive

Name: GC_LIQ1
Long-name: initial gridcell total liq content
Units: mm

Name: GC_LIQ2
Long-name: initial gridcell total liq content
Units: mm
Active/Inactive: inactive

Name: H2OSNO_TOP
Long-name: mass of snow in top snow layer
Units: kg

Name: HEAT_FROM_AC
Long-name: sensible heat flux put into canyon due to heat removed from air conditioning
Units: watt/m^2

Name: HK
Long-name: hydraulic conductivity
Units: mm/s
Active/Inactive: inactive

Name: ISOPRENE
Long-name: isoprene flux
Units: uGC/M2/H

Name: LAND_USE_FLUX
Long-name: total C emitted from land cover conversion and wood product pools
Units: gC/m^2/s

Name: LAND_UPTAKE
Long-name: NEE minus LAND_USE_FLUX, negative for update
Units: gC/m^2/s

Name: LWup
Long-name: upwelling longwave radiation
Units: watt/m^2
Active/Inactive: inactive

Name: MONOTERP
Long-name: monoterpene flux
Units: uGC/M2/H

Name: NBP
Long-name: net biome production, includes fire, landuse, and harvest flux, positive for sink
Units: gC/m^2/s

Name: OCDEP
Long-name: total OC deposition (dry+wet) from atmosphere

Units: kg/m^2/s

Name: OVOC
Long-name: other VOC flux
Units: uGC/M2/H

Name: ORVOC
Long-name: other reactive VOC flux
Units: uGC/M2/H

Name: PBOT
Long-name: atmospheric pressure
Units: Pa

Name: PCO2
Long-name: atmospheric partial pressure of CO_2
Units: Pa

Name: PRODUCT_CLOSS
Long-name: total carbon loss from wood product pools
Units: gC/m^2/s

Name: PRODUCT_NLOSS
Long-name: total N loss from wood product pools
Units: gN/m^2/s

Name: Qair
Long-name: atmospheric specific humidity
Units: kg/kg
Active/Inactive: inactive

Name: Qanth
Long-name: anthropogenic heat flux
Units: watt/m^2
Active/Inactive: inactive

Name: Qtau
Long-name: momentum flux
Units: kg/m/s^2

Name: QFLX_LIQ_DYNBAL
Long-name: liq dynamic land cover change conversion runoff flux
Units: mm/s

Name: QFLX_ICE_DYNBAL
Long-name: ice dynamic land cover change conversion runoff flux
Units: mm/s

Name: QRUNOFF_NODYNLNDUSE
Long-name: total liquid runoff not including correction for land use change (does not include QSNWCPICE)
Units: mm/s

Name: QSNWCPICE
Long-name: excess snowfall due to snow capping
Units: mm/s

Name: QSNWCPICE_NODYNLNDUSE
Long-name: excess snowfall due to snow capping not including correction for land use change
Units: mm/s

Name: QSNWCPLIQ
Long-name: excess rainfall due to snow capping
Units: mm/s
Active/Inactive: inactive

Name: SMP
Long-name: soil matric potential
Units: mm
Active/Inactive: inactive

Name: SNOAERFRC2L
Long-name: surface forcing of all aerosols in snow, averaged only when snow is present (land)
Units: watt/m^2

Name: SNOAERFRCL
Long-name: surface forcing of all aerosols in snow (land)
Units: watt/m^2

Name: SNOBCFRCL
Long-name: surface forcing of BC in snow (land)
Units: watt/m^2

Name: SNOBCMCL
Long-name: mass of BC in snow column
Units: kg/m2

Name: SNOBCMSL
Long-name: mass of BC in top snow layer
Units: kg/m2

Name: SNOdTdzL
Long-name: top snow layer temperature gradient (land)
Units: K/m

Name: SNODSTFRC2L
Long-name: surface forcing of dust in snow, averaged only when snow is present (land)
Units: watt/m^2

Name: SNODSTFRCL
Long-name: surface forcing of dust in snow (land)
Units: watt/m^2

Name: SNODSTMCL
Long-name: mass of dust in snow column
Units: kg/m2

Name: SNODSTMSL
Long-name: mass of dust in top snow layer
Units: kg/m2

Name: SNOFSRND
Long-name: direct nir reflected solar radiation from snow

Units: watt/m^2
Active/Inactive: inactive

Name: SNOFSRNI
Long-name: diffuse nir reflected solar radiation from snow
Units: watt/m^2
Active/Inactive: inactive

Name: SNOFSRVD
Long-name: direct vis reflected solar radiation from snow
Units: watt/m^2
Active/Inactive: inactive

Name: SNOFSRVI
Long-name: diffuse vis reflected solar radiation from snow
Units: watt/m^2
Active/Inactive: inactive

Name: SNOFSDSND
Long-name: direct nir incident solar radiation on snow
Units: watt/m^2
Active/Inactive: inactive

Name: SNOFSDSNI
Long-name: diffuse nir incident solar radiation on snow
Units: watt/m^2
Active/Inactive: inactive

Name: SNOFSDSVD
Long-name: direct vis incident solar radiation on snow
Units: watt/m^2
Active/Inactive: inactive

Name: SNOFSDSVI
Long-name: diffuse vis incident solar radiation on snow
Units: watt/m^2
Active/Inactive: inactive

Name: SNOLIQFL
Long-name: top snow layer liquid water fraction (land)
Units: fraction
Active/Inactive: inactive

Name: SNOOCMCL
Long-name: mass of OC in snow column
Units: kg/m2

Name: SNOOCMSL
Long-name: mass of OC in top snow layer
Units: Kg/m2

Name: SNOOCFRC2L
Long-name: surface forcing of OC in snow, averaged only when snow is present (land)
Units: watt/m^2

Name: SNOOCFRCL
Long-name: surface forcing of OC in snow (land)
Units: watt/m^2

Name: SNORDSL
Long-name: top snow layer effective grain radius
Units: m^-6
Active/Inactive: inactive

Name: SNOTTOPL
Long-name: snow temperature (top layer)
Units: K/m
Active/Inactive: inactive

Name: SWup
Long-name: upwelling shortwave radiation
Units: watt/m^2
Active/Inactive: inactive

Name: TSOI_10CM
Long-name: soil temperature in top 10cm of soil
Units: K

Name: URBAN_AC
Long-name: urban air conditioning flux
Units: watt/m^2

Name: URBAN_HEAT
Long-name: urban heating flux
Units: watt/m^2

Name: VOCFLXT
Long-name: total VOC flux into atmosphere
Units: uGC/M2/H

Name: Wind
Long-name: atmospheric wind velocity magnitude
Units: m/s
Active/Inactive: inactive

Name: WOOD_HARVESTC
Long-name: wood harvest (to product pools)
Units: gC/m^2/s

Name: WOOD_HARVEST
Long-name: wood harvest (to product pools)
Units: gN/m^2/s

History field name changes:

Old: ANNSUM_PLANT_NDEMAND
New: = ANNSUM_POTENTIAL_GPP

Old: ANNSUM_RETRANSN
New: = ANNMAX_RETRANSN

Old: C13_DWT_PROD10C_LOSS
New: = C13_PROD10C_LOSS

Old: C13_DWT_PROD100C_LOSS
New: = C13_PROD100C_LOSS

Old: C13_DWT_PROD10N_LOSS
New: = C13_PROD10N_LOSS

Old: C13_DWT_PROD100C_LOSS
New: = C13_PROD100C_LOSS

Old: DWT_PROD100N_LOSS
New: = PROD10N_LOSS

Old: DWT_PROD100N_LOSS
New: = PROD100N_LOSS

Old: DWT_PROD100C_LOSS
New: = PROD10C_LOSS

Old: DWT_PROD100C_LOSS
New: = PROD100C_LOSS

Old: HCSOISNO
New: = HC

Old: TEMPSUM_PLANT_NDEMAND
New: = TEMPSUM_POTENTIAL_GPP

Old: TEMPSUM_RETRANSN
New: = TEMPMAX_RETRANSN

History field names deleted include: SNOWAGE, TSNOW, FMICR, FCO2, DMI, QFLX_SNOWCAP

Add new urban oriented _U, and _R (Urban and Rural) for the following history variables: EFLX_LH_TOT, FGR, FIRA, FSH, FSM, Q2M, QRUNOFF, RH2M, SoilAlpha, TG, TREFMNAV, TREFMXAV, and TSA (missing _R for SoilAlpha)

Note: We are missing the Rural soil-alpha variable: SoilAlpha_R

Quickstart to using CLM4

Before working with CLM4 read the QuickStart Guide in the CCSM4.0 Scripts User's Guide (http://www.ccsm.ucar.edu/models/ccsm4.0/ccsm_doc/book1.html). Once you are familiar with how to setup cases for any type of simulation with CCSM you will want to direct your attention to the specifics of using CLM.

For some of the details of setting up cases for CLM4 read the README and text files available from the "models/lnd/clm/doc" directory (see the "CLM Web pages" section for a link to the list of these files). Here are the important ones that you should be familiar with.

1. README (../README) file describing the directory structure.
2. Quickstart.userdatasets (../Quickstart.userdatasets) file describing how to use your own datasets in the model (also see the Section called *Creating your own single-point/regional surface datasets* in Chapter 5).
3. KnownBugs (../KnownBugs) file describing known problems in CLM4.

The *IMPORTANT_NOTES* file is given in the next chapter on what is functional/validated in CLM4?

The *ChangeLog/ChangeSum* files are largely explained in the previous chapter on "What is new with CLM4?"

Note other directories have README files that explain different components and tools used when running CLM and are useful in understanding how those parts of the model work and should be consulted when using tools in those directories. For more details on configuring and customizing a case with CLM see Chapter 1.

The *Quickstart.GUIDE* (which can be found in `models/lnd/clm/doc)` is repeated here.

```
          Quick-Start to Using cpl7 Scripts for clm4
          ===========================================

Assumptions: You want to use bluefire with clm4
             to do a clm simulation with data atmosphere and the
             latest atm forcing files and settings. You also want to cycle
             the atm data between 1948 to 2004 and you want to run at
             1.9x2.5 degree resolution.

Process:

   # Create the case

   cd scripts

   ./create_newcase -case <testcase> -mach bluevista -res f19_g15 -compset I4804
   (./create_newcase -help -- to get help on the script)

   # Configure the case

   cd <testcase>
   $EDITOR env_run.xml env_conf.xml   # If you need to make changes
                                      # (or use the xmlchange script)
   ./configure -case
   (./configure -help -- to get help on the script)

   # Make any changes to the namelist

   $EDITOR Buildconf/clm.buildnml_prestage.csh

   # Compile the code
   ./<testcase>.build

   # Submit the run

   bsub < <testcase>.run

Information on Compsets:

     "I" compsets are the ones with clm and datm7 without ice and ocean.
     The latest "I" compsets use the new CLM_QIAN data with solar following
     the cosine of solar zenith angle, precipitation constant, and other
     variables linear interpolated in time (and with appropriate time-stamps on
     the date). Some of the I compsets are:

     Name           (short-n): Description
     ------------------------------------------------------------------------
     I_2000         (I):       CLM to simulate year=2000
     I_1850         (I1850):   CLM to simulate year=1850
     I_1948_2004    (I4804):   CLM running with atm data over 1948-2004
     I_1850-2000    (I8520):   CLM with transient PFT over 1850-2000
     I_2000_CN      (ICN):     CLM with CN on to simulate year=2000
     I_1850_CN      (I1850CN): CLM with CN on to simulate year=1850
     I_1948-2004_CN (I4804CN): CLM with CN on with atm data over 1948-2004
     I_1850-2000_CN (I8520CN): CLM with CN on with transient PFT over 1850-2000

Automatically resubmitting jobs:

   After doing a short simulation that you believe is correct

   xmlchange -file env_run.xml -id CONTINUE_RUN -val TRUE

   # Change RESUBMIT to number greater than 0, and CONTINUE_RUN to TRUE...

   bsub < <testcase>.run
```

What is scientifically validated and functional in CLM4?

In this section we go over what has been extensively tested and scientifically validated with CLM4.0, and maybe more importantly what has NOT been tested and may NOT be scientifically validated. You can use all features of CLM, but need to realize that some things haven't been tested extensively or validated scientifically. When you use these features you may run into trouble doing so, and will need to do your own work to make sure the science is reasonable.

Standard Configuration and Namelist Options that are Validated

The standard version of the model is CLMCN at 1-degree horizontal resolution (0.9x1.25). This version has been scientifically validated with long simulations for: fully coupled simulations ("B" cases), coupled to atmosphere model CAM ("F" cases), and stand-alone CLM cases ("I" cases). We've also done both long simulations for 1850 conditions, and transient 20th century simulations from 1850 to 2005 (with transient land-use, Nitrogen and Aerosol deposition). To a lesser extent there have also been simulations done at 2-degree horizontal resolution (1.9x2.5), and with CLMSP for these resolutions. As such we have provided appropriate 1-degree and 2-degree initial condition datasets for these configurations. Other resolutions, configurations, and namelist options are less well tested or scientifically validated. The further you get away from the standard configurations and resolutions, the more likely you are to run into trouble, and/or need to scientifically validate your work.

In the sections below we go through configuration and/or namelist options or modes that the user should be especially wary of using. You are of course free to use these options, and you may find that they work functionally. Although in some cases you will find issues even with functionality of using them. If so you will need to test, debug and find solutions for these issues on your own. But in every case you will need to go through more extensive work to validate these options from a scientific standpoint.

Configure Modes NOT scientifically validated, documented, supported or, in some cases, even advised to be used:

1. `C13`

   `(-c13)`

   The C13 mode for bgc=cn is NOT scientifically validated or documented and is NOT recommended for use.

2. `CASA`

```
(-bgc casa)
```

The bgc=casa mode is NOT scientifically validated or documented and is NOT recommended for use.

3. `SUPLN`

   ```
   (-bgc cn -supln on)
   ```

   The supplemental Nitrogen mode of the CN Biogeochemistry model supplies unlimited nitrogen and therefore vegetation is over-productive in this mode.

4. `BUILDPIO`

   ```
   (-pio)
   ```

   This mode is NOT tested and is not functional and hence should NOT be used. PIO WILL be provided in future versions however.

5. `SNICAR_FRC`

   ```
   (-snicar_frc)
   ```

   This mode is tested and functional, but is NOT constantly scientifically validated, and should be considered experimental.

6. `PERGRO`

   ```
   (-pergro)
   ```

   This mode is tested to be functional, but NOT scientifically validated. It's purpose is to help validate that a port to a new machine is reasonable, or that changes that are only roundoff will NOT affect the climate. Using PERGRO as a validation tool remains experimental. See the Section called *Doing perturbation error growth tests* in Chapter 4 for more information on how to do this.

Namelist options that should NOT be exercised:

Build-Namelist options that should NOT be exercised:

1. *-lnd_res:* Fine-mesh mode, functional, but experimental
2. *-rcp:* Representative Concentration Pathway (RCP) for future scenarios, functional for limited resolutions, but experimental
3. *-datm_*:* All options that start with "datm_" they are only used for CLM stand-alone testing.
4. *-drv_*:* All options that start with "drv_" they are only used for CLM stand-alone testing.

Namelist items that should NOT be exercised:

1. *casa namelist options:* lnpp, lalloc, q10, spunup, and fcpool CASA has NOT been scientifically validated in CLM4.
2. *create_crop_landunit:* Functional, but experimental
3. *fine-mesh namelist options:* flndtopo, and fatmtopo. These options are functional but experimental. See the -lnd_res option above.
4. *pio namelist options:* hist_pioflag, ncd_lowmem2d, ncd_pio_def, ncd_pio_UseRearranger, ncd_pio_UseBoxRearr, ncd_pio_SerialCDF, ncd_pio_DebugLevel, and ncd_pio_num_iotasks These options are NOT currently tested or functional. See the pio **configure** mode above.
5. *urban_traffic:* Not currently functional

What are the UNIX utilities required to use CLM?

Running the CLM requires a suite of UNIX utilities and programs and you should make sure you have all of these available before trying to go forward with using it. If you are missing one of these you should contact the systems administrator for the machine you wish to run on and make sure they are installed.

FORTRAN-90 compiler
"C" compiler
GNU make
UNIX csh and tcsh shells
UNIX sh shell
UNIX bash shell
UNIX awk
UNIX sed
NetCDF library
MPI Library
"C" pre-processor
Perl
Autoconf
m4 macro processor
Parallel NetCDF (optional)
NCL (for some of the offline tools for creating/modifying CLM input datasets see Chapter 2 for more information on NCL

Other resources to get help from

In addition to this users-guide there are several other resources that are available to help you use CLM. The first one is the CCSM User's-Guide, which documents the entire process of creating cases with CCSM. The next is the CCSM bulletin board which is a web-site for exchanging information between users of CCSM. There are also CLM web-pages specific for CLM, and finally there is an email address to report bugs that you find in CCSM4.0.

The CCSM User's-Guide

CLM4 is always run from within the standard CCSM4.0 build and run scripts. Therefore, the user of CLM4 should familiarize themselves with the CCSM4.0 scripts and understand how to work with them. User's-Guide documentation on the CCSM4.0 scripts are available from the following web-page. The purpose of this CLM4 User's Guide is to give the CLM4 user more complete details on how to work with CLM and the set of tools that support CLM, as well as to give examples that are unique to the use of CLM. However, the CCSM4.0 Scripts User's-Guide remains the primary source to get detailed information on how to build and run the CCSM system.

CCSM4.0 Scripts User's-Guide (http://www.ccsm.ucar.edu/models/ccsm4.0/ccsm_doc/book1.html)

The CCSM Bulletin Board

There is a rich and diverse set of people that use the CCSM, and often it is useful to be in contact with others to get help in solving problems or trying something new. To facilitate this we have an online Bulletin Board for questions on the CCSM. There are also different sections in the Bulletin Board for the different component models or for different topics.

CCSM Online Bulletin Board (http://bb.cgd.ucar.edu/)

The CLM web pages

The main CLM web page contains information on the CLM, it's history, developers, as well as downloads for previous model versions. There are also documentation text files in the models/lnd/clm/doc directory that give some quick information on using CLM.

CLM web page (http://www.cgd.ucar.edu/tss/clm/)
CLM Documentation Text Files (../)

Reporting bugs in CLM4

If you have any problems, additional questions, bug reports, or any other feedback, please send an email to <`ccsm4-help@cgd.ucar.edu`>. If you find bad, wrong, or misleading information in this users guide send an email to <`erik@ucar.edu`>.

Chapter 1. How to customize the configuration for a case with CLM

The CCSM User's Guide (http://www.ccsm.ucar.edu/models/ccsm4.0/ccsm_doc/book1.html) gives you the details on how to setup, **configure**, build, and run a case. That is the document to give you the details on using the CCSM scripts. The purpose of this document is to give you the details when using CCSM with CLM on how to customize and use advanced features in CLM. You should be familiar with the CCSM User's Guide and how to setup cases with CCSM4.0 before referring to this document.

In this chapter we deal with three different ways of customizing a case: Choosing a compset, Customizing Configuration options, and customizing the CLM Namelist. There are many different compsets that use CLM and many are setup to enable special features of CLM from the start. So the first thing you want to be familiar with are the different options in the compsets. The next section shows the different options for customizing the configuration options for CLM. Here we introduce the CLM **configure** and **build-namelist** scripts and how using the `env_conf.xml` options you can customize the configuration and the initial namelist. The final section tells you about the CLM namelist and how you could customize the namelist once you have run "**configure** -case" and have an initial namelist in `BuildConf/clm.buildnml.csh` that you can customize by hand. You can also use `env_conf.xml` options to change your namelist as well.

Choosing a compset using CLM

When setting up a new case one of the first choices to make is which "component set" (or compset) to use. The compset refers to which component models are used as well as specific settings for them. We label the different types of compsets with a different letter of the alphabet from "A" (for all data model) to "X" (for all dead model). The compsets of interest when working with CLM are the "I" compsets (which contain CLM with a data atmosphere model and a stub ocean, and stub sea-ice models), "E" and "F" compsets (which contain CLM with the active atmosphere model (CAM), prescribed sea-ice model, and a data ocean model), and "B" compsets which have all active components. Below we go into details on the "I" compsets which emphasize CLM as the only active model, and just mention the two other categories.

When working with CLM you usually want to start with a relevant "I" compset before moving to the more complex cases that involve other active model components. The "I" compsets can exercise CLM in a way that is similar to the coupled modes, but with much lower computational cost and faster turnaround times.

Compsets coupled to data atmosphere and stub ocean/sea-ice ("I" compsets)

1. `I_2000`(`I`) Active land model with QIAN atm input data for 2003 and Satellite phenology (SP), CO2 level and Aerosol deposition for 2000
2. `I_1850`(`I1850`) Active land model with QIAN atm input data for 1948 to 1972 and Satellite phenology (SP), CO2 level and Aerosol deposition for 1850

3. `I_1948-2004`(`I4804`) Active land model with QIAN atm input data for 1948 to 2004 and Satellite phenology (SP), CO2 level and Aerosol deposition for 2000
4. `I_1850-2000`(`I8520`) Active land model with QIAN atm input data for 1948 to 2004 and transient Satellite phenology (SP), and Aerosol deposition from 1850 to 2000 and 2000 CO2 level
5. `I_2000_CN`(`ICN`) Active land model with QIAN atm input data for 2003 and CN (Carbon Nitrogen) biogeochemistry, CO2 level and Aerosol deposition for 2000
6. `I_1850_CN`(`I1850CN`) Active land model with QIAN atm input data for 1948 to 1972 and CN (Carbon Nitrogen) biogeochemistry, CO2 level and Aerosol deposition for 1850
7. `I_1948-2004_CN`(`I4804CN`) Active land model with QIAN atm input data for 1948 to 2004 and CN (Carbon Nitrogen) biogeochemistry, CO2 level and Aerosol deposition for 2000
8. `I_1850-2000_CN`(`I8520CN`) Active land model with QIAN atm input data for 1948 to 1972 and transient CN (Carbon Nitrogen) biogeochemistry, and Aerosol deposition from 1850 to 2000 and 2000 CO2 level

Compsets coupled to active atmosphere with data ocean

CAM compsets are compsets that start with "E" or "F" in the name. They are described more fully in the scripts documentation or the CAM documentation. "E" compsets have a slab ocean model while "F" compsets have a data ocean model.

Fully coupled compsets with fully active ocean, sea-ice, and atmosphere

Fully coupled compsets are compsets that start with "B" in the name. They are described more fully in the scripts documentation.

Conclusion to choosing a compset

We've introduced the basic type of compsets that use CLM and given some further details for the "standalone CLM" (or "I" compsets). The config_compsets.xml (../../../../../scripts/ccsm_utils/Case.template/config_compsets.xml) lists all of the compsets and gives a full description of each of them. In the next section we look into customizing the **configure** time options for compsets using CLM.

Customizing the CLM configuration

The "Creating a Case" section of the CCSM4.0 Scripts User's-Guide (http://www.ccsm.ucar.edu/models/ccsm4.0/ccsm_doc/book1.html) gives instructions on creating a case. What is of interest here is how to customize your use of CLM for the case that you created. In this section we discuss how to customize your case before the first step -- the "**configure** -case" step is done.

In the next section we will discuss how to customize your CLM namelist after "**configure** -case" has already been done.

For CLM when "**configure** -case" is called there are two steps that take place:

1. The CLM "**configure**" script is called to setup the build-time configuration for CLM (more information on **configure** is given in the Section called *More information on the CLM **configure** script*).
2. The CLM "**build-namelist**" script is called to generate the initial run-time namelist for CLM (more information on **build-namelist** is given below in the Section called *Definition of Namelist items and their default values*.

When customizing your case at the **configure** step you are able to modify the process by effecting either one or both of these steps. The CLM "**configure**" and "**build-namelist**" scripts are both available in the "models/lnd/clm/bld" directory in the distribution. Both of these scripts have a "-help" option that is useful to examine to see what types of options you can give either of them.

There are five different types of customization for the configuration that we will discuss: CCSM4.0 CLM configuration items, Configure time User Namelist, other noteworthy CCSM configuration items, the CLM **configure** script options, and the CLM **build-namelist** script options.

Information on all of the script, configuration, build and run items is found under `scripts/ccsm_utils/Case.template` in the `config_definition.xml` (../../../../../scripts/ccsm_utils/Case.template/config_definition.xml) file.

CLM Script configuration items

Below we list each of the CCSM configuration items that are specific to CLM. All of these are available in your: `env_conf.xml` file.

CLM_CONFIG_OPTS
CLM_NAMELIST_OPTS
CLM_FORCE_COLDSTART
CLM_NML_USE_CASE
CLM_1PT_NAME
CLM_USRDAT_NAME
CLM_CO2_TYPE

The first item CLM_CONFIG_OPTS has to do with customizing the CLM configuration options for your case, the rest all have to do with generating the initial namelist.

CLM_CONFIG_OPTS

The option CLM_CONFIG_OPTS is all about passing command line arguments to the CLM **configure** script. It is important to note that some compsets, may already put a value into the CLM_CONFIG_OPTS variable. You can still add more options to your CLM_CONFIG_OPTS but make sure you add to what is already there rather than replacing it. Hence, it may be best to edit the `env_conf.xml` file to do this rather than use the xmlchange script. In the the Section called *More information on the CLM **configure** script* below we will go into more details on options that can be customized in the CLM "**configure**" script. It's also important to note that the CLM template may already invoke certain CLM **configure** options and as such those command line options are NOT going to be available to change at this step (nor would you want to change them). The options to

configure are given with the "-help" option which is given in the Section called *More information on the CLM **configure** script*.

CLM_NML_USE_CASE

CLM_NML_USE_CASE is used to set a particular set of conditions that set multiple namelist items, all centering around a particular usage of the model. To list the valid options do the following:

```
> cd models/lnd/clm/doc
> ../bld/build-namelist -use_case list
```

The output of the above command is:

```
build-namelist - use cases: 1850-2100_rcp2.6_transient 1850-2100_rcp6_transient 1850_control 2000-2100_rcp8.5_transient 2000_control 20
Use cases are:...

1850-2100_rcp2.6_transient = Simulate transient land-use, and aerosol deposition changes with historical data from 1850 to 2005 and the

1850-2100_rcp6_transient = Simulate transient land-use, and aerosol deposition changes with historical data from 1850 to 2005 and then

   1850_control = Conditions to simulate 1850 land-use
2000-2100_rcp8.5_transient = Simulate transient land-use, and aerosol deposition changes with historical data from 2000 to 2005 and the

   2000_control = Conditions to simulate 2000 land-use
20thC_transient = Simulate transient land-use, and aerosol deposition changes from 1850 to 2005
         pergro = Perturbation error growth test with initial conditions perturbed by roundoff level
        pergro0 = Perturbation error growth test with unperturbed initial conditions
```

CLM_NAMELIST_OPTS

The option CLM_NAMELIST_OPTS is for passing namelist items into the "clm_inparm" namelist. Any items that are set in CLM_NAMELIST_OPTS will be set in your namelist after "**configure** -case" is done.

> **Important:** For character namelist items you need to use "&apos;" as quotes for strings so that the scripts don't get confused with other quotes they use.

Example, you want to set `hist_dov2xy` to `.false.` so that you get vector output to your history files. To do so edit `env_conf.xml` and add a setting for `hist_dov2xy`. So do the following:

```
> xmlchange -file env_conf.xml -id CLM_NAMELIST_OPTS \
-val hist_dov2xy=.false.
```

Example, you want to set `hist_fincl1` to add the variable `'HK'` to your history files. To do so edit `env_conf.xml` and add a setting for `hist_fincl1`. So do the following:

```
> xmlchange -file env_conf.xml -id CLM_NAMELIST_OPTS \
 -val "hist_fincl1=&apos;HK&apos;"
```

CLM_CO2_TYPE

CLM_CO2_TYPE sets the type of input CO_2 for either "constant", "diagnostic" or prognostic". If "constant" the value from CCSM_CO2_PPMV will be used. If "diagnostic" or "prognostic" the values MUST be sent from the atmosphere model. For more information on how to send CO_2 from the data atmosphere model see the Section called *Running stand-alone CLM with transient historical CO_2 concentration* in Chapter 4.

CLM_FORCE_COLDSTART

CLM_FORCE_COLDSTART when set to `on`, *requires* that your simulation do a cold start from arbitrary initial conditions. If this is NOT set, it will use an initial condition file if it can find an appropriate one, and otherwise do a cold start. CLM_FORCE_COLDSTART is a good way to ensure that you are doing a cold start if that is what you want to do.

CLM_1PT_NAME

CLM_1PT_NAME is used *ONLY* for a `pt1_pt1` resolution simulation to set the name of the single-point files to use. To see a list of the valid resolutions do this:

```
> cd models/lnd/clm/doc
> ../bld/build-namelist -res list
```

The output of the above command is:

```
build-namelist - valid values for res: default 128x256 64x128 48x96 32x64 8x16 94x192 0.23x0.31 0.47x0.63 0.9x1.25 1.9x2.5 2.65x3.33 4
                 default = 1.9x2.5
                 (NOTE: resolution and mask and other settings may influence what the default is)
```

the valid resolutions that can be used with CLM_1PT_NAME are the ones that have city or nation names such as: 5x5_amazon, 1x1_vancouverCAN 1x1_mexicocityMEX, or 1x1_brazil. The "1x1_" prefix means the file is for a single-point, while "5x5_" prefix means it's for a region of five points in latitude by five points in longitude. Both regional and single point datasets can be used for CLM_1PT_NAME.

CLM_USRDAT_NAME

CLM_USRDAT_NAME provides a way to enter your own datasets into the initial namelist setup at "**configure** -case". The files you create must be named with specific naming conventions outlined in: the Section called *Creating your own single-point/regional surface datasets* in Chapter 5. To see what the expected names of the files are, use the **queryDefaultNamelist.pl** to see what the names will need to be. For example if your CLM_USRDAT_NAME will be "1x1_boulderCO", with a "navy" land-mask, constant simulation year range, for 1850, the following will list what your filenames should be:

```
> cd models/lnd/clm/bld
> queryDefaultNamelist.pl -usrname "1x1_boulderCO" -options \
mask=navy,sim_year=1850,sim_year_range="constant"  -csmdata $CSMDATA
```

Configure time User Namelist

CLM_NAMELIST_OPTS as described above allows you to set any extra namelist items you would like to appear in your namelist after first **configure**d. However, it only allows you a single line to enter namelist items, and strings must be quoted with &apos; which is a bit awkward. If you have a long list of namelist items you want to set (such as a long list of history fields) a convenient way to do it is to create a `user_nl_clm` that contains just the list of namelist variables you want to add to your initial namelist. The `user_nl_clm` will only be used when **configure** is run, so if you change it after **configure** -- it won't change anything. The file needs to be in valid FORTRAN namelist format, and the **configure** step will abort if there are syntax errors. It merely needs to be named correctly `user_nl_clm` and placed in your case directory (where your other `env_*.xml` files are). The namelist name actually doesn't have to be valid, but all the variable names must be. Here's an example `user_nl_clm` namelist that sets a bunch of history file related items, to create output history files monthly, daily, every six and 1 hours.

Example 1-1. Example `user_nl_clm` namelist file

```
&clmexp
 hist_fincl2    = 'TG','TBOT','FIRE','FIRA','FLDS','FSDS',
                  'FSR','FSA','FGEV','FSH','FGR','TSOI',
                  'ERRSOI','BUILDHEAT','SABV','SABG',
                  'FSDSVD','FSDSND','FSDSVI','FSDSNI',
                  'FSRVD','FSRND','FSRVI','FSRNI',
                  'TSA','FCTR','FCEV','QBOT','RH2M','H2OSOI',
                  'H2OSNO','SOILLIQ','SOILICE',
                  'TSA_U', 'TSA_R',
                  'TREFMNAV_U', 'TREFMNAV_R',
                  'TREFMXAV_U', 'TREFMXAV_R',
                  'TG_U', 'TG_R',
                  'RH2M_U', 'RH2M_R',
                  'QRUNOFF_U', 'QRUNOFF_R',
                  'SoilAlpha_U',
                  'Qanth', 'SWup', 'LWup', 'URBAN_AC', 'URBAN_HEAT'
  hist_fincl3 = 'TG:I', 'FSA:I', 'SWup:I', 'URBAN_AC:I', 'URBAN_HEAT:I',
                'TG_U:I', 'TG_R:I',
  hist_fincl4 = 'TG', 'FSA', 'SWup', 'URBAN_AC', 'URBAN_HEAT'
  hist_mfilt  = 1, 30,  28, 24
  hist_nhtfrq = 0, -24, -6, -1
/
```

Note: In the above example we use an invalid namelist name &clmexp -- but it works anyway because the CLM **build-namelist** knows the namelist that specific variable names belong to, and it puts them there.

Obviously, all of this would be difficult to put in the CLM_NAMELIST_OPTS variable, especially having to put &apos; around all the character strings. For more information on the namelist variables being set here and what they mean, see the section on CLM namelists below, as well as the namelist definition that gives details on each variable.

Other noteworthy configuration items

For running "I" cases there are several other noteworthy configuration items that you may want to work with. Most of these involve settings for the DATM, but one CCSM_CO2_PPMV applies to all models. If you are running an B, E, or F case that doesn't use the DATM obviously the DATM_* settings will not be used. All of the settings below are in your `env_conf.xml` file

CCSM_CO2_PPMV
DATM_MODE
DATM_CLMNCEP_YR_ALIGN
DATM_CLMNCEP_YR_START
DATM_CLMNCEP_YR_END

CCSM_CO2_PPMV

CCSM_CO2_PPMV sets the mixing ratio of CO_2 in parts per million by volume for ALL CCSM components to use. Note that most compsets already set this value to something reasonable. Also note that some compsets may tell the atmosphere model to override this value with either historic or ramped values. If the CCSM_BGC variable is set to something other than "none" the atmosphere model will determine CO_2, and CLM will listen and use what the atmosphere sends it. On the CLM side the namelist item `co2_type` tells CLM to use the value sent from the atmosphere rather than a value set on it's own namelist.

DATM_MODE

DATM_MODE sets the mode that the DATM model should run in this determines how data is handled as well as what the source of the data will be. Many of the modes are setup specifically to be used for ocean and/or sea-ice modeling. The modes that are designed for use by CLM are:
`CLM_QIAN`
CLM_1PT_NAME

`CLM_QIAN` is for the standard mode of using global atmospheric data that was developed by Qian et. al. for CLM using NCEP data from 1948 to 2004. `CLM1PT` is for the special cases where we have single-point tower data for particular sites. Right now we only have data for three urban locations: MexicoCity Mexico, Vancouver Canada, and the urban-c alpha site. This is non-standard functionality and we recommend that users stick with the `CLM_QIAN` mode of operation.

Warning

Currently the urban locations are NOT functional, because the surface datasets need to be updated.

DATM_CLMNCEP_YR_START

DATM_CLMNCEP_YR_START sets the beginning year to cycle the atmospheric data over for the `CLM_QIAN` mode.

DATM_CLMNCEP_YR_END

DATM_CLMNCEP_YR_END sets the ending year to cycle the atmospheric data over for the `CLM_QIAN` mode.

DATM_CLMNCEP_YR_ALIGN

DATM_CLMNCEP_YR_START and DATM_CLMNCEP_YR_END determine the range of years to cycle the atmospheric data over, and DATM_CLMNCEP_YR_ALIGN determines which year in that range of years the simulation will start with.

Customizing via the template files

The final thing that the user may wish to do before **configure** is run is to edit the template files which determine the configuration and initial namelist. The variables in `env_conf.xml` typically mean you will NOT have to edit the template. But, there are rare instances where it is useful to do so. Appendix A gives the details on how to do this. The template files are copied to your case directory and are available under `Tools/Templates`. The list of template files you might wish to edit are:

```
clm.cpl7.template
datm.cpl7.template
cpl.template
```

More information on the CLM configure script

The **configure** script defines the details of a clm configuration and summarizes it into a `config_cache.xml` file. The `config_cache.xml` will be placed in your case directory under `Buildconf/clmconf`. The config_definition.xml (../../bld/config_files/config_definition.xml) in `models/lnd/clm/bld/config_files` gives a definition of each clm configuration item. Many of these items are things that you would NOT change, but looking through the list gives you the valid options, and a good description of each. Coupling this with looking at the options to **configure** with "-help" as below will enable you to understand how to set the different options.

```
> cd models/lnd/clm/bld
> configure -help
```

The output to the above command is as follows:

```
SYNOPSIS
     configure [options]
OPTIONS
     User supplied values are denoted in angle brackets (<>).  Any value that contains
     white-space must be quoted.  Long option names may be supplied with either single
     or double leading dashes.  A consequence of this is that single letter options may
     NOT be bundled.

     -ad_spinup <name>   Turn on AD_SPINUP mode for bgc setting of CN [on | off] (default is off).
     -bgc <name>         Build CLM with bgc package [ none | cn | casa | cndv ] (default is none).
                         NOTE: When set to cn or casa -- also turns on the CLAMP CPP macro.
     -c13 <name>         Turn on C13 mode for bgc setting of CN [on | off] (default is off).
     -cache <file>       Name of output cache file (default: config_cache.xml).
     -cachedir <file>    Name of directory where output cache file is written (default: CLM build directory).
     -clm_root <dir>     Root directory of clm source code (default: directory above location of this script)
     -cppdefs <string>   A string of user specified CPP defines.  Appended to
                         Makefile defaults.  E.g. -cppdefs '-DVAR1 -DVAR2'
     -comp_intf <name>   Component interface to use (ESMF, MCT or cpl_$COMP) (default MCT)
     -defaults <file>    Specify full path to a configuration file which will be used to supply defaults instead of
                         the defaults in bld/config_files.  This file is used to specify model
                         configuration parameters only.  Parameters relating to the build which
                         are system dependent will be ignored.
     -dust <name>        Turn on DUST [on | off] (default is on).
     -exit_spinup <name> Turn on EXIT_SPINUP mode for bgc setting of CN [on | off] (default is off).
     -help [or -h]       Print usage to STDOUT.
     -mach <name>        Machine name to use for CCSM build.
     -maxpft <n>         Value of maxpatch_pft (default is numpft+1)
                         (required to be numpft+1 for CN or CNDV or transient cases)
     -mode <name>        CLM mode [ccsm_seq | ext_ccsm_seq]
     -nofire             Turn off wildfires for bgc setting of CN (default includes fire for CN)
     -pergro <name>      Switch enables building CLM for perturbation growth tests. [on | off] (default is off)
     -prog_seasalt<name> Turn on prognostic sea-salt (PROGSSLT cppdef token) [on | off] (default is on)
     -rtm <name>         Turn on RTM  [on | off] (default is on)
     -silent [or -s]     Turns on silent mode - only fatal messages issued.
     -snicar_frc <name>  Turn on SNICAR radiative forcing calculation. [on | off] (default is off)
     -[no]smp            Switch on [off] SMP parallelism.
     -[no]spmd           Switch on [off] SPMD parallelism.
     -target_os          Override the os setting for cross platform compilation [aix | darwin | dec_osf | irix |
                     linux | solaris | super-ux | bgl | bgp ].  Default: OS on which configure is
    executing as defined by the perl $OSNAME variable.
     -[no]test           Switch on [off] testing of Fortran compiler and external libraries.
     -usr_src <dir1>[,<dir2>[,<dir3>[...]]]
                         Directories containing user source code.
     -verbose [or -v]    Turn on verbose echoing of settings made by configure.
     -version            Echo the SVN tag name used to check out this CLM distribution.


OPTIONS used ONLY for clm standalone testing: (i.e. ONLY used when mode=ccsm_seq)

    The CCSM scripts already take care of everything here. These options exist for
    use with the CLM standalone testing in the models/lnd/clm/test/system directory.

     -ccsm_bld <name>    Turn use of CCSM build and makefiles. [on | off] (default is off)
                         Can ONLY be turned on for mode=ccsm_seq.
     -cc <name>          User specified C compiler (linux or darwin only).  Overrides Makefile default.
     -cflags <string>    A string of user specified C compiler options.  Appended to
                         Makefile defaults.
     -debug              Switch to turn on building CLM with debugging compiler options.
     -esmf_libdir <dir>  Directory containing ESMF library and esmf.mk file.
     -fc <name>          User specified Fortran compiler.  Overrides Makefile default.
     -fflags <string>    A string of user specified Fortran compiler flags.  Appended to
                         Makefile defaults.  See -fopt to override optimization flags.
     -fopt <string>      A string of user specified Fortran compiler optimization flags.
                         Overrides Makefile defaults.
     -gmake <name>       Name of the GNU make program on your system.  Supply the absolute
                         pathname if the program is not in your path (or fix your path).
     -ldflags <string>   A string of user specified load options.  Appended to
                         Makefile defaults.
     -linker <name>      User specified linker.  Overrides Makefile default of $(FC).
     -mpi_inc <dir>      Directory containing MPI include files.
     -mpi_lib <dir>      Directory containing MPI library.
     -nc_inc <dir>       Directory containing netCDF include files.
     -nc_lib <dir>       Directory containing netCDF library.
     -nc_mod <dir>       Directory containing netCDF module files.
     -voc <name>         Turn on passing of Volatile Organic Compounds (VOC's) to driver [on | off] (default is on)
```

We've given details on how to use the options in `env_conf.xml` to interact with the CLM "**configure**" and "**build-namelist**" scripts, as well as giving a good understanding of how these scripts work and the options to them. In the next section we give further details on the CLM namelist. You could customize the namelist for these options after "**configure** -case" is run.

Customizing the CLM namelist

Once a case is **configure**d, we can then customize the case further, by editing the run-time namelist for CLM. First let's list the definition of each namelist item and their valid values, and then we'll list the default values for them. Next for some of the most used or tricky namelist items we'll give examples of their use, and give you example namelists that highlight these features.

Definition of Namelist items and their default values

Here we point to you where you can find the definition of each namelist item and separately the default values for them. The default values may change depending on the resolution, land-mask, simulation-year and other attributes.

1. Definition of each Namelist Item (../../bld/namelist_files/namelist_definition.xml)

2. Default values of each CLM Namelist Item (../../bld/namelist_files/namelist_defaults_clm.xml)

Examples of using different namelist features

Below we will give examples of user namelists that activate different commonly used namelist features. We will discuss the namelist features in different examples and then show a user namelist that includes an example of the use of these features. First we will show the default namelist that doesn't activate any user options.

The default namelist

Here we give the default namelist as it would be created for a I1850CN compset at 0.9x1.25 resolution with a gx1v6 land-mask. To edit the namelist you would edit the `BuildConf/clm.buildnml.csh` under your case (or before **configure** include a user namelist with just the items you want to change). For simplicity we will just show the namelist and NOT the entire file. In the sections below, for simplicity we will just show the user namelist (`user_nl_clm`) that will add (or modify existing) namelist items to the namelist. Again, just adding the `user_nl_clm` file to your case directory, before "**configure** -case" is invoked will cause the given namelist items to appear in your clm namelist.

Example 1-2. Default CLM Namelist

```
 &clm_inparm
  co2_ppmv               = 284.7
  co2_type               = 'constant'
  dtime                  = 1800
  faerdep                =
'$DIN_LOC_ROOT/atm/cam/chem/trop_mozart_aero/aero/aerosoldep_monthly_1850_0.9x1.25_c090828.nc'
  fatmgrid               =
'$DIN_LOC_ROOT/lnd/clm2/griddata/griddata_0.9x1.25_070212.nc'
  fatmlndfrc             =
'$DIN_LOC_ROOT/lnd/clm2/griddata/fracdata_0.9x1.25_gx1v6_c090317.nc'
  finidat                = 'I1850CN_f09_c100305.clm2.r.0001-01-01-00000.nc'
  fndepdat               =
'$DIN_LOC_ROOT/lnd/clm2/ndepdata/ndep_clm_simyr1850_0.9x1.25_c091106.nc'
  fpftcon                =
'$DIN_LOC_ROOT/lnd/clm2/pftdata/pft-physiology.c100226'
  frivinp_rtm            = '$DIN_LOC_ROOT/lnd/clm2/rtmdata/rdirc.05.061026'
  fsnowaging             =
'$DIN_LOC_ROOT/lnd/clm2/snicardata/snicar_drdt_bst_fit_60_c070416.nc'
  fsnowoptics            =
'$DIN_LOC_ROOT/lnd/clm2/snicardata/snicar_optics_5bnd_c090915.nc'
  fsurdat                =
'$DIN_LOC_ROOT/lnd/clm2/surfdata/surfdata_0.9x1.25_simyr1850_c091006.nc'
  hist_crtinic           = 'NONE'
  hist_mfilt             = 1
  hist_nhtfrq            = 0
  outnc_large_files      = .true.
  rtm_nsteps             = 6
  urban_hac              = 'ON_WASTEHEAT'
  urban_traffic          = .false.
 /
```

Note that the namelist introduces some of the history namelist options that will be talked about in further detail below (`hist_mfilt` and `hist_nhtfrq`).

Adding/removing fields on your primary history file

The primary history files are output monthly, and contain an extensive list of fieldnames, but the list of fieldnames can be added to using `hist_fincl1` or removed from by adding fieldnames to `hist_fexcl1`. A sample user namelist `user_nl_clm` adding few new fields (cosine of solar zenith angle, and solar declination) and excluding a few standard fields is (ground temperature, vegetation temperature, soil temperature and soil water).:

Example 1-3. Example `user_nl_clm` namelist adding and removing fields on primary history file

```
&clm_inparm
 hist_fincl1 = 'COSZEN', 'DECL'
 hist_fexcl1 = 'TG', 'TV', 'TSOI', 'H2OSOI'
/
```

Adding auxiliary history files and changing output frequency

The `hist_fincl2` through `hist_fincl6` set of namelist variables add given history fieldnames to auxiliary history file "streams", and `hist_fexcl2` through `hist_fexcl6` set of namelist variables remove given history fieldnames from history file auxiliary "streams". A history "stream" is a set of history files that are produced at a given frequency. By default there is only one stream of monthly data files. To add more streams you add history fieldnames to `hist_fincl2` through `hist_fincl6`. The output frequency and the way averaging is done can be different for each history file stream. By default the primary history files are monthly and any others are daily. You can have up to six active history streams, but you need to activate them in order. So if you activate stream "6" by setting `hist_fincl6`, but if any of `hist_fincl2` through `hist_fincl5` are unset, only the history streams up to the first blank one will be activated.

The frequency of the history file streams is given by the namelist variable `hist_nhtfrq` which is an array of rank six for each history stream. The values of the array `hist_nhtfrq` must be integers, where the following values have the given meaning:

Positive value means the output frequency is the number of model steps between output.
Negative value means the output frequency is the absolute value in hours given (i.e -1 would mean an hour and -24 would
Zero means the output frequency is monthly. This is the default for the primary history files.

The number of samples on each history file stream is given by the namelist variable `hist_mfilt` which is an array of rank six for each history stream. The values of the array `hist_mfilt` must be positive integers. By default the primary history file stream has one time sample on it (i.e. output is to separate monthly files), and all other streams have thirty time samples on them.

A sample user namelist `user_nl_clm` turning on four extra file streams for output: daily, six-hourly, hourly, and every time-step, leaving the primary history files as monthly, and changing the number of samples on the streams to: yearly (12), thirty, weekly (28), daily (24), and daily (48) is:

Example 1-4. Example `user_nl_clm` namelist adding auxiliary history files and changing output frequency

```
&clm_inparm
 hist_fincl2 = 'TG', 'TV'
 hist_fincl3 = 'TG', 'TV'
 hist_fincl4 = 'TG', 'TV'
 hist_fincl5 = 'TG', 'TV'
 hist_nhtfrq = 0, -24, -6, -1, 1
 hist_mfilt  = 12, 30, 28, 24, 48
/
```

Removing all history fields

Sometimes for various reasons you want to remove all the history fields either because you want to do testing without any output, or you only want a very small custom list of output fields rather than the default extensive list of fields. By default only the primary history files are active, so technically using `hist_fexcl1` explained in the first example, you could list *ALL* of the history fields that are output in `hist_fexcl1` and then you wouldn't get any output. However, as the list is very extensive this would be a cumbersome thing to do. So to facilitate this `hist_empty_htapes` allows you to turn off all default output. You can still use `hist_fincl1` to turn your own list of fields on, but you then start from a clean slate. A sample user namelist `user_nl_clm` turning off all history fields and then activating just a few selected fields (ground and vegetation temperatures and absorbed solar radiation) is:

Example 1-5. Example `user_nl_clm` namelist removing all history fields

```
&clm_inparm
 hist_empty_htapes = .true.
 hist_fincl1 = 'TG', 'TV', 'FSA'
/
```

Various ways to change history output averaging flags

There are two ways to change the averaging of output history fields. The first is using `hist_avgflag_pertape` which gives a default value for each history stream, the second is when you add fields using `hist_fincl*`, you add an averaging flag to the end of the field name after a colon (for example 'TSOI:X', would output the maximum of TSOI). The types of averaging that can be done are:

A Average, over the output interval.
I Instantaneous, output the value at the output interval.
X Maximum, over the output interval.
M Minimum, over the output interval.

The default averaging depends on the specific fields, but for most fields is an average. A sample user namelist `user_nl_clm` making the monthly output fields all averages (except TSOI for the first two streams and FIRE for the 5th stream), and adding auxiliary file streams for instantaneous (6-hourly), maximum (daily), minimum (daily), and average (daily). For some of the fields we diverge from the per-tape value given and customize to some different type of optimization.

Example 1-6. Example `user_nl_clm` namelist with various ways to average history fields

```
&clm_inparm
 hist_empty_htapes = .true.
 hist_fincl1 = 'TSOI:X', 'TG',   'TV',   'FIRE',   'FSR', 'FSH',
               'EFLX_LH_TOT', 'WT'
 hist_fincl2 = 'TSOI:X', 'TG',   'TV',   'FIRE',   'FSR', 'FSH',
               'EFLX_LH_TOT', 'WT'
 hist_fincl3 = 'TSOI',   'TG:I', 'TV',   'FIRE',   'FSR', 'FSH',
               'EFLX_LH_TOT', 'WT'
 hist_fincl4 = 'TSOI',   'TG',   'TV:I', 'FIRE',   'FSR', 'FSH',
               'EFLX_LH_TOT', 'WT'
 hist_fincl5 = 'TSOI',   'TG',   'TV',   'FIRE:I', 'FSR', 'FSH',
               'EFLX_LH_TOT', 'WT'
 hist_avgflag_pertape = 'A', 'I', 'X',   'M', 'A'
 hist_nhtfrq = 0, -6, -24, -24, -24
/
```

In the example we put the same list of fields on each of the tapes: soil-temperature, ground temperature, vegetation temperature, emitted longwave radiation, reflected solar radiation, sensible heat, total latent-heat, and total water storage. We also modify the soil-temperature for the primary and secondary auxiliary tapes by outputting them for a maximum instead of the prescribed per-tape of average and instantaneous respectively. For the tertiary auxiliary tape we output ground temperature instantaneous instead of as a maximum, and for the fourth auxiliary tape we output vegetation temperature instantaneous instead of as a minimum. Finally, for the fifth auxiliary tapes we output `FIRE` instantaneously instead of as an average.

Note: We also use `hist_empty_htapes` as in the previous example, so we can list ONLY the fields that we want on the primary history tapes.

Outputting history files as a vector in order to analyze the plant function types within gridcells

By default the output to history files are the grid-cell average of all land-units, and vegetation types within that grid-cell, and output is on the full 2D latitude/longitude grid with ocean masked out. Sometimes it's important to understand how different land-units or vegetation types are acting within a grid-cell. The way to do this is to output history files as a 1D-vector of all land-units and vegetation types. In order to display this, you'll need to do extensive post-processing to make sense of the output. Often you may only be interested in a few points, so once you figure out the 1D indices for the grid-cells of interest, you can easily view that data. 1D vector output can also be useful for single point datasets, since it's then obvious that all data is for the same grid cell.

To do this you use `hist_dov2xy` which is an array of rank six for each history stream. Set it to `.false.` if you want one of the history streams to be a 1D vector. You can also use `hist_type1d_pertape` if you want to average over all the: Plant-Function-Types, columns, land-units, or grid-cells. A sample user namelist `user_nl_clm` leaving the primary monthly files as 2D, and then doing grid-cell (GRID), column (COLS), and no averaging over auxiliary tapes output daily for a single field (ground temperature) is:

Example 1-7. Example `user_nl_clm` namelist outputting some files in 1D Vector format

```
&clm_inparm
 hist_fincl2 = 'TG'
 hist_fincl3 = 'TG'
 hist_fincl4 = 'TG'
 hist_fincl5 = 'TG'
 hist_fincl6 = 'TG'
 hist_dov2xy = .true., .false., .false., .false.
 hist_type2d_pertape = ' ', 'GRID', 'COLS', ' '
 hist_nhtfrq = 0, -24, -24, -24
/
```

Warning

LAND and COLS are also options to the pertape averaging, but currently there is a bug with them and they fail to work. We expect to fix this problem by the CESM1.0 release.

Note: Technically the default for `hist_nhtfrq` is for primary files output monthly and the other auxiliary tapes for daily, so we don't actually have to include `hist_nhtfrq`, we could use the default for it. Here we specify it for clarity.

Caution

Visualizing global 1D vector files will take effort. You'll probably want to do some post-processing and possibly just extract out single points of interest to see what is going on. Since, the output is a 1D vector, of only land-points traditional plots won't be helpful. The number of points per grid-cell will also vary for anything, but grid-cell averaging. You'll need to use the output fields `pfts1d_ixy`, and `pfts1d_jxy`, to get the mapping of the fields to the global 2D array. `pfts1d_itype_veg` gives you the PFT number for each PFT. Most likely you'll want to do this analysis in a data processing tool (such as NCL, Matlab, Mathmatica, IDL, etcetera that is able to read and process NetCDF data files).

Conclusion to namelist examples

We've given various examples of namelists that feature the use of different namelist options to customize a case for particular uses. Most the examples revolve around how to customize the output history fields. This should give you a good basis for setting up your own CLM namelist.

Conclusion to customizing chapter

We’ve given extensive details on customizing cases with CLM, by choosing compsets, by changing **configure** options and interacting with the CLM "**configure**" and "**build-namelist**" scripts, and finally we’ve given details on all of the CLM namelist items. In the next chapter we talk about further ways to customize cases with CLM by creating your own datasets using the tools provided in CLM.

Chapter 2. Using the CLM tools to create your own input datasets

There are several tools provided with CLM that allow you to create your own input datasets at resolutions you choose, or to interpolate initial conditions to a different resolution, or used to compare CLM history files between different cases. The tools are all available in the `models/lnd/clm/tools` directory. Most of the tools are FORTRAN stand-alone programs in their own directory, but there is also a suite of NCL scripts in the `ncl_scripts` directory. Some of the NCL scripts are very specialized and not meant for general use, and we won't document them here. They still contain documentation in the script itself and the README file in the tools directory. But, the list of generally important scripts and programs are:

1. *cprnc* to compare NetCDF files with a time axis.
2. *interpinic* to interpolate initial condition files.
3. *mkgriddata* to create grid datasets.
4. *mkdatadomain* to create domain files from grid datasets used by datm or docn.
5. *mksurfdata* to create surface datasets from grid datasets.
6. *ncl_scripts/getregional_datasets.pl* script to extract a region or a single-point from global input datasets. See the single-point chapter for more information on this.
7. *ncl_scripts/npdepregrid.ncl* interpolate the Nitrogen deposition datasets to a new resolution.
8. *ncl_scripts/aerdepregrid.ncl* interpolate the Aerosol deposition datasets to a new resolution.

In the sections to come we will go into detailed description of how to use each of these tools in turn. First, however we will discuss the common environment variables and options that are used by all of the FORTRAN tools. Second, we go over the outline of the entire file creation process for all input files needed by CLM for a new resolution, then we turn to each tool. In the last section we will discuss how to customize files for particular observational sites.

Common environment variables and options used in building the FORTRAN tools

The FORTRAN tools all have similar makefiles, and similar options for building. All of the Makefiles use GNU Make extensions and thus require that you use GNU make to use them. They also auto detect the type of platform you are on, using "uname -s" and set the compiler, compiler flags and such accordingly. There are also environment variables that can be set to set things that must be customized. All the tools use NetCDF and hence require the path to the NetCDF libraries and include files. On some platforms (such as Linux) multiple compilers can be used, and hence there are env variables that can be set to change the FORTRAN and/or "C" compilers used. The tools other than **cprnc** also allow finer control, by also allowing the user to add compiler flags they choose, for both FORTRAN and "C", as well as picking the compiler, linker and and add linker options. Finally the tools other than **cprnc** allow you to turn optimization on (which is off by default) with the OPT flag so that the tool will run faster. To get even faster performance, the interpinic, mksurfdata, and mkgriddata programs allow you to also use the SMP to turn on multiple shared memory processors. When SMP=TRUE you set the number of threads used by the program with the OMP_NUM_THREADS environment variable.

Options used by all: cprnc, interpinic, mkdatadomain, mkgriddata, and mksurfdata

LIB_NETCDF -- sets the location of the NetCDF library.
INC_NETCDF -- sets the location of the NetCDF include files.
USER_FC -- sets the name of the FORTRAN compiler.

Options used by: interpinic, mkdatadomain, mkgriddata, and mksurfdata
MOD_NETCDF -- sets the location of the NetCDF FORTRAN module.
USER_LINKER -- sets the name of the linker to use.
USER_CPPDEFS -- adds any CPP defines to use.
USER_CFLAGS -- add any "C" compiler flags to use.
USER_FFLAGS -- add any FORTRAN compiler flags to use.
USER_LDFLAGS -- add any linker flags to use.
USER_CC -- sets the name of the "C" compiler to use.
OPT -- set to TRUE to compile the code optimized (TRUE or FALSE)

Options used by: interpinic, mkgriddata, and mksurfdata:
SMP -- set to TRUE to turn on shared memory parallelism (i.e. OpenMP) (TRUE or FALSE)
`Filepath` -- list of directories to build source code from.
`Srcfiles` -- list of source code filenames to build executable from.

Options used only by cprnc:
EXEDIR -- sets the location where the executable will be built.
VPATH -- colon delimited path list to find the source files.

More details on each environment variable.

LIB_NETCDF

This variable sets the path to the NetCDF library file (`libnetcdf.a`). If not set it defaults to `/usr/local/lib`. In order to use the tools you need to build the NetCDF library and be able to link to it. In order to build the model with a particular compiler you may have to compile the NetCDF library with the same compiler (or at least a compatible one).

INC_NETCDF

This variable sets the path to the NetCDF include directory (in order to find the include file `netcdf.inc`). if not set it defaults to `/usr/local/include`.

MOD_NETCDF

This variable sets the path to the NetCDF module directory (in order to find the NetCDF FORTRAN-90 module file when NetCDF is used with a FORTRAN-90 **use statement**. When not set it defaults to the LIB_NETCDF value.

USER_FC

This variable sets the command name to the FORTRAN-90 compiler to use when compiling the tool. The default compiler to use depends on the platform. And for example, on the AIX platform this variable is NOT used

USER_LINKER

This variable sets the command name to the linker to use when linking the object files from the compiler together to build the executable. By default this is set to the value of the FORTRAN-90 compiler used to compile the source code.

USER_CPPDEFS

This variable adds additional optional values to define for the C preprocessor. Normally, there is no reason to do this as there are very few CPP tokens in the CLM tools. However, if you modify the tools there may be a reason to define new CPP tokens.

USER_CC

This variable sets the command name to the "C" compiler to use when compiling the tool. The default compiler to use depends on the platform. And for example, on the AIX platform this variable is NOT used

USER_CFLAGS

This variable adds additional compiler options for the "C" compiler to use when compiling the tool. By default the compiler options are picked according to the platform and compiler that will be used.

USER_FFLAGS

This variable adds additional compiler options for the FORTRAN-90 compiler to use when compiling the tool. By default the compiler options are picked according to the platform and compiler that will be used.

USER_LDFLAGS

This variable adds additional options to the linker that will be used when linking the object files into the executable. By default the linker options are picked according to the platform and compiler that is used.

SMP

This variable flags if shared memory parallelism (using OpenMP) should be used when compiling the tool. It can be set to either `TRUE` or `FALSE`, by default it is set to `FALSE`, so shared memory parallelism is NOT used. When set to `TRUE` you can set the number of threads by using the OMP_NUM_THREADS environment variable. Normally, the most you would set this to would be to the number of on-node CPU processors. Turning this on should make the tool run much faster.

Caution

Note, that depending on the compiler answers may be different when SMP is activated.

OMP

This variable flags if compiler optimization should be used when compiling the tool. It can be set to either `TRUE` or `FALSE`, by default it is set to `FALSE`, so compiler optimization is NOT used. Turning this on should make the tool run much faster.

Caution

Note, you should expect that answers will be different when OMP is activated.

`Filepath`

cprnc unlike the other CLM tools doesn't need any outside code to operate. The other tools are use some code outside their directory that is in the CCSM distribution (either csm_share code or CLM source code). They all have a `Filepath` file to set the location of the source code directories that will be used.

`Srcfiles`

The `Srcfiles` lists the filenames of the source code to use when building the tool.

`EXEDIR`

The **cprnc** tool uses this variable to set the location of where the executable will be built. The default is the current directory.

`VPATH`

The **cprnc** tool uses this variable to set the colon delimited pathnames of where the source code exists. The default is the current directory.

The File Creation Process

When just creating a replacement file for an existing one, the relevant tool should be used directly to create the file. When you are creating a set of files for a new resolution there are some dependencies between the tools that you need to keep in mind when creating them. The main dependency is that the **mkgriddata** MUST be done first as the grid dataset is then input into the other tools. Also look at Table 3-1.

Creating a complete set of files for input to CLM

1. Create grid and fraction datasets

   First use **mkgriddata** to create grid and fraction datasets. See the Section called *Using **mkgriddata** to create grid datasets* for more information on this.

2. Create domain dataset (if NOT already done)

   Next use **mkdatadomain** to create a domain file for use by datm from the grid and fraction datasets just created. This is required, unless a domain file already created was input into **mkgriddata** on the previous step. See the Section called *Using **mkdatadomain** to create domain datasets for datm or docn from CLM grid datasets* for more information on this.

3. Create surface datasets

Next use **mksurfdata** to create a surface dataset, using the grid dataset as input. See the Section called *Using mksurfdata to create surface datasets from grid datasets* for more information on this.

4. Create aerosol deposition datasets

   Next use **aerdepregrid.ncl** to regrid aerosol deposition datasets to your new resolution using the grid dataset as input. See the Section called *Using **aerdepregrid.ncl** to interpolate Aerosol deposition datasets* for more information on this.

5. Create Nitrogen deposition datasets (if running CLMCN)

   Next use **ndepregrid.ncl** to regrid Nitrogen deposition datasets to your new resolution using the grid dataset as input. This is required if you are using CLMCN and otherwise is NOT. See the Section called *Using **ndepregrid.ncl** to interpolate Nitrogen deposition datasets* for more information on this.

6. Create some sort of initial condition dataset

   You then need to do one of the following three options to have an initial dataset to start from.

   a. Use spinup-procedures to create initial condition datasets

      The first option is to do the spinup procedures from arbitrary initial conditions to get good initial datasets. This is the most robust method to use. See the Section called *Spinning up the Satellite Phenology Model (CLMSP spinup)* in Chapter 4, the Section called *Spinning up the biogeochemistry Carbon-Nitrogen Model (CN spinup)* in Chapter 4, or the Section called *Spinning up the Carbon-Nitrogen Dynamic Global Vegetation Model (CNDV spinup)* in Chapter 4 for more information on this.

   b. Use **interpinic** to create interpolate existing initial condition datasets

      The next option is to interpolate from spunup datasets at a different resolution, using **interpinic**. See the Section called *Using interpinic to interpolate initial conditions to different resolutions* for more information on this.

   c. Start up from arbitrary initial conditions

      The last alternative is to run from arbitrary initial conditions without using any spun-up datasets. This is inappropriate when using CLMCN (bgc=cn or cndv) as it takes a long time to spinup Carbon pools.

      Warning

      This is NOT recommended as many fields in CLM take a long time to equilibrate.

7. Enter the new datasets into the **build-namelist** XML database

   The last optional thing to do is to enter the new datasets into the **build-namelist** XML database. See Chapter 3 for more information on doing this. This is optional because the user may enter these files into their namelists manually. The advantage of entering them into the database is so that they automatically come up when you create new cases.

Using the cprnc tool to compare two history files

cprnc is a tool shared by both CAM and CLM to compare two NetCDF history files. It differences every field that has a time-axis that is also shared on both files, and reports a summary of the difference. The summary includes the three largest differences, as well as the root mean square (RMS) difference. It also gives some summary information on the field as well. You have to enter at least one file, and up to two files. With one file it gives you summary information on the file, and with two it gives you information on the differences between the two. At the end it will give you a summary of the fields compared and how many fields were different and how many were identical.

Options:

-m = do NOT align time-stamps before comparing
-v = verbose output
-ipr
-jpr
-kpr

See the **cprnc** README (../../tools/cprnc/README) file for more details.

Using interpinic to interpolate initial conditions to different resolutions

"interpinic" is used to interpolate initial conditions from one resolution to another. In order to do the interpolation you must first run CLM to create a restart file to use as the "template" to interpolate into. Running from arbitrary initial conditions (i.e. finidat = ' ') for a single time-step is sufficient to do this. Make sure the model produces a restart file. You also need to make sure that you setup the same configuration that you want to run the model with, when you create the template file.

Command line options to **interpinic**:

-i = Input filename to interpolate from
-o = Output interpolated file, and starting template file

There is a sample template file in the `models/lnd/clm/tools/interpinic` directory and can be used to run interpolate to. However, this file was created with an older version of CLM and hence we actually recommend that you would do a short run with CLM to create a template file to use.

Example 2-1. Example of running CLM to create a template file for interpinic to interpolate to

```
> cd scripts
> create_newcase -case cr_f10_TmpltI1850CN -res f10_f10 -compset I1850CN -mach bluefire
> cd cr_f10_TmpltI1850CN
# Set starting date to end of year
> xmlchange -file env_conf.xml -id RUN_STARTDATE -val 1948-12-31
# Set year align to starting year
> xmlchange -file env_conf.xml -id DATM_CLMNCEP_YR_ALIGN -val 1948
# Set to run a cold start
> xmlchange -file env_conf.xml -id CLM_FORCE_COLDSTART -val on
# Set to run only a single day, so a restart file will be created on Jan/1/1949
> xmlchange -file env_run.xml -id STOP_N -val 1
# Then configure, build and run as normal
> configure -case
> cr_f10_TmpltI1850CN.bluefire.build
> bsub < cr_f10_TmpltI1850CN.bluefire.run
# And copy the resulting restart file to your interpinic directory
> cd ../models/lnd/clm/tools/interpinic
> cp /ptmp/$LOGIN/cr_f10_TmpltI1850CN/run/cr_f10_TmpltI1850CN.clm2.r.1949-01-01-00000.nc .
```

In the next example we build **interpinic** optimized with shared memory on for 64 threads so that it runs as fast as possible, to interpolate one of the standard 1-degree datasets to the above 10x15 template file that we created.

Example 2-2. Example of building and running interpinic to interpolate a 1-degree `finidat` dataset to 10x15

```
> cd models/lnd/clm/tools/interpinic
> gmake OPT=TRUE SMP=TRUE
> env OMP_NUM_THREADS=64 interpinic -o cr_f10_TmpltI1850CN.clm2.r.1949-01-01-00000.nc /
-i /fs/cgd/csm/inputdata/ccsm4_init/b40.1850.track1.1deg.006/0863-01-01/b40.1850.track1.1deg.006.clm2.r.0863-01-01-00000.nc
```

Tip: Running **interpinic** at high resolution can take a long time, so we recommend that you always build it optimized and with shared memory processing on, to cut down the run time as much as possible.

Warning

> **interpinic** does NOT work for CNDV (bgc=cndv).

In Appendix B we give a simpler way to run **interpinic** for several standard resolutions at once, with a script to loop over several resolutions. This is useful for CLM developers who need to create many `finidat` files at once.

Using mkgriddata to create grid datasets

mkgriddata is used to create grid, fraction, and topography datasets to run CLM at a new resolution. It is typically the first step in creating datasets needed to run CLM at a new resolution (followed by **mksurfdata**, and then the interpolation programs, **aerdepregrid.ncl**, and **ndepregrid.ncl** when running with CN).

mkgriddata namelist

mkgriddata is controlled by a namelist. There are ten different namelist items, and you need to use enough of them so that files will be output. The different types of input datasets contain different input data types, that correspond to the three different types of output files: grid, fraction, and topography. Output files for each of these will only be output if there is input data that correspond to these. If you only have input data for grid locations -- you will only get an output grid file. If you have both grid and fraction data you will get grid and fraction data files. If you also have topography data you will also get topo files.

Namelist options to **mkgriddata** include:

`mksrf_fnavyoro` -- Navy orography file to use for land fraction and surface heights.
`mksrf_frawtopo` -- Raw topography file with just surface heights.
`mksrf_fcamfile` -- CAM initial conditions file with land-fractions and topography
`mksrf_fclmgrid` -- CLM grid file
`mksrf_fccsmdom` -- CCSM domain file
`mksrf_fcamtopo` -- CAM topography file
`mksrf_lsmlon` -- number of longitude for regional grid
`mksrf_lsmlat`number of latitudes for regional grid
`mksrf_edgen` -- Northern edge for regional grid
`mksrf_edgee` -- Southern edge for regional grid
`mksrf_edges` -- Eastern edge for regional grid
`mksrf_edgew` -- Western edge for regional grid

You need to enter one of the following four options:

```
mksrf_fnavyoro    - high resolution topo dataset (topo data)
mksrf_lsmlon      - number of longitudes
mksrf_lsmlat      - number of latitudes
mksrf_edgen       - northern edge of grid (degrees)
mksrf_edgee       - eastern  edge of grid (degrees)
mksrf_edges       - southern edge of grid (degrees)
mksrf_edgew       - western  edge of grid (degrees)
```

or

```
mksrf_fcamfile    - cam topo file (grid and possibly fraction data)
```

or

```
mksrf_fccsmdom    - ccsm domain file (both grid, and fraction data)
```

or

```
mksrf_fclmgrid    - clm grid or surface dataset file (grid data)
```

Note, you can provide more than one of the needed datasets, and the output data will be determined by the datasets according to an order of precedence. The order of precedence for data is as follows:

1. `mksrf_fcamfile`
2. `mksrf_fclmgrid`
3. `mksrf_fnavyoro`

4. `mksrf_fccsmdom`

Grid data then will be established by the file with the highest precedence. CCSM domain files sometimes have latitudes and longitudes that are "off" from the standard by a small amount. By establishing an order of precedence you can ensure that grid locations exactly match a given standard file, even if the values in the domain file are off from that.

There are three different major modes for using "mkgriddata" to create grid files for CLM:

`mksrf_fnavyoro` -- Navy orography file to use for land fraction and surface heights.
`mksrf_frawtopo` -- Raw topography file with just surface heights.
`mksrf_fcamfile` -- CAM initial conditions file with land-fractions and topography
`mksrf_fclmgrid` -- CLM grid file
`mksrf_fccsmdom` -- CCSM domain file
`mksrf_fcamtopo` -- CAM topography file
`mksrf_lsmlon` -- number of longitude for regional grid
`mksrf_lsmlat`number of latitudes for regional grid
`mksrf_edgen` -- Northern edge for regional grid
`mksrf_edgee` -- Southern edge for regional grid
`mksrf_edges` -- Eastern edge for regional grid
`mksrf_edgew` -- Western edge for regional grid

You need to enter one of the following four options:

```
    mksrf_fnavyoro    - high resolution topo dataset (topo data)
    mksrf_lsmlon      - number of longitudes
    mksrf_lsmlat      - number of latitudes
    mksrf_edgen       - northern edge of grid (degrees)
    mksrf_edgee       - eastern  edge of grid (degrees)
    mksrf_edges       - southern edge of grid (degrees)
    mksrf_edgew       - western  edge of grid (degrees)
```

or

```
    mksrf_fcamfile    - cam topo file (grid and possibly fraction data)
```

or

```
    mksrf_fccsmdom    - ccsm domain file (both grid, and fraction data)
```

or

```
    mksrf_fclmgrid    - clm grid or surface dataset file (grid data)
```

Note, you can provide more than one of the needed datasets, and the output data will be determined by the datasets according to an order of precedence. The order of precedence for data is as follows:

1. `mksrf_fcamfile`
2. `mksrf_fclmgrid`
3. `mksrf_fnavyoro`

4. `mksrf_fccsmdom`

Grid data then will be established by the file with the highest precedence. CCSM domain files sometimes have latitudes and longitudes that are "off" from the standard by a small amount. By establishing an order of precedence you can ensure that grid locations exactly match a given standard file, even if the values in the domain file are off from that.

There are three different major modes for using **mkgriddata** to create grid files for CLM:

Convert CCSM domain files to CLM grid files
Create single point or regional area grid files
Convert CAM files to CLM grid files

Convert CCSM domain files to CLM grid files

CCSM domain files such as used for datm, include all the information needed to create CLM grid and fraction files.

Example 2-3. Example mkgriddata namelist to convert CCSM 4x5 domain files to CLM grid files

```
&clmexp
 mksrf_fccsmdom=
'/fs/cgd/csm/inputdata/lnd/dlnd7/domain.lnd.4x5_gx3v5.060404.nc'
 mksrf_fclmgrid=
'/fs/cgd/csm/inputdata/lnd/clm2/griddata/griddata_4x5_060404.nc'
/
```

Tip: Notice that in the above example, a clm grid file is included as well, even though it's not required. The reason for this is to ensure that the latitude and longitudes on the output files exactly match a standard grid file.

Create single point or regional area grid files

The process to create single-point or regional area CLM grid files is the same. You enter the number of latitudes and longitudes you want on your output file and the extent of the grid: North, East, South and West. You also tell **mkgriddata** that you are entering a "regional" grid and you also enter the standard Navy orography dataset (or your own orography file if desired). For a single point you simply enter "1" for the number of latitudes and longitudes, but you still enter the grid extent (of the single grid cell). Here is a sample regional namelist to create a 5x5 regional grid over the Amazon:

Example 2-4. Example mkgriddata namelist to create regional grid over Amazon

```
&clmexp
 mksrf_fnavyoro=
"/fs/cgd/csm/inputdata/lnd/clm2/rawdata/mksrf_navyoro_20min.c010129.nc"
 mksrf_lsmlon = 5
 mksrf_lsmlat = 5
 mksrf_edgee = 303.75
 mksrf_edgew = 286.25
 mksrf_edges = -15.
 mksrf_edgen = -4.
/
```

Warning

Currently you can *NOT* have regional grids that straddle both sides of the Greenich (longitude = zero) line.

Important: You should enter longitudes with values from 0 to 360 East.

Convert CAM files to CLM grid files

Older CAM initial files included all the information needed to create CLM grid files. Newer CAM files no longer include land fraction data. Hence you can use CAM files to give you the grid coordinates, but you need other data to give you the land-mask and topography. Since, CAM files no longer contain the needed information, this option is now deprecated. In most cases you should use one of the other two options.

Using mkdatadomain to create domain datasets for datm or docn from CLM grid datasets

"mkdatadomain" is used to convert CLM grid and fraction datasets into domain datasets that can be used by either the "datm" or "docn" models. Most often CLM users will want to convert the grid datasets they just created using **mkgriddata** into domain datasets to be used by datm for an "I" case. **mkdatadomain** is controlled by a namelist, and has a very straight forward operation with only four namelist items all of which are required. You specify which output mode you want "datm" or "docn", and then set the input CLM grid and frac datasets, and the output domain file.

Example 2-5. Example mkdatadomain namelist to create a domain file from CLM frac and grid data files

```
&domain_nl
 dtype = "datm"
 f_fracdata =
'/fs/cgd/csm/inputdata/lnd/clm2/griddata/fracdata_4x5_USGS_070110.nc'
 f_griddata =
'/fs/cgd/csm/inputdata/lnd/clm2/griddata/griddata_4x5_060404.nc'
 f_domain   =
'domain.lnd.fv4x5_USGS.090117.nc'
/
```

Using mksurfdata to create surface datasets from grid datasets

mksurfdata is used to create surface-datasets from grid datasets and raw datafiles at half-degree resolution to produce files that describe the surface characteristics needed by CLM (fraction of grid cell covered by different land-unit types, and fraction for different vegetation types, as well as things like soil color, and soil texture, etc.). To run **mksurfdata** you can either use the **mksurfdata.pl** which will create namelists for you using the **build-namelist** XML database, or you can run it by hand using a namelist that you provide (possibly modeled after one of the examples that is provided in the `models/lnd/clm/tools/mksurfdata` directory. In the next section we describe how to use the **mksurfdata.pl** script and the following section gives more details on running **mksurfdata** by hand and the various namelist input variables to it.

Running mksurfdata.pl

The script **mksurfdata.pl** can be used to run the **mksurfdata** program for several resolutions, simulation-years and simulation year ranges. It will create the needed namelists for you and move the files over to your inputdata directory location (and create a list of the files created, and for developers this file is also a script to import the files into the svn inputdata repository. It will also use the **build-namelist** XML database to determine the correct input files to use. And in the case of urban single-point datasets (where surface datasets are actually input into **mksurfdata**) it will do the additional processing required so that the output dataset can be used once again by **mksurfdata**. Because, it figures out namelist and input files for you, it is recommended that you use this script for creation of standard surface datasets. If you need to create surface datasets for customized cases, you will be better off running **mksurfdata** on it's own. For help on **mksurfdata.pl** you can use the "-help" option as below:

```
> cd models/lnd/clm/tools/mksurfdata
> mksurdata.pl -help
```

The output of the above command is:

```
SYNOPSIS
     mksurfdata.pl [options]
OPTIONS
     -dinlc [or -l]               Enter the directory location for inputdata
                                  (default /fs/cgd/csm/inputdata)
     -debug [or -d]               Don't actually run -- just print out what
                                  would happen if ran.
     -years [or -y]               Simulation year(s) to run over (by default 1850,2000)
                                  (can also be a simulation year range: i.e. 1850-2000)
     -help  [or -h]               Display this help.
     -res   [or -r] "resolution"   Resolution(s) to use for files (by default all ).
     -rcp   [or -c] "rep-con-path" Representative concentration pathway(s) to use for
                                  future scenarios
                                  (by default -999.9, where -999.9 means historical ).

NOTE: years, res, and rcp can be comma delimited lists.
```

To run the script with optimized **mksurfdata** for a 4x5 degree grid for 1850 conditions, on bluefire you would do the following:

Example 2-6. Example of running mksurfdata.pl to create a 4x5 resolution `fsurdat` for a 1850 simulation year

```
> cd models/lnd/clm/tools/mksurfdata
> gmake
> mksurfdata.pl -y 1850 -r 4x5
```

Running mksurfdata by Hand

In the above section we show how to run **mksurfdata** through the **mksurfdata.pl** using input datasets that are in the **build-namelist** XML database. When you are running with input datasets that are NOT available in the XML database you either need to add them as outlined in Chapter 3, or you need to run **mksurfdata** by hand, as we will outline here.

Preparing your mksurfdata namelist

When running **mksurfdata** by hand you will need to prepare your own input namelist. There are sample namelists that are setup for running on the NCAR machine bluefire. You will need to change the filepaths to run on a different machine. The list of sample namelists include

`mksurfdata.namelist` -- standard sample namelist.
`mksurfdata.pftdyn` -- sample namelist to build transient PFT land-use and land cover change over 1850 to 2005.
`mksurfdata.regional` -- sample namelist to build for a regional grid dataset (5x5_amazon)
`mksurfdata.singlept` -- sample namelist to build for a single point grid dataset (1x1_brazil)

Note, that one of the inputs `mksrf_fdynuse` is a filename that includes the filepaths to other files. The filepaths in this file will have to be changed as well. You also need to make sure that the line lengths remain the same as the read is a formatted read, so the placement of the year in the file, must remain the same, even with the new filenames.

We list the namelist items below. Most of the namelist items are filepaths to give to the input half degree resolution datasets that you will use to scale from to the resolution of your grid dataset. You must first specify the input grid dataset for the resolution to output for:

1. `mksrf_fgrid` Grid dataset

Then you must specify settings for input high resolution datafiles

1. `mksrf_ffrac` land fraction and land mask dataset
2. `mksrf_fglacier` Glacier dataset
3. `mksrf_flai` Leaf Area Index dataset
4. `mksrf_flanwat` Land water dataset
5. `mksrf_forganic` Organic soil carbon dataset
6. `mksrf_fmax` Max fractional saturated area dataset
7. `mksrf_fsoicol` Soil color dataset

8. `mksrf_fsoitex` Soil texture dataset
9. `mksrf_furban` Urban dataset
10. `mksrf_fvegtyp` PFT vegetation type dataset
11. `mksrf_fvocef` Volatile Organic Compound Emission Factor dataset

And optionally you can specify settings for:

1. `all_urban` If entire area is urban (typically used for single-point urban datasets, that you want to be exclusively urban)
2. `mksrf_fdynuse` "dynamic land use" for transient land-use/land-cover changes. This is an ASCII text file that lists the filepaths to files for each year and then the year it represents (note: you MUST change the filepaths inside the file when running on a machine NOT at NCAR). Normally we always use this file, even for creating datasets of a fixed year.
3. `mksrf_firrig` Irrigation dataset (experimental mode, normally NOT used)
4. `mksrf_ftopo` Topography dataset (this is used to limit the extent of urban regions and is used for glacier multiple elevation classes -- normally always used)
5. `mksrf_gridnm` Name of output grid resolution (if not set the files will be named according to the number of longitudes by latitudes)
6. `mksrf_gridtype` Type of grid (default is 'global')
7. `outnc_large_files` If output should be in NetCDF large file format
8. `outnc_double` If output should be in double precision (normally we turn this on)

After creating your namelist, when running on a non NCAR you will need to get the files from the inputdata repository. In order to retrieve the files needed for mksurfdata you can do the following on your namelist to get the files from the inputdata repository, using the **check_input_data** script which also allows you to export data to your local disk.

Example 2-7. Getting the raw datasets for mksurfdata to your local machine using the check_input_data script

```
> cd models/lnd/clm/tools/mksurfdata
# First remove any quotes and copy into a filename that can be read by the
# check_input_data script
> sed "s/'//g" namelist > clm.input_data_list
# Run the script with -export and give the location of your inputdata with $CSMDATA
> ../../../../../scripts/ccsm_utils/Tools/check_input_data -datalistdir . \
-inputdata $CSMDATA -check -export
```

There is one option to **mksurfdata** that is purely experimental, and should NOT be used (changes are NOT implemented in CLM4.0 to use it yet). This option is: `mksrf_firrig`. It separates out crop land-units into irrigated and non-irrigated columns. This option is experimental and not even enabled for use with CLM4.0.

Standard Practices when using mksurfdata

In this section we give the recommendations

If you look at the standard surface datasets that we have created and provided for use, there are three practices that we have consistently done in each (you also see these in the sample namelists and in the **mksurfdata.pl** script). The first is that we always output data in double precision (hence `outnc_double` is set to `.true.`. The next is that we always use the procedure for creating transient datasets (using `mksrf_fdynuse`) even when creating datasets for a fixed simulation year. This is to ensure that the fixed year datasets will be consistent with the transient datasets. When this is done a "surfdata.pftdyn" dataset will be created -- but will NOT be used in clm. If you look at the sample namelist `mksurfdata.namelist` you note that it sets `mksrf_fdynuse` to the file `pftdyn_hist_simyr2000.txt`, where the single file entered is the same PFT file used in the rest of the namelist (as `mksrf_fvegtyp`). The last practice that we always do is to always set `mksrf_ftopo`, even though multiple glacier elevation classes are NOT a part of CLM4. This is important in limiting urban areas based on topographic height, and hence is important to use all the time. In future versions of CLM the glacier multiple elevation classes will be used as well.

There are two other important practices for creating urban single point datasets. The first is that you often will want to set `all_urban` to `.true.` so that the dataset will have 100% of the gridcell output as urban rather than some mix of: urban, vegetation types, and other landunits. The next practice is that most of our specialized urban datasets have custom values for the urban parameters, hence we do NOT want to use the global urban dataset to get urban parameters -- we use a previous version of the surface dataset for the urban parameters. However, in order to do this, we need to append onto the previous surface dataset the grid and land mask/land fraction information from the grid and fraction datasets. This is done in **mksurfdata.pl** using the NCO program **ncks**. An example of doing this for the Mexicocity, Mexico urban surface dataset is as follows:

```
> ncks -A $CSMDATA/lnd/clm2/griddata/griddata_1x1pt_mexicocityMEX_c090715.nc \
$CSMDATA/lnd/clm2/surfdata/surfdata_1x1_mexicocityMEX_simyr2000_c100407.nc
> ncks -A $CSMDATA/lnd/clm2/griddata/fracdata_1x1pt_mexicocityMEX_navy_c090715.nc \
$CSMDATA/lnd/clm2/surfdata/surfdata_1x1_mexicocityMEX_simyr2000_c100407.nc
```

Note, if you look at the current single point urban surface datasets you will note that the above has already been done.

The final issue is how to build **mksurfdata**. When NOT optimized **mksurfdata** is very slow, and can take many hours to days to even run for medium resolutions such as one or two degree. So usually you will want to run it optimized. Possibly you also want to use shared memory parallelism using OpenMP with the SMP option. The problem with running optimized is that answers will be different when running optimized versus non-optimized for most compilers. So if you want answers to be the same as a previous surface dataset, you will need to run it on the same platform and optimization level. Likewise, running with or without OpenMP may also change answers (for most compilers it will NOT, however it does for the IBM compiler). However, answers should be the same regardless of the number of threads used when OpenMP is enabled. Note, that the output surface datasets will have attributes that describe whether the file was written out optimized or not, with threading or not and the number of threads used, to enable the user to more easily try to match datasets created previously. For more information on the different compiler options for the CLM4 tools see the Section called *Common environment variables and options used in building the FORTRAN tools*.

Using NCL scripts ndepregrid.ncl and aerdepregrid.ncl to interpolate aerosol deposition datasets

Unlike the other tools, these are NCAR Command Language (NCL) scripts and you will need to get a copy of NCL in order to use them. You also won't have to build an executable in order to use them, hence no Makefile is provided. NCL is provided for free download as either binaries or source code from: http://www.ncl.ucar.edu/. The NCL web-site also contains documentation on NCL and it's use.

Both the **ndepregrid.ncl** and **aerdepregrid.ncl** scripts have similar interfaces and you customize the output resolution and characteristics based on the settings of environment variables that you set (if you don't set any of the variables, the script has defaults that it will use). The list of environment variables that can be set are:

```
RES -- output resolution name
RCP -- representative concentration pathway for future scenarios (example 2.6, 4.5, 6, or 8.5)
SIM_YR -- simulation year (example 1850 or 2000)
SIM_YR_RNG -- simulation year range (example 1850-2000 or 1850-2100)
GRDFIL -- full pathname of grid file to use (in place of getting the default grid file based on the RES value)
CSMDATA -- CCSM inputdata directory
CLM_ROOT -- root directory for clm (models/lnd/clm directory)
```

Important: You *MUST* provide either RES or both GRDFIL *AND* RES. If you just give RES the default namelist database in `models/lnd/clm/bld` will be used to find the default grid file based on the resolution name RES. If you provide GRDFIL the input pathname of the gridfile provided will be used, and the output filename will include RES as part of it's name to designate it as an output file at that resolution.

Both scripts assume that you will be interpolating from a native resolution of 1.9x2.5 and using the default files found in the namelist database to interpolate from. If you want to interpolate from another resolution or use other files, you would need to edit the scripts to do so. Both scripts also use a bilinear interpolation to do the regridding. The environment variables: RCP, SIM_YR, and SIM_YR_RNG will be used to query the namelist database to determine which native dataset to interpolate from. If you don't provide valid values for these variables, it won't be able to find a dataset to interpolate from. You can use the build-namelist script to query what the valid values for these can be. Likewise, when you use RES to determine the grid file to interpolate to, it needs to be a valid value from the namelist database.

The scripts can be used to interpolate from (and create output) constant or transient datasets. Constant datasets specify the SIM_YR and set SIM_YR_RNG to `constant` (which is also the default). Transient datasets need to specify both SIM_YR and SIM_YR_RNG, where SIM_YR is set to the first year in the interval (typically 1850).

The default for CSMDATA works for NCAR computers, but will need to be set to the top level directory location of your CCSM input data on other computers. If you set this as a default for your shell when you login (for example with your `$HOME/.cshrc` if you use csh) you won't have to set it each time you run the script. CLM_ROOT will default to the proper location when you run it in the `models/lnd/clm/tools/ncl_script` directory. It is only useful if you want to run the script out of a different directory.

Using ndepregrid.ncl to interpolate Nitrogen deposition datasets

ndepregrid.ncl interpolates the Nitrogen deposition datasets from one resolution to another. It can be used to interpolate either constant datasets (`fndepdat` files) or transient datasets (`fndepdyn` files).

For example, to interpolate to an output resolution of 0.9x1.25, for a constant simulation-year of 1850, you would do the following:

```
> env RES=0.9x1.25 SIM_YR=1850 ncl ndepregrid.ncl
```

Using aerdepregrid.ncl to interpolate Aerosol deposition datasets

aerdepregrid.ncl interpolates the Aerosol deposition datasets from one resolution. It can be used to interpolate either constant datasets (for example: `aerosoldep_monthly_2000_0.9x1.25_c090828.nc`) or transient datasets (for example: `aerosoldep_monthly_1849-2006_0.9x1.25_c090830.nc`). The aerosol datasets unlike the Nitrogen deposition datasets, are named the same `faerdep` whether they contain one time-sample or a time-series of data. But, in any case, this script can be used to interpolate either kind of file.

For example, to interpolate to an output resolution of 4x5, for a transient simulation-year range of 1850 to 2100 and the rcp of 8.5, you would do the following:

```
> env RES=4x5 SIM_YR=1850 SIM_YR_RNG=1850-2100 RCP=8.5 ncl ndepregrid.ncl
```

How to Customize Datasets for particular Observational Sites

There are two ways to customize datasets for a particular observational site. The first is to customize the input to the tools that create the dataset, and the second is to over-write the default data after you've created a given dataset. Depending on the tool it might be easier to do it one way or the other. In Table 3-1 we list the files that are most likely to be customized and the way they might be customized. Of those files, the ones you are most likely to customize are: fatmlndfrc, fsurdat, faerdep, and fndepdep.

Conclusion of tools description

We've given a description of how to use the different tools with CLM to create customized datasets. In the next chapter we will talk about how to make these files available for build-namelist so that you can easily create simulations that include them. In the chapter on single-point and regional datasets we also give an alternative way to enter new datasets without having to edit files.

Chapter 3. Adding New Resolutions or New Files to the build-namelist Database

In the last chapter we gave the details on how to create new files for input into CLM. These files could be either global resolutions, regional-grids or even a single grid point. If you want to easily have these files available for continued use in your development you will then want to include them in the build-namelist database so that build-namelist can easily find them for you. You can deal with them, just by editing your namelist by hand (or using a `user_nl_clm` namelist file), or by using CLM_USRDAT_NAME. Another way to deal with them is to enter them into the database for build-namelist, so that build-namelist can find them for you. This keeps one central database for all your files, rather than having multiple locations to keep track of files. If you have a LOT of files to keep track of it also might be easier than keeping track by hand, especially if you have to periodically update your files. If you just have a few quick experiments to try, for a short time period you might be best off using the other methods mentioned above.

There are two parts to adding files to the build-namelist database. The first part is adding new resolution names which is done in the `models/lnd/clm/bld/namelist_files/namelist_definition.xml` file. The second part is actually adding the new filenames which is done in the `models/lnd/clm/bld/namelist_files/namelist_defaults_clm.xml` file. If you aren't adding any new resolutions, and you are just changing the files for existing resolutions, you don't need to edit the namelist_definition file.

Adding Resolution Names

If you are adding files for new resolutions which aren't covered in the namelist_definition file -- you'll need to add them in. The list of valid resolutions is in the id="res" entry in the `models/lnd/clm/bld/namelist_files/namelist_definition.xml` file. You need to choose a name for your new resolution and simply add it to the comma delimited list of valid_values for the id="res" entry. The convention for global Gaussian grids is number_of_latitudes x number_of_longitudes. The convention for global finite volume grids is latitude_grid_size x longitude_grid_size where latitude and longitude is measured in degrees. For regional or single-point datasets the names have a grid size number_of_latitudes x number_of_longitudes followed by an underscore and then a descriptive name such as a City name followed by an abbreviation for the Country in caps. The only hard requirement is that names be unique for different grid files. Here's what the entry for resolutions looks like in the file:

```
<entry id="res" type="char*30" category="default_settings"
       group="default_settings"
       valid_values=
"360x720,128x256,64x128,48x96,32x64,8x16,94x192,0.23x0.31,0.47x0.63,
0.9x1.25,1x1.25,1.9x2.5,2x2.5,2.5x3.33,2.65x3.33,4x5,10x15,5x5_amazon,
1x1_tropicAtl,1x1_camdenNJ,1x1_vancouverCAN,1x1_mexicocityMEX,
1x1_asphaltjungleNJ,1x1_brazil,1x1_urbanc_alpha">
Horizontal resolutions
</entry>
```

As you can see you just add your new resolution names to the end of the valid_values list.

Adding or Changing Default Filenames

To add or change the default filenames you edit the `models/lnd/clm/bld/namelist_files/namelist_defaults_clm.xml` and either change an existing filename or add a new one. Most entries in the default namelist files, include different attributes that describe the different properties that describe the differences in the datasets. Attributes include the: resolution, year to simulation, range of years to simulate for transient datafiles, the land-mask, the representative concentration pathway (rcp) for future scenarios, and the type of biogeochemistry (bgc) model used. For example the `fatmgrid` for the 1.9x2.5 resolution is as follows:

```
<fatmgrid hgrid="1.9x2.5" >lnd/clm2/griddata/griddata_1.9x2.5_060404.nc
</fatmgrid>
```

Other `fatmgrid` files are distinguished from this one by their resolution (hgrid) attribute.

What are the required files?

Different types of simulations and different types of configurations for CLM require different lists of files. The Carbon Nitrogen (cn) Biogeochemistry model for example requires `fndepdat` files, which are NOT required by other bgc modes. Transient simulations also require transient datasets, and the names of these datasets are sometimes different from the static versions (sometimes both are required as in the dynamic PFT cases).

In the following table we list the different files used by CLM, they are listed in order of importance, dependencies, and customizing. So the required files are all near the top, and the files used only under different conditions are listed later, and files with the fewest dependencies are near the top, as are the files that are least likely to be customized.

Table 3-1. Required Files for Different Configurations and Simulation Types

Filename	Config. type	Simulation type	Resol. Dependent?	Other Dependencies
	Notes			
fpftcon	ALL	ALL	No	No
	Not usually customized, as describes plant function type properties.			
fsnowoptics	ALL	ALL	No	No
	Not usually customized as describes global snow optical properties.			
fsnowaging	ALL	ALL	No	No
	Not usually customized as describes global snow aging properties.			
fatmgrid	ALL	ALL	Yes	No

Filename	Config. type	Simulation type	Resol. Dependent?	Other Dependencies
	Notes Creating, using **mkgriddata** usually gives you the amount of customization you need, as it just describes the grid and grid extents.			
fatmlndfrc	ALL	ALL	Yes	land-mask
	Describes the land-mask for points with active land, as well as the fraction of each grid-cell covered by land. You might customize it to make sure the land-fraction of your grid-cell matches the expected values for your site. But, usually you will just use what mkgriddata gives you.			
fsurdat	ALL	ALL	Yes	simulation-year
	Describes percentages of different land-units, columns and vegetation types within each grid-cell. To customize for a specific point or region you may want to use custom input datasets to mksurfdata when creating the file.			
faerdep	ALL	Constant and transient scenarios	Yes	simulation-year,simulation-year range, and possibly representative concentration pathway (rcp)
	You may customize this file to get the aerosol deposition characteristics of your site if available. Or use a higher resolution dataset when you use **aerdepregrid.ncl** to create this file. Note, both constant and transient files have the same namelist entry name.			
fndepdat	bgc=cn/cndv	Constant scenarios	Yes	simulation-year and possibly representative concentration pathway (rcp)
	You may customize this file to get the Nitrogen deposition characteristics of your site if available. Or use a higher resolution dataset when you use **ndepregrid.ncl** to create this file.			
fpftdyn	ALL	transient land-use land-cover change	Yes	Simulation year range, and possib representative concentration pathway (rcp)
	See notes on fsurdat files.			
frivinp_rtm	RTM only	ALL	No	No

Filename	Config. type	Simulation type	Resol. Dependent?	Other Dependencies
	Notes We only provide a half-degree global river routing file. If you want to model river flow for a smaller scale, or a basin regional scale, you would need to create your own custom file to do that. Normally, we turn river-routing OFF for regional or single point simulations.			
flndtopo	ALL	fine-mesh simulations (specifying land resolution as a finer grid than atmosphere resolution).	Yes	No
	You may customize to give better surface heights for your site, or input a higher resolution orography file when you create it using **mkgriddata**.			
fatmtopo	ALL	fine-mesh simulations (specifying land resolution as a finer grid than atmosphere resolution).	Yes	No
	You may customize to give better surface heights for your site, or input a higher resolution orography file when you create it using **mkgriddata**.			
fndepdyn	bgc=cn/cndv	transient simulations	Yes	Simulation year range
	You may customize this file to get the Nitrogen deposition characteristics of your site if available. Or use a higher resolution dataset when you use **ndepregrid.ncl** to create this file.			
finidat	ALL	RUN_TYPE="startup", CLM_FORCE_COLDSTART="off"	Yes	mask, maxpft, bg sim_year, start-date
	Used for starting the model from a spun-up state. Create these files by running the model for multiple years and saving the restart file from the end of a spin-up simulation.			

Chapter 4. How to run some special cases

In this chapter we describe how to run some special cases that take more than one step to do. The straightforward cases have compsets and/or build-namelist use-cases setup for them or require simple editing of a single-case. All of the cases here require you to do at least two simulations with different configurations, or require more complex editing of the case (changing the streams files).

The five cases we will describe are:

1. *Spinning up the Satellite Phenology Model (CLMSP spinup)*
2. *Spinning up the biogeochemistry Carbon-Nitrogen Model (CN spinup)*
3. *Spinning up the Carbon-Nitrogen Dynamic Global Vegetation Model (CNDV spinup)*
4. *Doing perturbation error growth tests*
5. *Running stand-alone CLM with transient historical CO_2 concentration*

Caution

The cases in this chapter are more sophisticated and require more technical knowledge and skill than cases in previous chapters. The user should be very familiar with doing simple cases before moving onto the cases described here.

Spinning up the Satellite Phenology Model (CLMSP spinup)

To spin-up the CLMSP model you merely need to run CLMSP for 50 simulation years starting from arbitrary initial conditions. You then use the final restart file for initial conditions in other simulations. Because, this is a straight forward operation we will NOT give the details on how to do that here, but leave it as an exercise for the reader. See the Example 4-3 as an example of doing this as the last step for CLMCN.

Spinning up the biogeochemistry Carbon-Nitrogen Model (CN spinup)

To get the CLMCN model to a steady state, you first run it from arbitrary initial conditions using the "accelerated decomposition spinup" (-ad_spinup in configure) mode for 600 simulation years. After this you branch from this mode in the "exit spinup" (-exit_spinup in configure), run for a simulation year, and then save a restart from that and use it as initial conditions for further spinup of CN (at least 50 simulation years).

Spinup of CLMCN

1. AD_SPINUP

   For the first step of running 600 years in "-ad_spinup" mode, you will setup a case, and then edit the values in `env_conf.xml` and `env_run.xml` so that the right configuration is turned on and the simulation is setup to run for the required length of simulation time. So do the following:

   Example 4-1. Example AD_SPINUP Simulation

   ```
   > cd scripts
   > create_newcase -case CN_spinup -res f19_g16 -compset ICN -mach bluefire
   > cd CN_spinup
   # Add "-ad_spinup on" to CLM_CONFIG_OPTS in env_conf.xml using your editor of choice
   > $EDITOR env_conf.xml
   # The following sets CLM_FORCE_COLDSTART to "on" in env_conf.xml (you could also use an editor)
   > xmlchange -file env_conf.xml -id CLM_FORCE_COLDSTART -val on
   # Configure
   > configure -case
   # The following sets RESUBMIT to 30 times in env_run.xml (you could also use an editor)
   > xmlchange -file env_run.xml -id RESUBMIT -val 30
   # The following sets STOP_OPTION to "nyears" in env_run.xml (you could also use an editor)
   > xmlchange -file env_run.xml -id STOP_OPTION -val nyears
   # The following sets STOP_N to 20 years in env_run.xml (you could also use an editor)
   > xmlchange -file env_run.xml -id STOP_N -val 20
   # The following sets STOP_DATE to Jan/1 of year 601 in env_run.xml (you could also use an editor)
   > xmlchange -file env_run.xml -id STOP_DATE -val 6010101
   # Make the output history files only annual, by adding the following to the user_nl_clm namelist
   > echo '&clm_inparm hist_nhtfrq = -8760 /' > user_nl_clm
   # Build and run normally
   > CN_spinup.build
   > bsub < CN_spinup.run
   ```

   Afterwards save the last restart file from this simulation to use in the next step.

2. EXIT_SPINIP

   Example 4-2. Example EXIT_SPINUP Simulation

   ```
   > cd scripts
   > create_newcase -case CN_exitspinup -res f19_g16 -compset ICN -mach bluefire
   > cd CN_exitspinup
   # Add "-exit_spinup on" to CLM_CONFIG_OPTS in env_conf.xml using your editor of choice
   > $EDITOR env_conf.xml
   # Change run type to branch and branch from the last year of the last simulation
   > xmlchange -file env_conf.xml -id RUN_TYPE    -val branch
   > xmlchange -file env_conf.xml -id RUN_REFCASE -val CN_spinup
   > xmlchange -file env_conf.xml -id RUN_REFDATE -val 0601-01-01
   > xmlchange -file env_conf.xml -id GET_REFCASE -val FALSE
   > configure -case
   # The following sets STOP_OPTION to "nyears" in env_run.xml (you could also use an editor)
   > xmlchange -file env_run.xml -id STOP_OPTION -val nyears
   > xmlchange -file env_run.xml -id STOP_N      -val 1
   # Go ahead and build, so that the run directory is created
   > CN_exitspinup.build
   # Now, Copy the last restart file from the earlier case into your run directory
   > cp /ptmp/$LOGIN/archive/CN_spinup/rest/CN_spinup.*.r*.0601-01-01-00000* /ptmp/$LOGIN/CN_exitspinup
   # Build and run normally
   > CN_exitspinup.build
   > bsub < CN_exitspinup.run
   ```

3. Final spinup

   Next save the last restart file from this step and use it as the "finidat" file to use for one more spinup for at least 50 years in normal mode. So do the following:

Example 4-3. Example Final CN Spinup Simulation

```
> cd scripts
> create_newcase -case CN_finalspinup -res f19_g16 -compset ICN -mach bluefire
> cd CN_finalspinup
# The following sets CLM_FORCE_COLDSTART to "on" in env_conf.xml (you could also use an editor)
> xmlchange -file env_conf.xml -id CLM_FORCE_COLDSTART -val on
# The following sets RESUBMIT to 5 times in env_run.xml (you could also use an editor)
> xmlchange -file env_run.xml -id RESUBMIT -val 5
# The following sets STOP_OPTION to "nyears" in env_run.xml (you could also use an editor)
> xmlchange -file env_run.xml -id STOP_OPTION -val nyears
# The following sets STOP_N to 10 years in env_run.xml (you could also use an editor)
> xmlchange -file env_run.xml -id STOP_N -val 10
> configure -case
# Set the finidat file to the last restart file saved in previous step
> $EDITOR Buildconf/clm.buildnml.csh
```

To assess if the model is spunup plot trends of CN variables of interest. If you see a trend, you may need to run the simulation longer. Finally save the restart file from the end of this simulation to use as an "finidat" file for future simulations.

Spinning up the Carbon-Nitrogen Dynamic Global Vegetation Model (CNDV spinup)

To spinup the CLM CNDV model -- you first follow the procedures above to spinup the CN model. Then you take the CN initial state file you created for the spinup with just CN, and run CNDV for 200 more years. We've provided such spunup files for two resolutions (f09 and f19) and two time-periods (1850 and 2000), so in this example we will use the files provided to start from. If you were to start from your own CLMCN spunup files -- the procedure would require some modification. There are no compsets using CNDV, so in `env_conf.xml` change CLM_CONFIG_OPTS to `-bgc cndv`.

Example 4-4. Example CNDV Spinup Simulation

```
> cd scripts
> create_newcase -case CNDV_spinup -res f19_g16 -compset ICN -mach bluefire
> cd CNDV_spinup
# Set run type to startup and do a cold start
> xmlchange -file env_conf.xml -id RUN_TYPE -val startup
# The following sets CLM_CONFIG_OPTS to "-bgc cndv" in env_conf.xml (you could also use an editor)
> xmlchange -file env_conf.xml -id CLM_CONFIG_OPTS  -val "-bgc cndv"
# The following sets RESUBMIT to 10 times in env_run.xml (you could also use an editor)
> xmlchange -file env_run.xml -id RESUBMIT -val 10
# The following sets STOP_OPTION to "nyears" in env_run.xml (you could also use an editor)
> xmlchange -file env_run.xml -id STOP_OPTION -val nyears
# The following sets STOP_N to 20 years in env_run.xml (you could also use an editor)
> xmlchange -file env_run.xml -id STOP_N -val 20
# Make sure you turn archiving on, so you save your files to long term archival
> xmlchange -file env_run.xml -id DOUT_L_MS -val TRUE
# Make the default primary history file annual and add an annual 1D vector auxiliary file
# By putting the following in a user_nl_clm file.
> cat << EOF > user_nl_clm
&clm_inparm
 hist_nhtfrq = -8760, -8760
 hist_mfilt  =     1, 1
 hist_fincl2 = 'TLAI', 'TSAI', 'HTOP', 'HBOT', 'NPP'
 hist_dov2xy = .true., .false.
/
> configure -case
```

```
# NOTE: If you were using your own CN spinup files you would edit the namelist to use it
# $EDITOR Buildconf/clm.buildnml.csh
#
# Now build and run as normal
> CNDV_spinup.build
> bsub < CNDV_spinup.run
```

In a data analysis tool you should examine the auxiliary file and examine the `pfts1d_wtgcell` to see where and what types of vegetation have been established. See the caution in Example 1-7 for more information on visualizing and analyzing 1D vector fields.

> **Note:** CNDV also writes out two vector fields to "hv" auxiliary files, on an annual basis by default.

Doing perturbation error growth tests

Doing perturbation error growth tests is a way to validate a port of the model to a new machine or to verify that changes are only roundoff. The steps are the same in either case, but in the discussion below I will assume you are doing a port validation to a new machine (but in parentheses I will put a reminder that it could also be for code-mods). The basic idea is to run a case on the trusted machine (trusted code) and another with initial conditions perturbed by roundoff and compare the results of the two. The difference between these two simulations (the error) will grow over time and describe a curve that we compare with the error growth on the new machine (code changes). The error growth on the new machine is the difference between the non-perturbed state on the trusted machine and the non-perturbed state on the new machine (code changes). If the new machine (code changes) are well-behaved the plot of this error growth compared to the error growth curve on the trusted machine should be similar. If the changes are NOT well-behaved the changes from the new machine (code changes) will be larger than the perturbation changes. In summary the simulations and steps that need to be performed are:

1. Run a simulation with the trusted code on the trusted machine.
2. Run a simulation with the trusted code on the trusted machine with initial conditions perturbed by roundoff (using a namelist item to do so).
3. Run a simulation with the new code on the non-trusted machine (code changes).
4. Do a plot of the RMS difference of TSOI between simulation 1 and simulation 2.
5. Do a plot of the RMS difference of TSOI between simulation 1 and simulation 3.
6. Compare the two plots in steps 4 and 5.
7. If the plots compare well the new machine (code changes) is running as well as the trusted machine.
8. If the plots do *NOT* compare well the new machine is *NOT* running as well as the trusted machine. Typically the recommendation here is to lower the optimization level on the new machine and try again (or in the case of code changes, modify or simplify the code changes to get something that should be closer).

Now we will give a detailed description of the procedure with examples and the exact steps to perform.

Using Perturbation Error Growth Analysis to Verify a Port to a New Machine

1. Running non-perturbed on trusted machine

   The first step is to run a non-perturbed case on the trusted machine. You need to run all of the steps with the same compset and same resolution. For these examples we will use 2-degree resolution with the ICN compset for 2000 conditions. You need to run for three days with a cold-start.

   Example 4-5. Example non-perturbed error growth simulation

```
> cd scripts
> create_newcase -case trustedMachinePergro0 -compset ICN -res f19_g16 \
-mach bluefire
> cd trustedMachinePergro0
# Set the non-perturbed PERGRO use-case
> xmlchange -file env_conf.xml -id CLM_NML_USE_CASE -val pergro0
# Set coldstart on so arbitrary initial conditions will be used
> xmlchange -file env_conf.xml -id CLM_FORCE_COLDSTART -val on
# Set PERGRO on in the configure
> $EDITOR env_conf.xml  # add "-pergro on" to CLM_CONFIG_OPTS
# Now configure and build
> configure -case
> trustedMachinePergro0.build
# Set it to run for three days and turn archiving off
> xmlchange -file env_run.xml -id STOP_N -val 3
> xmlchange -file env_run.xml -id DOUT_S -val FALSE
# Run the case and then you will save the history file output for later use
> bsub < trustedMachinePergro0.run
```

2. Running perturbed on the trusted machine

   The next step is to run a perturbed case on the trusted machine.

   Example 4-6. Example perturbed error growth simulation

```
> cd scripts
> create_newcase -case trustedMachinePergroRnd -compset ICN -res f19_g16 \
-mach bluefire
> cd trustedMachinePergroRnd
# Set the perturbed PERGRO use-case
> xmlchange -file env_conf.xml -id CLM_NML_USE_CASE -val pergro
# Set coldstart on so arbitrary initial conditions will be used
> xmlchange -file env_conf.xml -id CLM_FORCE_COLDSTART -val on
# Set PERGRO on in the configure
> $EDITOR env_conf.xml  # add "-pergro on" to CLM_CONFIG_OPTS
# Now configure and build
> configure -case
> trustedMachinePergroRnd.build
# Set it to run for three days and turn archiving off
> xmlchange -file env_run.xml -id STOP_N -val 3
> xmlchange -file env_run.xml -id DOUT_S -val FALSE
# Run the case and then you will save the history file output for later use
> bsub < trustedMachinePergroRnd.run
```

3. Running non-perturbed on the new machine

   The next step is to run a non-perturbed case on the new machine. Here we will demonstrate using the machine intrepid.

```
> cd scripts
> create_newcase -case newMachinePergro0 -compset ICN -res f19_g16 \
-mach intrepid
> cd newMachinePergro0
```

```
# Set the non-perturbed PERGRO use-case
> xmlchange -file env_conf.xml -id CLM_NML_USE_CASE -val pergro0
> xmlchange -file env_conf.xml -id CLM_FORCE_COLDSTART -val on
# Set PERGRO on in the configure
> $EDITOR env_conf.xml  # add "-pergro on" to CLM_CONFIG_OPTS
# Now configure and build
> configure -case
> newMachinePergro0.build
# Set it to run for three days and turn archiving off
> xmlchange -file env_run.xml -id STOP_N -val 3
> xmlchange -file env_run.xml -id DOUT_S -val FALSE
# Run the case and then you will save the history file output for later use
> bsub < newMachinePergro0.run
```

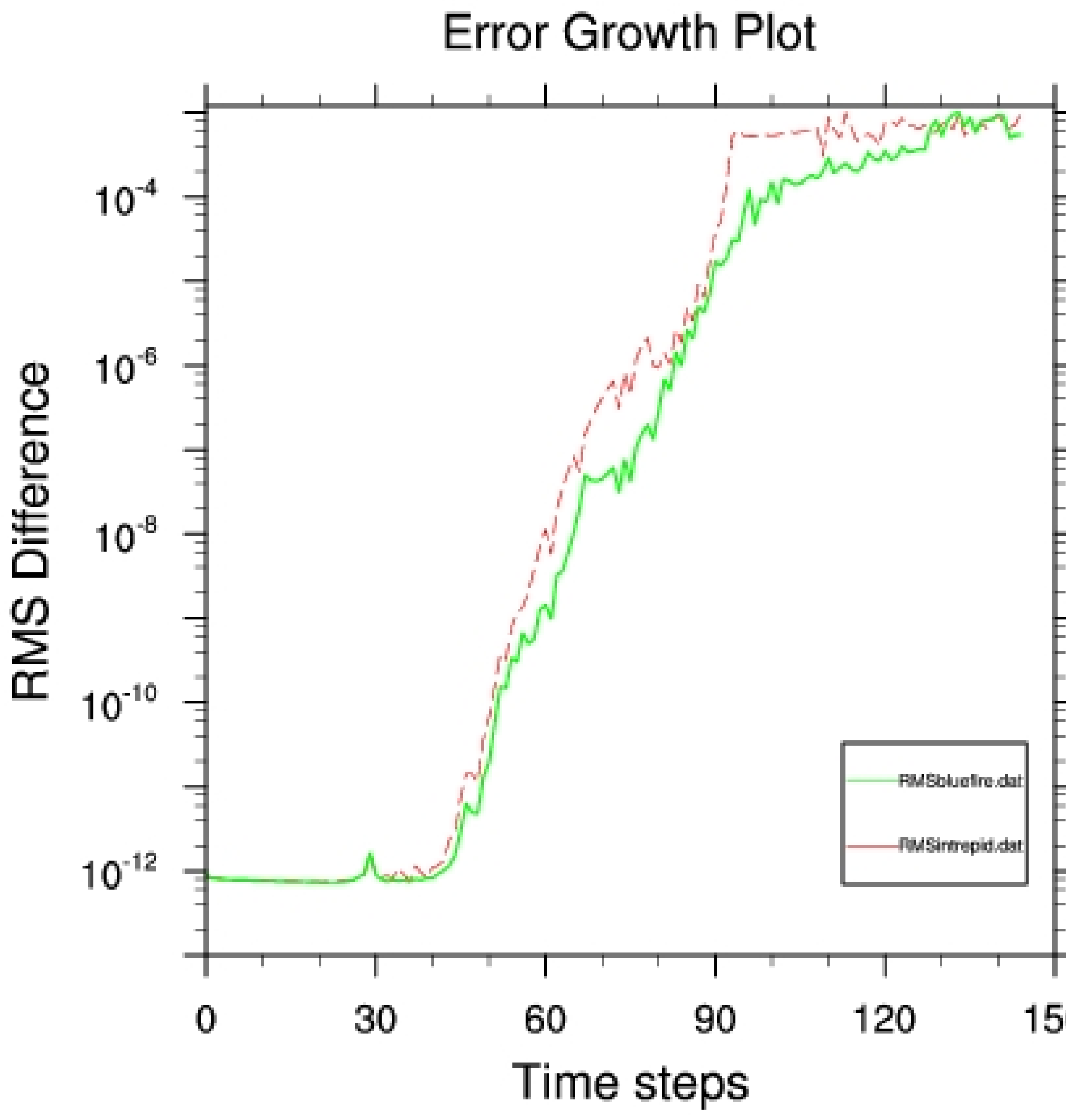
4. Plotting the differences

   You can use the **cprnc** program to compute root mean square differences between the relevant history files. See the Section called *Using the **cprnc** tool to compare two history files* in Chapter 2 for more information on it and how to build it. On many platforms you will need to set some environment variables in order to complete the build (see the Section called *Common environment variables and options used in building the FORTRAN tools* in Chapter 2 for more information on building the tools).

```
# Build the cprnc program
> cd models/lnd/clm/tools/cprnc
> gmake
# Now go to your case directory and run cprnc on the trusted-machine with and without
# perturbation
> cd ../../../../../scripts/trustedMachinePergro0
> ../../models/lnd/clm/tools/cprnc/cprnc trustedMachinePergro0.clm2.h0.001-01-01.00000.nc \
../trustedMachinePergroRnd/trustedMachinePergroRnd.clm2.h0.001-01-01.00000.nc > trustedPergro.log
# Copy the history file from the new machine to here
#
# And now run cprnc on the trusted-machine and the new machine both without perturbation
> ../../models/lnd/clm/tools/cprnc/cprnc trustedMachinePergro0.clm2.h0.001-01-01.00000.nc \
../newMachinePergro0/newMachinePergro0.clm2.h0.001-01-01.00000.nc > newPergro.log
# Now extract out the RMS differences to both
> grep RMS trustedPergro.log | awk '{print $3}' > trustedPergro.dat
> grep RMS newPergro.log     | awk '{print $3}' > newPergro.dat
# And plot the two curves up to your screen
> env TYPE=x11 RMSDAT=newPergro.dat RMSDAT2=trustedPergro.dat ncl \
../../models/lnd/clm/tools/ncl_scripts/pergroPlot.ncl
```

   Here is a sample plot for two trusted machines (using data that Sheri Mickelson provided to us). The green line is the error growth for bluefire, and the red is the error growth for intrepid.

Figure 4-1. Sample Perturbation Error Growth Curve



Running stand-alone CLM with transient historical CO_2 concentration

...

In this case you want to run a simulation with stand-alone CLM responding to changes in CO_2 for a historical period. For this example, we will start with the "I_1850-2000_CN" compset that has transient: land-use, Nitrogen and Aerosol deposition already. You could also use another compset if you didn't want these other features to be transient. In order to get CO_2 to be transient we need to edit the datm template so that we add an extra streams file to describe how CO_2 varies over the historical period. You also need a NetCDF datafile that datm can read that gives the variation. You could supply your own file, but we have a standard file that is used by CAM for this and our example will make use of this file.

Note: Most everything here has to do with changing datm rather than CLM to allow this to happen. As such the user that wishes to do this should first become more familiar with datm and read the CCSM Data Model User's Guide (http://www.ccsm.ucar.edu/models/ccsm4.0/data8/book1.html) especially as it pertains to the datm. Note, also that in this example we show how to edit the datm "buildnml" file for your case, but you could do something similar by editing the datm template.

Warning

This section documents the process for doing something that is non-standard. There may be errors with the documentation and process, and you may have to do some work before all of this works for you. If that is the case, we recommend that you do further research into understanding the process and the files, as well as understanding the datm and how it works. You may have to read documentation found in the code for datm as well as "csm_share".

The datm has "streams" files that have rough XML-like syntax and specify the location and file to get data from, as well as information on the variable names and the data locations of the grid points. The datm expects specific variable names and the datm "maps" the expected variable names from the file to the names expected by datm. The file we are working with here is a file with a single-point, that covers the entire globe (so the vertices go from -90 to 90 degrees in latitude and 0 to 360 degrees in longitude). Since it's a single point it's a little easier to work with than datasets that may be at a given horizontal resolution. The datm also expects that variables will be in certain units, and only expects a limited number of variables so arbitrary fields can NOT be exchanged this way. However, the process would be similar for datasets that do contain more than one point.

The three things that are needed: a domain file, a data file, and a streams text file. The domain file is a CF-compliant NetCDF file that has information on the grid points (latitudes and longitudes for cell-centers and vertices, mask , fraction, and areas). The datafile is a CF-compliant NetCDF file with the data that will be mapped. The streams text file is the XML-like file that tells datm how to find the files and how to map the variables datm knows about to the variable names on the NetCDF files. Note, that in our case the domain file and the data file are the same file. In other cases, the domain file may be separate from the data file.

First we are going to create a case, and we will edit the `Buildconf/datm.buildnml.csh` so that we add a CO_2 data stream in. There is a streams text file available in `models/lnd/clm/doc/UsersGuide/co2_streams.txt`, that includes file with a CO_2 time-series from 1765 to 2007.

Example 4-7. Example Transient Simulation with Historical CO_2

```
> cd scripts
> create_newcase -case DATM_CO2_TSERIES -res f19_g16 -compset I_1850-2000_CN -mach bluefire
> cd DATM_CO2_TSERIES
# Set CCSM_BGC to CO2A so that CO2 will be passed from atmosphere to land
> xmlchange -file env_conf.xml -id CCSM_BGC -val CO2A
# Set CLM_CO2_TYPE to diagnostic so that the land will use the value sent from the atmosphere
> xmlchange -file env_conf.xml -id CLM_CO2_TYPE -val diagnostic
> configure -case
> cd Buildconf
# Copy the sample streams file over
> cp ../../../models/lnd/clm/doc/UsersGuide/co2_streams.txt .
```

The first thing we will do is to edit the datm buildnml script to add a CO2 file stream in. To do this we will apply a patch with the differences needed. The patch file `addco2_datm.buildnml.diff` is in `models/lnd/clm/doc/UsersGuide` and looks like this...

```
*** datm.buildnml.csh Tue Mar 23 15:35:39 2010
--- datm.buildnml.csh Tue Mar 23 15:30:04 2010
***************
*** 18,23 ****
--- 18,24 ----
  set STREAM1TXT = "clm_qian.T62.stream.Solar.txt"
  set STREAM2TXT = "clm_qian.T62.stream.Precip.txt"
  set STREAM3TXT = "clm_qian.T62.stream.TPQW.txt"
+ set STREAMCO2  = "datm.global1val.stream.CO2.txt"

  # Only change these in env_conf -- so that the streams file will be generated consistently
  # Rerun configure if you want to change these values (doing configure -cleanall first)
***************
*** 36,49 ****
     domainFile     = '$DOMAINFILE'
     streams        = '$STREAM1TXT $year_align $year_start $year_end ',
                      '$STREAM2TXT $year_align $year_start $year_end ',
!                     '$STREAM3TXT $year_align $year_start $year_end '
     vectors        = 'null'
     mapmask        = 'nomask',
                      'nomask',
                      'nomask'
     tintalgo       = 'coszen',
                      'nearest',
!                     'linear'
    /
  EOF1

--- 37,57 ----
     domainFile     = '$DOMAINFILE'
     streams        = '$STREAM1TXT $year_align $year_start $year_end ',
                      '$STREAM2TXT $year_align $year_start $year_end ',
!                     '$STREAM3TXT $year_align $year_start $year_end ',
!                     '$STREAMCO2 1766 1766 2005 '
     vectors        = 'null'
     mapmask        = 'nomask',
                      'nomask',
+                     'nomask',
                      'nomask'
+    mapalgo        = 'bilinear',
+                     'bilinear',
+                     'bilinear',
+                       'copy'
       tintalgo       = 'coszen',
                        'nearest',
!                       'linear',
!                       'nearest'
     /
   EOF1

***************
*** 1112,1121 ****
--- 1124,1135 ----
   </streamstemplate>
   EOF1

+ cp $CASEBUILD/co2_streams.txt $STREAMCO2

   $CASETOOLS/listfilesin_streams -input_data_list -t $STREAM1TXT >> $CASEBUILD/datm.input_data_list
   $CASETOOLS/listfilesin_streams -input_data_list -t $STREAM2TXT >> $CASEBUILD/datm.input_data_list
   $CASETOOLS/listfilesin_streams -input_data_list -t $STREAM3TXT >> $CASEBUILD/datm.input_data_list
+ $CASETOOLS/listfilesin_streams -input_data_list -t $STREAMCO2  >> $CASEBUILD/datm.input_data_list

   cat >! datm_in << EOF1
     &datm_nml
```

So to apply the patch you do this...

```
> cd scripts/DATM_CO2_TSERIES/Buildconf
> patch < ../../../models/lnd/clm/doc/UsersGuide/addco2_datm.buildnml.diff
```

Once, you've done that you can build and run your case normally.

> **Note:** If the patch fails, you will have to add the changes to the `datm.buildnml.csh` found in the above patch file by hand.

Warning

The streams file above is hard-coded for the path of the file on NCAR computers. To use it on an outside machine you'll need to edit the filepath in the streams file to point to the location where you have the file.

After going through these steps, you will have a case where you have datm reading in an extra streams text file that points to a data file with CO_2 data on it that will send that data to the CLM.

Chapter 5. How to run Single-Point/Regional cases

The CLM also allows you to set up and run cases with a single-point or a local region as well as global resolutions. This is often useful for running quick cases for testing, evaluating specific vegetation types, or land-units, or running with observed data for a specific site. There are three different ways to do this: PTS_MODE, CLM_1PT_NAME, and CLM_USRDAT_NAME.

PTS_MODE -- to run for a single point using global datasets.
CLM_1PT_NAME -- to run for a supported single-point or regional dataset.
CLM_USRDAT_NAME -- to run using your own datasets (single-point or regional).

Note: PTS_MODE only works for a single point, while the other two options can also work for regional datasets as well.

Running PTS_MODE configurations

PTS_MODE enables you to run the model using global datasets, but just picking a single point from those datasets and operating on it. It can be a very quick way to do fast simulations and get a quick turnaround.

To setup a PTS_MODE simulation you use the "-pts_lat" and "-pts_lon" arguments to create_newcase to give the latitude and longitude of the point you want to simulate for (the code will pick the point on the global grid nearest to the point you give. Here's an example to setup a simulation for the nearest point at 2-degree resolution to Boulder Colorado.

```
> cd scripts
> create_newcase -case testPTS_MODE -res f19_g16 -compset I -mach bluefire \
-pts_lat 40.0 -pts_lon -105
```

Then configure, build and run as normal.

Important: By default it sets up to run with USE_MPISERIAL (in the `env_builld.xml` file) turned on, which allows you to run the model interactively. On some machines this mode is NOT supported and you may need to change it to FALSE before you are able to build.

Warning

PTS_MODE currently does *NOT* restart nor is it able to startup from global initial condition files.

Note: You can change the point you are simulating for at run-time by changing the values of PTS_LAT and PTS_LON in the `env_run.xml` file.

Running Supported Single-point/Regional Datasets

In addition to PTS_MODE the CLM supports running using single-point or regional datasets that are customized to a particular region. In the section below we tell the user how to create their own dataset, but we also support a small number of single-point and regional datasets that are ready to setup and run in the CCSM modeling system.

To get the list of supported dataset resolutions see , which results in the following:

```
build-namelist - valid values for res: default 128x256 64x128 48x96 32x64 8x16 94x192 0.23x0.31 0.47x0.63 0.9x1.25 1.9x2.5 2.65x3.33
                  default = 1.9x2.5
                  (NOTE: resolution and mask and other settings may influence what the default is)
```

The resolution names that have an underscore in them ("_") are all single-point or regional resolutions. To run with the supported single-point and regional datasets, you setup a simulation for the "pt1_pt1" resolution and give the short-name for the file to use in the `env_conf.xml` file. Then to run for the urban MexicoCity Mexico test site do the following:

```
> cd scripts
> create_newcase -case testSPDATASET -res pt1_pt1 -compset I \
-mach bluefire
> cd testSPDATASET
> xmlchange -file env_conf.xml -id CLM_1PT_NAME -val 1x1_mexicocityMEX
```

Then configure, build and run normally.

Important: Just like PTS_MODE above, By default it sets up to run with USE_MPISERIAL (in the `env_build.xml` file) turned on, which allows you to run the model interactively. On some machines this mode is NOT supported and you may need to change it to FALSE before you are able to build.

Warning

Currently the urban locations are NOT functional, because the surface datasets need to be updated.

Creating your own single-point/regional surface datasets

The file: Quickstart.userdatasets (../Quickstart.userdatasets) in the `models/lnd/clm/doc` directory gives guidelines on how to create and run with your own single-point or regional datasets. Below we reprint the above guide.

```
          Quick-Start to using your own datasets in clm4
          ===============================================

Assumptions: You are already familiar with the use of the cpl7 scripts
             for creating cases to run with "standalone" clm. See the
             Quickstart.GUIDE and the README files and documentation in
             the scripts directory for more information on this process.
             We also assume that the env variable $CSMDATA points to the
             location of the standard datasets for your machine
             (/fs/cgd/csm/inputdata on bluefire). We also assume that the
             following variables are used to point to the appropriate
             values that you want to use for your case. Mask is included
             as part of your resolution for your case, and SIM_YEAR and
             SIM_YEAR_RANGE will be set appropriately for the particular use
             case that you choose for your compset (i.e. 1850_control,
             20thC_transient etc.).

                 SIM_YEAR -------- Simulation year       (i.e. 1850, or 2000)
                 SIM_YEAR_RANGE -- Simulation year range (i.e. constant, or 1850-2000)
                 MASK ------------ Land mask             (i.e. navy, USGS, or gx1v6)

Process:

       0.) Why do this?

       An alternative to the steps below, is to create your case, and hand-edit the
       relevant namelists as appropriate with your own datasets. One reason for
       the process below is so that we can do automated testing on dataset inclusion.
       But, it also provides the following functionality to the user:
           a.) New cases with the same datasets only require a small change to
               env_conf.xml and env_run.xml (steps 5,6, and 8)
           b.) You can clone new cases based on a working case, without having to
               hand-edit all of the namelists for the new case in the same way.
           c.) The process will check for the existence of files when cases are
               configured so you can have the scripts check that datasets exist
               rather than finding out at run-time after submitted to batch.
           d.) The process checks for valid namelists, and makes it less likely
               for you to put an error or typo in the namelists.
           e.) The *.input_data_list files will be accurate for your case,
               you can use the check_input_data script to do queries on the files.
           f.) Your dataset names will be closer to standard names, and easier
               for inclusion in standard clm (with the exception of creation dates).
           g.) The regional extraction script (see 3.b below) will automatically create
               files with names following this convention.

       1.) Create your own dataset area -- link it to standard dataset location

       Create a directory to put your own datasets (such as /ptmp/$USER/my_inputdata).
       Use the script link_dirtree to link the standard datasets into this location.
       If you already have complete control over the datasets in $CSMDATA -- you
       can skip this step.

         setenv MYCSMDATA /ptmp/$USER/my_inputdata
         scripts/link_dirtree $CSMDATA $MYCSMDATA

       If you do this you can find the files you've added with...

           find $MYCSMDATA -type f -print

       and you can find the files that are linked to the standard location with...

           find $MYCSMDATA -type l -print

       2.) Establish a "user dataset identifier name" string

       You need a unique identifier for your datasets for a given resolution,
       mask, area, simulation-year, and simulation year-range. The identifier
       can be any string you want -- but we have the following suggestions:

       Suggestions for global grids:
```

```
  setenv MYDATAID ${degLat}x${degLon}

Suggestions for regional grids: either give the number of points in the grid

  setenv MYDATAID nxmpt_citySTATE
  setenv MYDATAID nxmpt_cityCOUNTRY
  setenv MYDATAID nxmpt_regionCOUNTRY
  setenv MYDATAID nxmpt_region

                or give the total size of the gridcells

  setenv MYDATAID nxmdeg_citySTATE
  setenv MYDATAID nxmdeg_cityCOUNTRY

 for example: setenv MYDATAID 10x15 -- global 10x15 grid
              setenv MYDATAID 1x1pt_boulderCO -- single-point for Boulder CO
              setenv MYDATAID 5x5pt_boulderCO -- 5x5 region around Boulder CO
              setenv MYDATAID 1x1deg_boulderCO - 1x1 degree region around Boulder CO
              setenv MYDATAID 13x12pt_f19_alaskaUSA1 - 13x12 gridcells from f19
                                          (1.9x2.5) global resolution over Alaska

3.) Add your own datasets in the standard locations in that area

3.a) Create datasets using the standard tools valid for any specific points

Use the tools in models/lnd/clm/tools to create new datasets. Tools
such as: mkgriddata, mksurfdata, mkdatadomain, and the regridding tools
in ncl_scripts

(see the models/lnd/clm/bld/namelist_files/namelist_defaults_usr_files.xml
 for the exact syntax for all files).

surfdata:    copy files into:
      $MYCSMDATA/lnd/clm2/surfdata/surfdata_${MYDATAID}_simyr${SIM_YEAR}.nc
fatmgrid:    copy files into:
      $MYCSMDATA/lnd/clm2/griddata/griddata_${MYDATAID}.nc
fatmlndfrc:  copy files into:
      $MYCSMDATA/lnd/clm2/griddata/fracdata_${MYDATAID}_${MASK}.nc
faerdep:     copy files into:
      $MYCSMDATA/lnd/clm2/snicardata/aerosoldep_monthly_${SIM_YEAR}_${MYDATAID}.nc
domainfile:  copy files into:
      $MYCSMDATA/atm/datm7/domain.clm/domain.lnd.${MYDATAID}_${MASK}.nc

3.b) Use the regional extraction script to get regional datasets from the global ones
Use the getregional_datasets.pl script to extract out regional datasets of interest.
Note, the script works on all files other than the "finidat" file as it's a 1D vector file.

For example, Run the extraction for data from 52-73 North latitude, 190-220 longitude
that creates 13x12 gridcell region from the f19 (1.9x2.5) global resolution over Alaska.

         cd models/lnd/clm/tools/ncl_scripts
         getregional_datasets.pl -sw 52,190 -ne 73,220 -id $MYDATAID -mycsmdata $MYCSMDATA

Repeat this process if you need files for multiple sim_year, and sim_year_range values.

4.) Setup your case

Follow the standard steps for executing "scripts/create_newcase" and customize
your case as appropriate. You still have to give it a valid dummy horizontal
resolution even if all your datasets are pointing at a different resolution.

i.e.

  create_newcase -case my_userdataset_test -res f19_g16 -compset I1850 -mach bluefire

The above example implies that: MASK=gx1v6, SIM_YEAR=1850, and SIM_YEAR_RANGE=constant.
TODO: Allow resolution to be be $MYDATAID and/or allow a generic "regional" resolution.
NOTE: The resolution pt1_pt1 for single point datasets is already valid.

5.) Edit the env_run.xml in the case to point to your new dataset area

Edit DIN_LOC_ROOT_CSMDATA in env_run.xml to point to $MYCSMDATA

  xmlchange -file env_run.xml -id DIN_LOC_ROOT_CSMDATA -val $MYCSMDATA

6.) Edit the env_conf.xml in the case to point to your user dataset identifier name
```

```
Edit CLM_USRDAT_NAME  to point to $MYDATAID

  xmlchange -file env_conf.xml -id CLM_USRDAT_NAME -val $MYDATAID

7.) Configure the case as normal

  configure -case

8.) Run your case as normal
```

Using getregional_datasets.pl to get a complete suite of single-point/regional surface datasets from global ones

Use the regional extraction script to get regional datasets from the global ones The getregional_datasets.pl script to extract out regional datasets of interest. Note, the script works on all files other than the "finidat" file as it's a 1D vector file. The script will extract out a block of gridpoints from all the input global datasets, and create the full suite of input datasets to run over that block. The input datasets will be named according to the input "id" you give them and the id can then be used as input to CLM_USRDAT_NAME to create a case that uses it. See the section on CLM Script Configuration Items for more information on setting CLM_USRDAT_NAME (in Chapter 1). The list of files extracted by their name used in the namelists are: `fatmgrid`, `fatmlndfrc`, `fsurdat`, `fpftdyn`, `flndtopo`, `faerdep`, `fndepdat`, `fndepdyn`, and the datm file `domainfile`. For more information on these files see the Table on required files.

The alternatives to using this script are to use PTS_MODE, discussed earlier, or creating the files individually using the different file creation tools (given in the Tools Chapter). Creating all the files individually takes quite a bit of effort and time. PTS_MODE has some limitations as discussed earlier, but also as it uses global files, is a bit slower when running simulations than using files that just have the set of points you want to run over. Another advantage is that once you've created the files using this script you can customize them if you have data on this specific location that you can replace with what's already in these files.

The script requires the use of both "Perl" and "NCL". See the NCL Script section in the Tools Chapter on getting and using NCL and NCL scripts. The main script to use is a perl script which will then in turn call the NCL script that actually creates the output files. The ncl script gets it's settings from environment variables set by the perl script. To get help with the script use "-help" as follows:

```
> cd models/lnd/clm/tools/ncl_scripts
> getregional_datasets.pl -help
```

The output of the above is:

```
SYNOPSIS
     getregional_datasets.pl [options]    Extracts out files for a single box region from the global
                                   grid for the region of interest. Choose a box determined by
                                   the NorthEast and SouthWest corners.
OPTIONS
     -debug [or -d]                Just debug by printing out what the script would do.
                                   This can be useful to find the size of the output area.
     -help [or -h]                 Print usage to STDOUT.
     -mask "landmask"              Type of land-mask (i.e. navy, gx3v7, gx1v6 etc.) (default gx1v6)
     -mycsmdata "dir"              Root directory of where to put your csmdata.
                                   (default /home/erik/inputdata or value of CSMDATA env variable)
     -mydataid "name" [or -id]     Your name for the region that will be extracted.               (REQUIRED)
                                   Recommended name: grid-size_global-resolution_location (?x?pt_f??_????)
                                   (i.e. 12x13pt_f19_alaskaUSA for 12x13 grid cells from the f19 global resolution over Alaska)
     -NE_corner "lat,lon" [or -ne] North East corner latitude and longitude                       (REQUIRED)
     -res "resolution"             Global horizontal resolution to extract data from (default 1.9x2.5).
     -rcp "pathway"                Representative concentration pathway for future scenarios
                                   Only used when simulation year range ends in a future
                                   year, such as 2100.
                                   (default 8.5).
     -sim_year   "year"            Year to simulate for input datasets (i.e. 1850, 2000) (default 2000)
(default 2000)
     -sim_yr_rng "year-range"      Range of years for transient simulations
                                   (i.e. 1850-2000, 1850-2100,  or constant) (default constant)

     -SW_corner "lat,lon" [or -sw] South West corner latitude and longitude                        (REQUIRED)
     -verbose [or -v]              Make output more verbose.
```

The *required* options are: `-id`, `-ne`, and `-se`, for the output identifier name to use in the filenames, latitude and longitude of the Northeast corner, and latitude and longitude of the SouthEast corner (in degrees). Options that specify which files will be used are: `-mask`, `-res`, `-rcp`, `-sim_year`, and `-sim_yr_rng` for the land-mask to use, global resolution name, representative concentration pathway for future scenarios, simulation year, and simulation year range. The location of the input and output files will be determined by the option `-mycsmdata` (can also be set by using the environment variable `$CSMDATA`). If you are running on a machine like at NCAR where you do NOT have write permission to the CCSM inputdata files, you should use the `scripts/link_dirtree` script to create softlinks of the original files to a location that you can write to. This way you can use both your new files you created as well as the original files and use them from the same location.

The remaining options to the script are `-debug`, and `-verbose`. `-debug` is used to show what would happen if the script was run, without creating the actual files. `-verbose` adds extra log output while creating the files so you can more easily see what the script is doing.

For example, Run the extraction for data from 52-73 North latitude, 190-220 longitude that creates 13x12 gridcell region from the f19 (1.9x2.5) global resolution over Alaska.

```
> cd models/lnd/clm/tools/ncl_scripts
> getregional_datasets.pl -sw 52,190 -ne 73,220 -id 13x12pt_f19_alaskaUSA -mycsmdata $CSMDATA
```

Repeat this process if you need files for multiple sim_year, resolutions, land-masks, and sim_year_range values.

Appendix A. Editing Template Files Before Configure

The last kind of customization that you can do for a case, before configure is run is to edit the templates. The clm template is in `models/lnd/clm/bld/clm.cpl7.template`, the datm template is in `models/atm/datm/bld/datm.cpl7.template`, and the driver templates are in the `models/drv/bld` directory and are named: `ccsm.template` and `cpl.template`. When a case is created they are also copied to the `Tools/Templates` directory underneath your case. If you want to make changes that will impact all your cases, you should edit the template files under the `models` directory, but if you want to make a change ONLY for a particular case you should edit the template under that specific case.

Note: Editing the template files is NOT for the faint of heart! We recommend this ONLY for experts! It's difficult to do because the template is a script that actually creates another script. So part of the script is echoing the script to be created and part of it is a script that is run when "configure -case" is run. As a result any variables in the part of the script that is being echoed have to be escaped like this:

```
\$VARIABLE
```

But, in other parts of the script that is run, you can NOT escape variables. So you need to understand if you are in a part of the script that is echoing the script to be created, or in the part of the script that is actually run.

If you can customize your case using: compsets, `env_*.xml` variables, or a user namelist, as outlined in Chapter 1 you should do so. The main reason to actually edit the template files, is if you are in a situation where the template aborts when you try it run it when "configure -case" is run. The other reason to edit the template is if you are CLM developer and need to make adjustments to the template because of code or script updates.

The outline of the clm template is as follows:

```
# set up options for clm configure and then run clm configure
$CODEROOT/lnd/clm*/bld/configure <options>
# set up options for clm build-namelist and then run clm build-namelist
$CODEROOT/lnd/clm*/bld/build-namelist <options>
# echo the $CASEBUILD/clm.buildnml.csh script out
cat >! $CASEBUILD/clm.buildnml.csh << EOF1
# NOTE: variables in this section must be escaped
EOF1
# Remove temporary namelist files

# echo the $CASEBUILD/clm.buildexe.csh script out
cat > $CASEBUILD/clm.buildexe.csh <<EOF2
# NOTE: variables in this section must be escaped
EOF2
# Remove temporary configure files
```

Appendix B. Using the Script `runinit_ibm.csh` to both Run CLM and Interpolate Datasets

The script `runinit_ibm.csh` can be used on the NCAR bluefire machine to run CLM to create a template file and then run **interpinic** and do this over a variety of standard resolutions. By default it is setup to loop over the following resolutions:

```
foreach res ( "1.9x2.5" "10x15" "4x5" "0.9x1.25" "2.5x3.33" "0.47x0.63" "48x96" )
```

It is also only setup to run CLMCN and only particular masks for each resolution. But, the script can be modified by the user to run over whatever list you would like it to. It is also hooked up to the **build-namelist** XML database, so will only use the datasets that are part of the database, see Chapter 3 to see how to add files to the database. The script runs CLM only using OpenMP threading and as such can be run interactively, but it can also be submitted to the batch que.

Appendix C. Scripts for testing CLM

Technically, you could use the customization we gave in Chapter 1 to test various configuration and namelist options for CLM. Sometimes, it's also useful to have automated tests though to test that restarts give exactly the same results as without a restart. It's also useful to have automated tests to run over a wide variety of configurations, resolutions, and namelist options. To do that we have several different types of scripts set up to make running comprehensive testing of CLM easy. There are two types of testing scripts for CLM. The first are the CCSM test scripts, which utilize the **create_newcase** scripts that we shown how to use in this User's Guide. The second are a set of stand-alone scripts that use the CLM **configure** and **build-namelist** scripts to build and test the model as well as testing the CLM tools as well. Below we will go into further details of how to use both methods.

Testing CLM Using the CCSM Test Scripts

We first introduce the test scripts that work for all CCSM components. We will use the **create_test** and then the **create_test_suite** scripts. The **create_test** runs a specific type of test, at a given resolution, for a given compset using a given machine. There is a list of different tests, but the "ERI" tests do several things at once, running from startup, as well as doing exact branch and restart tests. So to run "ERI" testing at 2-degree with the I1850CN compset on bluefire you do the following.

```
> cd scripts
> create_test -testname ERI.f19_g16.I1850CN.bluefire
> cd ERI.f19_g16.I1850CN.bluefire.$id
> ERI.f19_g16.I1850CN.bluefire.$id.build
> bsub < ERI.f19_g16.I1850CN.bluefire.$id.test
```

When the test is done it will update the file `TestStatus` with either a PASS or FAIL message.

To run a suite of tests from a list of tests with syntax similar to above you use **create_test_suite** as follows passing it a ASCII list of tests. There are already some test lists in the `scripts/ccsm_utils/Testlists` directory a few of which are specific to CLM. To run for the CLM bluefire test list, on bluefire, you would do the following:

```
> cd scripts
> create_test_suite -input_list ccsm_utils/Testlists/bluefire.clm.auxtest
# Submit the suite of tests (note $id refers to the integer job number for this job)
> cs.submit.$id.bluefire
# Later check the tests with...
> cs.status.$id
# The above will give a PASS or FAIL message for each test.
```

For more information on doing testing with the CCSM scripts see the CCSM4.0 User's Guide (http://www.ccsm.ucar.edu/models/ccsm4.0/ccsm_doc/book1.html) on testing.

Testing CLM Using the CLM Stand-Alone Testing Scripts

In the `models/lnd/clm/test/system` directory there is a set of test scripts that is specific to stand-alone CLM. It does testing on configurations harder to test for in the CCSM test scripts, and also allows you to test the CLM tools such as **mkgriddata** and **mksurfdata**. The main driver script is called `test_driver.sh` and it can run both interactively as well as being submitted to the batch queue. Like other scripts you can get help on it by running the "-help" option as: **test_driver.sh -help**. There is also a `README` file that gives details about environment variables that can be given to **test_driver.sh** to change it's operation.

To submit a suite of stand-alone tests to the batch que:

```
> cd models/lnd/clm/test/system
> test_driver.sh
```

You can also run tests interactively:

```
> cd models/lnd/clm/test/system
> test_driver.sh -i
```

The output of the help option is as follows:

```
ERROR: machine poorman.cgd.ucar.edu not currently supported
```

A table of the list of tests and the machines they are run on is available from: test_table.html (../../test/system/test_table.html)