CCSM4 Data Model v8: User's Guide

Brian Kauffman
NCAR

Tony Craig
NCAR

CCSM4 Data Model v8: User's Guide
by Brian Kauffman and Tony Craig

Table of Contents

Chapter 1.

Introduction

Overview

The CCSM4 data models perform the basic function of reading external data files, modifying that data, and then sending it to the coupler via standard CCSM coupling interfaces. The coupler and other models have no fundamental knowledge of whether another component is fully active or just a data model. In some cases, data models are prognostic, that is, they also receive and use data sent by the coupler to the data model. But in most cases, the data models are not running prognostically and have no need to receive any data from the coupler.

The CCSM4 data models have been parallelized and share significant amounts of source code. Methods for reading and interpolating data have been established and can easily be reused. There is a natural hierarchy in the system. The data model calls strdata ("stream data") methods which then call stream methods. There are inputs associated with the data model, strdata, and streams to configure the setup. The stream methods are responsible for managing lists of input data files and their time axis. The information is then passed up to the strdata methods where the data is read and interpolated in space and time. The interpolated data is passed up to the data model where final fields are derived, packed, and returned to the coupler.

Design

The strdata implementation is hardwired to execute a set of specific operations associated with reading and interpolating data in space and time. The text box below shows the sequencing of the computation of model fields using the strdata methods.

```
STRDATA Implementation:
  for the current model time
  determine nearest lower and upper bound data from the input dataset
  if that is new data then
     read lower and upper bound data
     fill lower and upper bound data
     spatially map lower and upper bound data to model grid
  endif
  time interpolate lower and upper bound data to model time
  return fields to data model
```

IO Through Data Models

The two timestamps of input data that bracket the present model time are read first. These are called the lower and upper bounds of data and will change as the model advances. Those two sets of inputdata are first filled based on the method and masking set by the user. That operation occurs on the input data grid. The lower and upper bound data are then spatially mapped to the model grid based upon map methods and masking set by the user in namelist. Spatial interpolation only occurs if the input data grid and model grid are not the identical, and this is determined in the strdata module automatically. Time interpolation is the final step and is done using a time interpolation method specified by the user in namelist. A final set of fields is then available to the data model on the model grid and for the current model time.

There are two primary costs associated with strdata, reading data and spatially mapping data. Time interpolation is relatively cheap in the current implementation. As much as possible, redundant operations are minimized. Fill and mapping weights

are generated at initialization and saved. The upper and lower bound mapped input data is saved between time steps to reduce mapping costs in cases where data is time interpolated more often than new data is read. If the input data timestep is relatively small (for example, hourly data as opposed to daily or monthly data) the cost of reading input data can be quite large. Also, there can be significant variation in cost of the data model over the coarse of the run, for instance, when new input-data must be read and interpolated, although it's relatively predictable. The present implementation doesn't support changing the order of operations, for instance, time interpolating the data before spatial mapping. Although because the present computations are always linear, changing the order of operations wouldn't fundamentally change the results. The present order of operations generally minimizes the mapping cost for typical data model use cases.

There are several limitations in both options and usage within the data models at the present time. Spatial interpolation can only be performed from a two-dimensional latitude-longitude input grid. The target grid can be arbitrary but the source grid must be able to be described by simple one-dimensional lists of longitudes and latitudes, although they don't have to have equally spaced.

At the present time, data models can only read netcdf data, and IO is handled through either standard netcdf interfaces or through the pio library using either netcdf or pnetcdf. If standard netcdf is used, global fields are read and then scattered one field at a time. If pio is used, then data will be read either serially or in parallel in chunks that are approximately the global field size divided by the number of io tasks. If pnetcdf is used through pio, then the pnetcdf library must be included during the build of CCSM. The pnetcdf path and option is hardwired into the Macros file for the specific machine. To turn on pnetcdf in the build, make sure the Macros variables PNETCDF_PATH, INC_PNETCDF, and LIB_PNETCDF are set and that the pio CONFIG_ARGS sets the PNETCDF_PATH argument. See the CCSM users guide for more information.

Beyond just the option of selecting IO with pio, several namelist are available to help optimize pio IO performance. Those are io_type, num_iotasks, io_root, and io_stride. The number of total mpi tasks that can be used for IO is limited to the total number of tasks used by the data model. Often though, fewer io tasks result in improved performance. In general, [io_root + (num_iotasks-1)*io_stride + 1] has to be less than the total number of data model tasks. In practice, pio seems to perform optimally somewhere between the extremes of 1 task and all tasks, and is highly machine and problem dependent.

Restart Files

Restart files are generated automatically by the data models based upon a flag sent from the coupler. The restart files must meet the naming convention specified by the CCSM project and an rpointer file is generated at the same time. An rpointer file is a *restart pointer* file which contains the name of the most recently created restart file. Normally, if restart files are read, the restart filenames are specified in the rpointer file. Optionally though, there are namelist variables such as restfilm to specify the restart filenames via namelist. If those namelist are set, the rpointer file will be ignored. The standard procedure in CCSM is to use the rpointer files to specify the restart filenames. In many cases, no model restart is required for the data models to restart exactly. This is because there is no memory between timesteps in many of the data model science modes. If a model restart is required, it will be written automatically and then must be used to continue the previous run.

There are separate stream restart files that only exist for performance reasons. A stream restart file contains information about the time axis of the input streams. This information helps reduce the start costs associated with reading the input dataset time axis information. If a stream restart file is missing, the code will restart without it but may need to reread data from the input data files that would have been stored in the stream restart file. This will take extra time but will not impact the results.

Hierarchy

The hierarchy of data models, strdata, and streams also compartmentalize grids and fields. Some data models communicate with the coupler with fields on multiple model grids. For instance, the data land model communicates with the coupler on two different grids, a land grid and a runoff grid. In those cases, there are multiple model grids and so there will be multiple strdata namelist, one for each unique data model grid. For each strdata namelist, data is interpolated to a single model grid. But each strdata namelist input can have multiple stream description files and each stream input file can contain data on a different grid. The strdata module will gracefully read the different streams of input data and interpolate both spatially and temporally to the appropriate final model grid and model time. The text box below provides a schematic of the hierarchy

```
  driver     :                 call data land model
  data model :                   data land model
  data model :           land_data               runoff_data
  data model :            grid1                    grid2
  data model :           strdata1                 strdata2
  strdata    :    interp1a interp1b interp1c      interp2a
  strdata    :    stream1a stream1b stream1c      stream2a
  stream     :     grid1a   grid1b   grid1c        grid2a
  stream     :   file1a_01 file1b_01 file1c_01    file2a_01
  stream     :     ...                ...          ...
  stream     :   file1a_04           file1c_96    file2a_12
```

Users will primarily setup different data model configurations through existing namelist settings. The strdata and stream input options and format are identical for all data models. The data model specific namelist has significant overlap between data models, but each data model has a slightly different set of input namelist variables and each model reads that namelist from a unique filename. The detailed namelist options for each data model will be described later, but each model will specify a filename or filenames for strdata namelist input and each strdata namelist will specify a set of stream input files.

To continue with the above example, the following inputs would be consistent with the above figure. This is a data land model that has data on two different model grids, land_data on grid1 and runoff_data on grid2. Therefore two different strdata namelist will be needed. The data model namelist input file is hardwired to "dlnd_in" and in this case, the namelist would look something like

```
file="dlnd_in":
&dlnd_nml
  lnd_in = 'dlnd_lnd_in'
  rof_in = 'dlnd_rof_in'
  decomp = '1d'
/
```

The lnd_in and rof_in specify the filenames associated with the strdata namelist input for the land and runoff data separately. The land and runoff strdata namelist would then look like

```
file="dlnd_lnd_in":
&shr_strdata_nml
  dataMode   = 'CPLHIST'
  domainFile = 'grid1.nc'
  streams    = 'stream1a',
               'stream1b',
               'stream1c'
  mapalgo    = 'interp1a',
               'interp1b',
               'interp1c'
/
```

```
file="dlnd_rof_in":
&shr_strdata_nml
  dataMode   = 'RX1'
  domainFile = 'grid2.nc'
  streams    = 'stream2a'
  mapalgo    = 'interp2a'
/
```

Four stream description files are then expected to be available, stream1a, stream1b, stream1c, and stream2a. Those files specify the input data filenames, input data grids, and input fields that are expected among other things. For instance, one of the stream description files might look like

```
<stream>
      <comment>
         NCEP "normal year" data
      </comment>
      <dataSource>
         NCEP
      </dataSource>
      <fieldInfo>
         <variableNames>
            dn10  dens
            slp_  pslv
            q_10  shum
            t_10  tbot
            u_10  u
            v_10  v
         </variableNames>
         <filePath>
            /fis/cgd/cseg/csm/inputdata/atm/datm7/NYF
         </filePath>
         <offset>
            0
         </offset>
         <fileNames>
            nyf.ncep.T62.050923.nc
         </fileNames>
      </fieldInfo>
      <domainInfo>
         <variableNames>
            time   time
            lon    lon
            lat    lat
            area   area
            mask   mask
         </variableNames>
         <filePath>
            /fis/cgd/cseg/csm/inputdata/atm/datm7/NYF
         </filePath>
         <fileNames>
            nyf.ncep.T62.050923.nc
         </fileNames>
      </domainInfo>
</stream>
```

The stream files are not Fortran namelist format. Their format and options will be described later. In general, these examples of input files are not complete, but they do show the general hierarchy and feel of the data model input.

Summary

In summary, for each data model a top level namelist will be set that will point to a file that contains the strdata namelist. That namelist will specify the data model mode, stream description text files, and interpolation options. The stream description files will be provided as separate input files and contain the files and fields that need to be read.

From a user perspective, for any data model, it's important to know what modes are supported and the internal field names in the data model. That information will be used in the strdata namelist and stream input files.

Next Sections

In the next sections, more details will be presented including a full description of the science modes and namelist settings for the data atmosphere, data land, data ocean, and data ice models; namelist settings for the strdata namelist input; a description of the format and options for the stream description input files; and a list of internal field names for each of the data components. The internal data model field names are important because they are used to setup the stream description files and to map the input data fields to the internal data model field names.

Chapter 2.

Input Data Streams

Overview

An *input data stream* is a time-series of input data files where all the fields in the stream are located in the same data file and all share the same spatial and temporal coordinates (ie. are all on the same grid and share the same time axis). Normally a time axis has a uniform dt, but this is not a requirement.

The data models can have multiple input streams.

The data for one stream may be all in one file or may be spread over several files. For example, 50 years of monthly average data might be contained all in one data file or it might be spread over 50 files, each containing one year of data.

The data models can *loop* over stream data -- repeatedly cycle over some subset of an input stream's time axis. When looping, the models can only loop over whole years. For example, an input stream might have SST data for years 1950 through 2000, but a model could loop over the data for years 1960 through 1980. A model *cannot* loop over partial years, for example, from 1950-Feb-10 through 1980-Mar-15.

The input data must be in a netcdf file and the time axis in that file must be CF-1.0 compliant.

There are two main categories of information that the data models need to know about a stream:

- data that describes what a user wants -- what streams to use and how to use them -- things that can be changed by a user.
- data that describes the stream data -- meta-data about the inherent properties of the data itself -- things that cannot be changed by a user.

Generally, information about what streams a user wants to use and how to use them is input via the strdata ("stream data") Fortran namelist, while meta-data that describes the stream data itself is found in an xml-like text file called a "stream description file."

Specifying What Streams to Use

The data models have a namelist variable that specifies which input streams to use and, for each input stream, the name of the corresponding stream description file, what years of data to use, and how to align the input stream time axis with the model run time axis. This input is set in the strdata namelist input.

General format:

```
&shr_strdata_nml
   streams = 'stream1.txt year_align year_first year_last ',
             'stream2.txt year_align year_first year_last ',
             ...
             'streamN.txt year_align year_first year_last '
/
```

Actual example:

```
&shr_strdata_nml
   streams = 'clm_qian.T62.stream.Solar.txt  1 1948 2004 ',
             'clm_qian.T62.stream.Precip.txt 1 1948 2004 ',
             'clm_qian.T62.stream.TPQW.txt   1 1948 2004 '
```

/

where:

`streamN.txt`

the stream description file, a plain text file containing details about the input stream (see below)

`year_first`

the first year of data that will be used

`year_last`

the last year of data that will be used

`year_align`

a model year that will be aligned with data for year_first

Stream Description File

The *stream description file* is not Fortran namelist, but a locally built xml-like parsing implementation. Sometimes it is called a "stream dot-text file" because it has a ".txt" extension. Normally these stream description files are built automatically by the CCSM scripts, although they can be custom made for non-standard stream data. They contain data that specifies the names of the fields in the stream, the names of the input data files, and the file system directory where the data files are located. In addition, a few other options are available such as the time axis offset parameter.

The data elements found in the stream description file are:

`comment`

A general comment about the data -- this is not used by the model.

`dataSource`

A comment about the source of the data -- this is not used by the model.

`fieldInfo`

Information about the field data for this stream...

`variableNames`

A list of the field variable names. This is a paired list with the name of the variable in the netCDF file on the left and the name of the corresponding model variable on the right. This is the list of fields to read in from the data file, there may be other fields in the file which are not read in (ie. they won't be used).

`filePath`

The file system directory where the data files are located.

`fileNames`

The list of data files to use. If there is more than one file, the files must be in chronological order, that is, the dates in time axis of the first file are before the dates in the time axis of the second file.

`tInterpAlgo`

The option is obsolete and no longer performs a function. Control of the time interpolation algorithm is in the strdata namelist, tinterp_algo and taxMode

`offset`

This offset allows a user to shift the time axis of a data stream by a fixed and constant number of seconds. For instance, if a data set contains daily average data with timestamps for the data at the end of the day, it might be appropriate to shift the time axis by 12 hours so the data is taken to be at the middle of the day instead of the end of the day. This feature supports only simple shifts in seconds as a way of correcting input data time axes without having to modify the input data time axis manually. This feature does not support more complex shifts such as end of month to mid-month. But in conjunction with the time interpolation methods in the strdata input, hopefully most user needs can be accommodated with the two settings. Note that a positive offset advances the input data time axis forward by that number of seconds.

`domainInfo`

Information about the domain data for this stream...

`variableNames`

A list of the domain variable names. This is a paired list with the name of the variable in the netCDF file on the left and the name of the corresponding model variable on the right. This data models require five variables in this list. The names of model's variables (names on the right) must be: "time," "lon," "lat," "area," and "mask."

`filePath`

The file system directory where the domain data file is located.

`fileNames`

The name of the domain data file. Often the domain data is located in the same file as the field data (above), in which case the name of the domain file could simply be the name of the first field data file. Sometimes the field data files don't contain the domain data required by the data models, in this case, one new file can be created that contains the required data.

Actual example:

```
<stream>
      <comment>
         NCEP "normal year" data
      </comment>
      <dataSource>
         NCEP
      </dataSource>
      <fieldInfo>
         <variableNames>
            dn10  dens
            slp_  pslv
            q_10  shum
            t_10  tbot
            u_10  u
            v_10  v
         </variableNames>
```

```
            <filePath>
               /fis/cgd/cseg/csm/inputdata/atm/datm7/NYF
            </filePath>
            <offset>
               0
            </offset>
            <fileNames>
               nyf.ncep.T62.050923.nc
            </fileNames>
         </fieldInfo>
         <domainInfo>
            <variableNames>
               time   time
               lon    lon
               lat    lat
               area   area
               mask   mask
            </variableNames>
            <filePath>
               /fis/cgd/cseg/csm/inputdata/atm/datm7/NYF
            </filePath>
            <fileNames>
               nyf.ncep.T62.050923.nc
            </fileNames>
         </domainInfo>
      </stream>
```

Chapter 3.

Stream Data

Namelist

The *strdata* (short for "stream data") input is set via a fortran namelist called shr_strdata_nml. That namelist, the strdata datatype, and the methods are contained in the share source code file, models/csm_share/shr/shr_strdata_mod.F90. In general, strdata input defines an array of input streams and operations to perform on those streams. Therefore, many namelist inputs are arrays of character strings. Different variable of the same index are associated. For instance, mapalgo(1) spatial interpolation will be performed between streams(1) and the target domain.

The following namelist are available with the strdata namelist. These are defined in greater detail below.

dataMode - component specific mode
domainFile - final domain
streams - input files
vectors - paired vector field names
fillalgo - fill algorithm
fillmask - fill mask
fillread - fill mapping file to read
fillwrite - fill mapping file to write
mapalgo - spatial interpolation algorithm
mapmask - spatial interpolation mask
mapread - spatial interpolation mapping file to read
mapwrite - spatial interpolation mapping file to write
tintalgo - time interpolation algorithm
taxMode - time interpolation mode
dtlimit - delta time axis limit
io_type - IO type
num_iotasks - number of IO tasks
io_root - IO root processor
io_stride - IO processor stride
num_agg - IO number of aggregators

The strdata namelist options are as follows. The namelist group is called "shr_strdata_nml". The variable formats are character string (char), integer (int), double precision real (r8), or logical (log) or one dimensional arrays of any of those things (array of ...).

`dataMode (char)`

general method that operates on the data. this is generally implemented in the data models but is set in the strdata method for convenience. valid options are dependent on the data model and will be described elsewhere. NULL is always a valid option and means no data will be generated. default='NULL'

`domainFile (char)`

spatial gridfile associated with the strdata. grid information will be read from this file and that grid will serve as the target grid for all input data for this strdata input. default="null".

`fillalgo (array of char)`

array of fill algorithms associated with the array of streams. valid options are just copy (ie. no fill), special value, nearest neighbor, nearest neighbor in "i" direction, or nearest neighbor in "j" direction. the valid fillalgo values are ['copy','spval','nn','nnoni','nnonj']. default="nn".

`fillmask (array of char)`

plays no role is fill algorithm at the present time. default="nomask".

`fillread (array of char)`

array of filenames associated with the array of streams. specifies the weights file to read instead of computing the weights on the fly for the fill operation. if this is set, fillalgo and fillmask are ignored. default="unset".

`fillwrite (array of char)`

array of filenames associated with the array of streams. specifies the weights file to generate after weights are computed on the fly for the fill operation. this allows a user to save and reuse a set of weights later. default="unset".

`dtlimit (array of r8)`

array of delta time ratio limits placed on the time interpolation associated with the array of streams. this real value causes the model to stop if the ratio of the running maximum delta time divided by the minimum delta time is greater than the dtlimit for that stream. for instance, with daily data, the delta time should be exactly one day throughout the dataset and the computed maximum divided by minimum delta time should always be 1.0. for monthly data, the delta time should be between 28 and 31 days and the maximum ratio should be about 1.1. the running value of the delta time is computed as data is read and any wraparound or cycling is also included. this input helps trap missing data or errors in cycling. to turn off trapping, set the value to 1.0e30 or something similar. default=1.5.

`io_root (int)`

specifies the rank of the root of set of iotasks when pio IO is active. this value should be between 0 and the total number of tasks for the data model minus one, and the io_root+(num_iotasks-1)*(io_stride)+1 should be less than the total number of tasks for the data model. this is unused with io_type='netcdf'. default=0.

`io_stride (int)`

specifies the stride of the iotasks across the global mpi ranks. the io_root+(num_iotasks-1)*(io_stride)+1 should be less than the total number of tasks for the data model. this is unused with io_type='netcdf'. default=4.

`io_type (char)`

this specifies the IO implementation method. valid options are standard netcdf, serial netcdf through the pio interface, or parallel netcdf through the pio interface. the valid io_type values are ['netcdf','pio_netcdf','pio_pnetcdf']. default='pio_netcdf'.

`mapalgo (array of char)`

array of fill algorithms associated with the array of streams. valid options are copy by index, set to special value, nearest neighbor, nearest neighbor in "i" direction, nearest neighbor in "j" direction, or bilinear. the valid mapalgo values are ['copy','spval','nn','nnoni','nnonj','bilinear']. default="bilinear".

`mapmask (array of char)`

array of masking algorithms for mapping input data associated with the array of streams. valid options are map only from valid src points, map only to valid destination points, ignore all masks, map only from valid src points to valid destination points. the valid mapmask values are ['srcmask', 'dstmask', 'nomask', 'bothmask']. default="dstmask".

`mapread (array of char)`

array of filenames associated with the array of streams. specifies the weights file to read instead of computing weights on the fly for the mapping (interpolation) operation. if this is set, mapalgo and mapmask are ignored. default="unset".

`mapwrite (array of char)`

array of filenames associated with the array of streams. specifies the weights file to generate after weights are computed on the fly for the mapping (interpolation) operation. this allows a user to save and reuse a set of weights later. default="unset".

`num_agg (int)`

this is needed only with pio binary IO which is currently not supported.

`num_iotasks (int)`

specifies the number of mpi tasks that should be active with pio IO. this should be no more than the total number of mpi tasks for the data model and can be as small as one. this is inactive with io_type='netcdf'. the io_root+(num_iotasks-1)*(io_stride)+1 should be less than the total number of tasks for the data model. default=1.

`streams (array of char)`

list of stream input files. this string is actually parsed by a stream method and so the format is specified by the stream module. this string consists of a "stream_input_filename year_align year_first year_last". the stream_input_filename is a stream text input file and the format and options are described elsewhere. year_align, year_first, and year_last provide information about the time axis of the file and how to relate the input time axis to the model time axis. default="null".

`taxMode (array of char)`

array of time axis modes associated with the array of streams for handling data outside the specified stream time axis. valid options are to cycle the data based on the first, last, and align settings associated with the stream dataset, to extend the first and last valid value indefinitely, or to limit the interpolated data to fall only between the least and greatest valid value of the time array. the valid taxMode values are ['cycle','extend','limit']. default="cycle".

`tintalgo (array of char)`

array of time interpolation options associated with the array of streams. valid options are lower bound, upper bound, nearest data, linear interpolation, or interpolation based upon the cosine of the zenith angle. the valid tintalgo values are ['lower','upper','nearest','linear','coszen']. default="linear".

`vectors (array of char)`

list of paired colon delimited field names that should be treated as vectors when carrying out spatial interpolation. unlike other character arrays in this namelist, this array is completely decoupled from the list of streams. this is a list of vector

pairs that span all input streams where different fields of the vector pair could appear in different streams. for example, vectors = 'u:v','taux:tauy'. default="".

Chapter 4.

Science Modes

Overview

When the data models run, the user must specify which *science mode* they will run in. Each data model has a fixed set of fields that it must send to the coupler, but it is the choice of mode that specifies how that set of fields is to be computed. Each mode activates various assumptions about what input fields are available from the input data streams, what input fields are available from the the coupler, and how to use this input data to compute the output fields sent to the coupler.

In general, a mode might specify...

- that fields be set to a time invariant constant (so that no input data is needed)
- that fields be taken directly from a input data files (the input streams)
- that fields be computed using data read in from input files
- that fields be computed using from data received from the coupler
- some combination of the above.

If a science mode is chosen that is not consistent with the input data provided, the model may abort (perhaps with a "missing data" error message), or the model may send erroneous data to the coupler (for example, if a mode assumes an input stream has temperature in Kelvin on it, but it really has temperature in Celsius). Such an error is unlikely unless a user has edited the run scripts to specify either non-standard input data or a non-standard science mode. When editing the run scripts to use non-standard stream data or modes, users must be careful that the input data is consistent with the science mode and should verify that the data model is providing data to the coupler as expected.

Modes Common to all Models

Each data model, datm, dice, dlnd, and docn, have their own set of valid modes which are listed in the later chapters that are specific to those models.

The mode is a character string that is set in the strdata namelist variable dataMode.

Two modes are common to all data models: COPYALL and NULL.

`dataMode = "COPYALL"`

The default mode is COPYALL -- the model will assume *all* the data that must be sent to the coupler will be found in the input data streams, and that this data can be sent to the coupler, unaltered, except for spatial and temporal interpolation.

`dataMode = "NULL"`

NULL mode turns off the data model as a provider of data to the coupler. The model_present flag (eg. atm_present) will be set to false and the coupler will assume no exchange of data to or from the data model.

Chapter 5.

Data Atmosphere Model

Modes

The default science mode of the data model is the COPYALL mode. COPYALL mode will examine the fields found in all input data streams, if any input field names match the field names used internally, they are copied into the export array and passed directly to the coupler without any special user code. There are several other scientific modes supported by the model, they are listed below. The mode is selected by a character string set in the strdata namelist variable dataMode.

`dataMode = "NULL"`

NULL mode turns off the data model as a provider of data to the coupler. The atm_present flag will be set to false and the coupler will assume no exchange of data to or from the data model.

`dataMode = "COPYALL"`

COPYALL mode copies all fields directly from the input data streams Any required fields not found on an input stream will be set to zero except for aerosol deposition fields which will be set to a special value.

`dataMode = "CPLHIST"`

CPLHIST mode is the same as COPYALL mode.

`dataMode = "CORE2"`

CORE2 mode, in conjunction with with CORE Version 2 atmospheric forcing data, provides the atmosphere forcing favored by the Ocean Model Working Group when coupling an active ocean model with observed atmospheric forcing. This mode and associated data sets implement the CORE-IAF Version 2 forcing data, as developed by Large and Yeager (2008) at NCAR. See the documentation for CORE version 2 datasets at http://data1.gfdl.noaa.gov/nomads/forms/mom4/COREv2.html. Also see W.G.Large, S.G.Yeager (2008), The global climatology of an interannually varying air-sea flux data set. Clm Dyn doi 10.1007/s00382-008-0441-3.

`dataMode = "CLMNCEP"`

CLMNCEP mode, in conjunction with NCEP climatological atmosphere data, provides the atmosphere forcing favored by the Land Model Working Group when coupling an active land model with observed atmospheric forcing. This mode replicates code previously found in CLM (circa 2005), before the LMWG started using the CCSM flux coupler and data models to do active-land-only simulations.

Namelist

The data atmosphere specific namelist input is as follows. The namelist input filename is hardwired in the data model code to "datm_in". The namelist group is called "datm_nml". The variable formats are character string (char), integer (int), double precision real (r8), or logical (log) or one dimensional arrays of any of those things (array of ...).

`atm_in (char)`

sets the filename for the data atmosphere strdata namelist. this must be set. default='unset'.

`decomp (char)`

set the decomposition option for the data model. valid options are placing the global array on the root task or a simple stride-one load balanced one-dimensional decomposition. other decompositions may be added in the future. valid values are ['root','1d']. default='1d'.

`iradsw (int)`

radiation setting used to compute the next shortwave Julian date. values greater than 1 set the next radiation to the present time plus 2 timesteps every iradsw. values less than 0 turn set the next radiation to the present time plus two timesteps every -iradsw hours. if iradsw is zero, the next radiation time is the present time plus 1 timestep. default=0.

`factorFn (char)`

filename containing correction factors. use with TN460 and CORE2 modes. default='unset'.

`restfilm (char)`

restart filename for the data atmosphere model data. this is optional. if this is unset, the restart filename will be read from the rpointer.atm file. default='unset'.

`restfils (char)`

restart filename for the data atmosphere stream data. this is optional. if this is unset, the restart filename will be read from the rpointer.atm file. default='unset'.

Fields

The pre-defined internal field names in the data atmosphere model are as follows. In general, the stream input file should translate the input variable names into these names for use within the data atmosphere model.

```
  (/"z              ","u              ","v              ","tbot           ", &
    "ptem           ","shum           ","dens           ","pbot           ", &
    "pslv           ","lwdn           ","rainc          ","rainl          ", &
    "snowc          ","snowl          ","swndr          ","swvdr          ", &
    "swndf          ","swvdf          ","swnet          ","co2prog        ", &
    "co2diag        ","bcphidry       ","bcphodry       ","bcphiwet       ", &
    "ocphidry       ","ocphodry       ","ocphiwet       ","dstwet1        ", &
    "dstwet2        ","dstwet3        ","dstwet4        ","dstdry1        ", &
    "dstdry2        ","dstdry3        ","dstdry4        ",                  &
    "tref           ","qref           ","avsdr          ","anidr          ", &
    "avsdf          ","anidf          ","ts             ","to             ", &
    "snowh          ","lfrac          ","ifrac          ","ofrac          ", &
    "taux           ","tauy           ","lat            ","sen            ", &
    "lwup           ","evap           ","co2lnd         ","co2ocn         ", &
    "dms            "                                                     /)
  (/"tbot           ","wind           ","z              ","pbot           ", &
    "shum           ","tdew           ","rh             ","lwdn           ", &
    "swdn           ","swdndf         ","swdndr         ","precc          ", &
    "precl          ","precn          ","co2prog        ","co2diag        ", &
    "swup           ","prec           " /)
```

Notes

1. http://data1.gfdl.noaa.gov/nomads/forms/mom4/COREv2.html

Chapter 6.

Data Land Model

Modes

The land model is unique because it supports land data, runoff data, and now snow data (*lnd*, *rof*, and *sno*) almost as if they were three separate components, but they are in fact running in one component model through one interface. The land data is associated with the main land model. The runoff data is associated with the runoff model that runs inside the land model and is normally on a different grid than the land data. The snow data is associated with snow accumulation code (for glacial formation) inside the land model and is normally on a different grid than the land data. The snow data is new and currently unused but it's a place holder for coupling with a future land-ice component in CCSM. Each of these models, land, runoff, and snow need to have their own strdata input because they are distinct sets of fields on distinct grids with their own modes. In the data model, the land, runoff, and snow modes are treated completely independently as if there were three models.

For all three "models", land, runoff, and snow, the default science mode of the data model is the COPYALL mode. COPYALL mode will examine the fields found in all input data streams, if any input field names match the field names used internally, they are copied into the export array and passed directly to the coupler without any special user code.

Land Modes

These apply to land data associated with the main land model.

`dataMode = "NULL"`

NULL mode turns off the data model as a provider of land data to the coupler. The lnd_present flag will be set to false and the coupler will assume no exchange of land data to or from the data model.

`dataMode = "COPYALL"`

COPYALL mode copies all fields directly from the input data streams Any required fields not found on an input stream will be set to zero.

`dataMode = "CPLHIST"`

Same as COPYALL mode.

Runoff Modes

These apply to runoff data associated with the runoff model embedded in the land model.

`dataMode = "NULL"`

NULL mode turns off the data model as a provider of runoff data to the coupler. The rof_present flag will be set to false and the coupler will assume no exchange of runoff data from the data model.

`dataMode = "COPYALL"`

COPYALL mode copies all fields directly from the input data streams Any required fields not found on an input stream will be set to zero.

`dataMode = "CPLHIST"`

Same as COPYALL mode.

`dataMode = "RX1"`

Same as COPYALL mode.

Snow Modes

These apply to snow data associated with the snow accumulation model embedded in the land model. The snow data is new and currently unused but it's a place holder for coupling with a future land-ice component in CCSM.

`dataMode = "NULL"`

NULL mode turns off the data model as a provider of snow data to the coupler. The sno_present flag will be set to false and the coupler will assume no exchange of snow data to or from the data model.

`dataMode = "COPYALL"`

COPYALL mode copies all fields directly from the input data streams Any required fields not found on an input stream will be set to zero.

Namelist

The data land specific namelist input is as follows. The namelist input filename is hardwired in the data model code to "dlnd_in". The namelist group is called "dlnd_nml". The variable formats are character string (char), integer (int), double precision real (r8), or logical (log) or one dimensional arrays of any of those things (array of ...).

`lnd_in (char)`

sets the filename for the land strdata namelist. this must be set. default='unset'.

`rof_in (char)`

sets the filename for the runoff strdata namelist. this must be set. default='unset'.

`sno_in (char)`

sets the filename for the sno strdata namelist. this must be set. default='unset'.

`decomp (char)`

set the decomposition option for the data model. valid options are placing the global array on the root task or a simple stride-one load balanced one-dimensional decomposition. other decompositions may be added in the future. valid values are ['root','1d']. default='1d'.

`restfilm (char)`

restart filename for the lnd model data. this is optional. if this is unset, the restart filename will be read from the rpointer.lnd file. default='unset'.

`restfilsl (char)`

restart filename for the lnd stream data. this is optional. if this is unset, the restart filename will be read from the rpointer.lnd file. default='unset'.

`restfilsr (char)`

restart filename for the runoff stream data. this is optional. if this is unset, the restart filename will be read from the rpointer.lnd file. default='unset'.

`restfilss (char)`

restart filename for the sno stream data. this is optional. if this is unset, the restart filename will be read from the rpointer.lnd file. default='unset'.

The pre-defined internal field names in the data land model are as follows. In general, the stream input file should translate the input variable names into these names for use within the data land model.

Fields

```
   (/ "roff        ","ioff        ", &
      "t           ","tref        ","qref        ","avsdr       ","anidr       ", &
      "avsdf       ","anidf       ","snowh       ","taux        ","tauy        ", &
      "lat         ","sen         ","lwup        ","evap        ","swnet       ", &
      "lfrac       ","fv          ","ram1        ", &
      "flddst1     ","flxdst2     ","flxdst3     ","flxdst4     "  /)
```

Chapter 7.

Data Ocean Model

Modes

The default science mode of the data model is the COPYALL mode. COPYALL mode will examine the fields found in all input data streams, if any input field names match the field names used internally, they are copied into the export array and passed directly to the coupler without any special user code. There are several other scientific modes supported by the model, they are listed below. The mode is selected by a character string set in the strdata namelist variable dataMode.

`dataMode = "NULL"`

NULL mode turns off the data model as a provider of data to the coupler. The ocn_present flag will be set to false and the coupler will assume no exchange of data to or from the data model.

`dataMode = "COPYALL"`

COPYALL mode copies all fields directly from the input data streams Any required fields not found on an input stream will be set to zero.

`dataMode = "SSTDATA"`

SSTDATA mode assumes the only field in the input stream is SST. It also assumes the SST is in Celsius and must be converted to Kelvin. All other fields are set to zero except for ocean salinity, which is set to a constant reference salinity value.

`dataMode = "SOM"`

SOM ("slab ocean model") mode is a prognostic mode. This mode computes a prognostic sea surface temperature and a freeze/melt potential (surface Q-flux) used by the sea ice model. This calculation requires an external SOM forcing data file that includes ocean mixed layer depths and bottom-of-the-slab Q-fluxes. Scientifically appropriate bottom-of-the-slab Q-fluxes are normally ocean resolution dependent and are derived from the ocean model output of a fully coupled CCSM run. Note that while this mode runs out of the box, the default SOM forcing file is not scientifically appropriate and is provided for testing and development purposes only. Users must create scientifically appropriate data for their particular application. A tool is available to derive valid SOM forcing. More information on creating the SOM forcing is available at: www.ccsm.ucar.edu/models/ccsm4.0/data8/SOM.pdf[1]

Namelist

The data ocean specific namelist input is as follows. The namelist input filename is hardwired in the data model code to "docn_in". The namelist group is called "docn_nml". The variable formats are character string (char), integer (int), double precision real (r8), or logical (log) or one dimensional arrays of any of those things (array of ...).

`ocn_in (char)`

sets the filename for the data ocean strdata namelist. this must be set. default='unset'.

`decomp (char)`

set the decomposition option for the data model. valid options are placing the global array on the root task or a simple stride-one load balanced one-dimensional decomposition. other decompositions may be added in the future. valid values are ['root','1d']. default='1d'.

`restfilm (char)`

restart filename for the data ocean model data. this is optional. if this is unset, the restart filename will be read from the rpointer.ocn file. default='unset'.

`restfils (char)`

restart filename for the data ocean stream data. this is optional. if this is unset, the restart filename will be read from the rpointer.ocn file. default='unset'.

The pre-defined internal field names in the data ocean model are as follows. In general, the stream input file should translate the input variable names into these names for use within the data ocean model.

Fields

```
(/ "ifrac       ","pslv        ","duu10n      ","taux        ","tauy        ", &
   "swnet       ","lat         ","sen         ","lwup        ","lwdn        ", &
   "melth       ","salt        ","prec        ","snow        ","rain        ", &
   "evap        ","meltw       ","roff        ","ioff        ",               &
   "t           ","u           ","v           ","dhdx        ","dhdy        ", &
   "s           ","q           ","h           ","qbot        "                 /)
```

Notes

1. http://www.ccsm.ucar.edu/models/ccsm4.0/data8/SOM.pdf

Chapter 8.

Data Ice Model

Modes

The default science mode of the data model is the COPYALL mode. COPYALL mode will examine the fields found in all input data streams, if any input field names match the field names used internally, they are copied into the export array and passed directly to the coupler without any special user code. There are several other scientific modes supported by the model, they are listed below. The mode is selected by a character string set in the strdata namelist variable dataMode.

`dataMode = "NULL"`

NULL mode turns off the data model as a provider of data to the coupler. The ice_present flag will be set to false and the coupler will assume no exchange of data to or from the data model.

`dataMode = "COPYALL"`

COPYALL mode copies all fields directly from the input data streams Any required fields not found on an input stream will be set to zero.

`dataMode = "SSTDATA"`

SSTDATA is a prognostic mode. It requires data be sent to the ice model. Ice fraction (extent) data is read from an input stream, atmosphere state variables are received from the coupler, and then an atmosphere-ice surface flux is computed and sent to the coupler. It is called "SSTDATA" mode because normally the ice fraction data is found in the same data files that provide SST data to the data ocean model. They are normally found in the same file because the SST and ice fraction data are derived from the same observational data sets and are consistent with each other.

Namelist

The data ice specific namelist input is as follows. The namelist input filename is hardwired in the data model code to "dice_in". The namelist group is called "dice_nml". The variable formats are character string (char), integer (int), double precision real (r8), or logical (log) or one dimensional arrays of any of those things (array of ...).

`ice_in (char)`

sets the filename for the data ice strdata namelist. this must be set. default='unset'.

`decomp (char)`

set the decomposition option for the data model. valid options are placing the global array on the root task or a simple stride-one load balanced one-dimensional decomposition. other decompositions may be added in the future. valid values are ['root','1d']. default='1d'.

`flux_swpf (r8)`

this is the shortwave penetration factor as a fraction where 1.0 is 100% penetration. used only in SSTDATA mode. default=0.0

`flux_Qmin (r8)`

this is the minimum bound on the melt rate in kg/s/m2. used only in SSTDATA mode. default=-300.0

`flux_Qacc (log)`

this logical activates the water accumulation associated with the ocean potential to melt ice. used only in SSTDATA mode. default=false.

`flux_Qacc0 (r8)`

this is the initial water accumulation value on a startup. used only in SSTDATA mode. default=0.0

`restfilm (char)`

restart filename for the data ice model data. this is optional. if this is unset, the restart filename will be read from the rpointer.ice file. default='unset'.

`restfils (char)`

restart filename for the data ice stream data. this is optional. if this is unset, the restart filename will be read from the rpointer.ice file. default='unset'.

The pre-defined internal field names in the data ice model are as follows. In general, the stream input file should translate the input variable names into these names for use within the data ice model.

Fields

```
(/"to            ","s             ","uo            ","vo            ", &
  "dhdx          ","dhdy          ","q             ","z             ", &
  "ua            ","va            ","ptem          ","tbot          ", &
  "shum          ","dens          ","swndr         ","swvdr         ", &
  "swndf         ","swvdf         ","lwdn          ","rain          ", &
  "snow          ","t             ","tref          ","qref          ", &
  "ifrac         ","avsdr         ","anidr         ","avsdf         ", &
  "anidf         ","tauxa         ","tauya         ","lat           ", &
  "sen           ","lwup          ","evap          ","swnet         ", &
  "swpen         ","melth         ","meltw         ","salt          ", &
  "tauxo         ","tauyo         " /)
```

Chapter 9.

Data Land-Ice Model

This model does not yet exist.