CESM1.2 Data Model v8: User's Guide

Mariana Vertenstein
NCAR

Tony Craig
NCAR

Brian Kauffman
NCAR

CESM1.2 Data Model v8: User's Guide

by Mariana Vertenstein, Tony Craig, and Brian Kauffman

Table of Contents

Chapter 1.

Introduction

Overview

The CESM1.2 data models continue to perform the basic function of reading external data files, modifying that data, and then sending it to the coupler via standard CESM 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 CESM data models have parallel capability 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 user setting of the namelist variables str_fillalgo and str_fillmask[1] That operation occurs on the input data grid. The lower and upper bound data are then spatially mapped to the model grid based upon the user setting of the namelist variables str_mapalgo and str_mapmask[2]. 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 (via the shr_strdata_nml namelist variable "tintalgo"). 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. Because the present computations are always linear, changing the order of operations will not 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 the model. 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 CESM1.2 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 TODO - list these. The 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 CESM 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[3] to specify the restart filenames via namelist. If those namelist are set, the rpointer file will be ignored. The standard procedure in 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. In CESM1.2, data models communicate with the coupler with fields on only the data model model grid (in CESM1.0 the data land model communicated with the coupler on two different grids, a land grid and a runoff grid). *Although for each strdata namelist, data is interpolated to a single model grid, 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
  data model :            grid
  data model :           strdata
  strdata    :    interpa   interpb   interpc
  strdata    :    streama   streamb   streamc
  stream     :     grida     gridb     gridc
  stream     :   filea_01   fileb_01  filec_01
  stream     :     ...                 ...
  stream     :   filea_04             filec_96
```

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. 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'
  decomp = '1d'
/
```

The lnd_in specifies 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 = 'grid.nc'
  streams    = 'streama',
               'streamb',
               'streamc'
  mapalgo    = 'interpa',
               'interpb',
               'interpc'
/
```

Three stream description files are then expected to be available, streama, streamb and streamc. 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>
      <dataSource>
         GENERIC
      </dataSource>
      <fieldInfo>
         <variableNames>
            dn10  dens
            slp_  pslv
            q_10  shum
            t_10  tbot
            u_10  u
            v_10  v
         </variableNames>
         <filePath>
            /glade/proj3/cseg/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>
            /glade/proj3/cseg/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 runoff, 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.

Notes

1. ../../cesm/doc/modelnl/nl_datm.html#stream
2. ../../cesm/doc/modelnl/nl_datm.html#stream
3. ../../cesm/doc/modelnl/nl_datm.html#nonstream

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."

Stream Data

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.

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

The set of shr_strdata_nml namelist keywords are the same for all data models. As a result, any of the data model namelist documentation can be used to view a full description. For example, see stream specific namelist settings [1].

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 '
/
```

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

The stream text files for a given data model mode are automatically generated by the corresponding data model **build-namelist** with present names. As an example we refer to the following `datm_atm_in` example file (that would appear in both `$CASEROOT/CaseDocs` and `$RUNDIR`):

```
datamode   = 'CLMNCEP'
domainfile = '/glade/proj3/cseg/inputdata/share/domains/domain.lnd.fv1.9x2.5_gx1v6.090
dtlimit    = 1.5,1.5,1.5,1.5
fillalgo   = 'nn','nn','nn','nn'
fillmask   = 'nomask','nomask','nomask','nomask'
mapalgo    = 'bilinear','bilinear','bilinear','bilinear'
mapmask    = 'nomask','nomask','nomask','nomask'
streams    = "datm.streams.txt.CLM_QIAN.Solar  1895 1948 1972  ",
       "datm.streams.txt.CLM_QIAN.Precip 1895 1948 1972  ",
       "datm.streams.txt.CLM_QIAN.TPQW   1895 1948 1972  ",
       "datm.streams.txt.presaero.trans_1850-2000 1849 1849 2006"
```

```
taxmode    = 'cycle','cycle','cycle','cycle'
tintalgo   = 'coszen','nearest','linear','linear'
vectors    = 'null'
```

As is discussed in the CESM1.2 User's Guide, to change the contents of `datm_atm_in`, you can edit `$CASEROOT/user_nl_datm` to change any of the above settings EXCEPT FOR THE NAMES `datm.streams.txt.CLM_QIAN.Solar`, `datm.streams.txt.CLM_QIAN.Precip`, `datm.streams.txt.CLM_QIAN.TPQW` and `datm.streams.txt.presaero.trans_1850-2000`. Note that any namelist variable from shr_strdata_nml and datm_nml can be modified by adding the appropriate keyword/value pairs to `user_nl_datm`. As an example, the following could be the contents of `$CASEROOT/user_nl_datm`:

```
!-------------------------------------------------------------------------
! Users should ONLY USE user_nl_datm to change namelists variables
! Users should add all user specific namelist changes below in the form of
! namelist_var = new_namelist_value
! Note that any namelist variable from shr_strdata_nml and datm_nml can
! be modified below using the above syntax
! User preview_namelists to view (not modify) the output namelist in the
! directory $CASEROOT/CaseDocs
! To modify the contents of a stream txt file, first use preview_namelists
! to obtain the contents of the stream txt files in CaseDocs, and then
! place a copy of the  modified stream txt file in $CASEROOT with the string
! user_ prepended.
!-------------------------------------------------------------------------
streams    = "datm.streams.txt.CLM_QIAN.Solar  1895 1948 1900  ",
       "datm.streams.txt.CLM_QIAN.Precip 1895 1948 1900  ",
       "datm.streams.txt.CLM_QIAN.TPQW   1895 1948 1900  ",
       "datm.streams.txt.presaero.trans_1850-2000 1849 1849 2006"
```

and the contents of shr_strdata_nml (in both $CASEROOT/CaseDocs and $RUNDIR) would be

```
datamode   = 'CLMNCEP'
domainfile = '/glade/proj3/cseg/inputdata/share/domains/domain.lnd.fv1.9x2.5_gx1v6.090
dtlimit    = 1.5,1.5,1.5,1.5
fillalgo   = 'nn','nn','nn','nn'
fillmask   = 'nomask','nomask','nomask','nomask'
mapalgo    = 'bilinear','bilinear','bilinear','bilinear'
mapmask    = 'nomask','nomask','nomask','nomask'
streams    = "datm.streams.txt.CLM_QIAN.Solar  1895 1948 1900  ",
       "datm.streams.txt.CLM_QIAN.Precip 1895 1948 1900  ",
       "datm.streams.txt.CLM_QIAN.TPQW   1895 1948 1900  ",
       "datm.streams.txt.presaero.trans_1850-2000 1849 1849 2006"
taxmode    = 'cycle','cycle','cycle','cycle'
tintalgo   = 'coszen','nearest','linear','linear'
vectors    = 'null'
```

As is discussed in the User's Guide, you should use **preview_namelists** to view (not modify) the output namelist in `CaseDocs`.

Stream Description File

The *stream description file* is not a Fortran namelist, but a locally built xml-like parsing implementation. Sometimes it is called a "stream dot-text file" because it has a ".txt." in the filename. Stream description files 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.

In CESM1.2, each data model's **build-namelist** utility (e.g. models/atm/datm/bld/build-namelist) automatically generates these stream description files. The directory contents of each data model will look like the following (using DATM as an example)

```
models/atm/datm/bld/build-namelist
models/atm/datm/bld/namelist_files/namelist_definition_datm.xml
models/atm/datm/bld/namelist_files/namelist_defaults_datm.xml
```

The `namelist_definition_datm.xml` file defines all the namelist variables and associated groups. The `namelist_defaults_datm.xml` provides the out of the box settings for the target data model and target stream. **build-namelist** utilizes these two files to construct the stream files for the given compset settings. You can modify the generated stream files for your particular needs by doing the following:

1. Call **setup** OR **preview_namelists**.
2. Copy the relevant description file from `$CASEROOT/CaseDocs` to `$CASEROOT` and pre-pend a "user_" string to the filename. Change the permission of the file to write. For example, assuming you are in **$CASEROOT**

   ```
   cp CaseDocs/datm.streams.txt.CLM_QIAN.Solar  user_datm.streams.txt.CLM_QIAN.Sola
   chmod u+w user_datm.streams.txt.CLM_QIAN.Solar
   ```

3.
   - Edit `user_datm.streams.txt.CLM_QIAN.Solar` with your desired changes.
   - *Be sure not to put any tab characters in the file: use spaces instead.*
   - In contrast to other user_nl_xxx files, be sure to set all relevant data model settings in the xml files, issue the **preview_namelist** command and THEN edit the `user_datm.streams.txt.CLM_QIAN.Solar` file.
   - **Once you have created a user_xxx.streams.txt.* file, further modifications to the relevant data model settings in the xml files will be ignored.**
   - If you later realize that you need to change some settings in an xml file, you should remove the user_xxx.streams.txt.* file(s), make the modifications in the xml file, rerun **preview_namelists,** and then reintroduce your modifications into a new user_xxx.streams.txt.* stream file(s).

4. Call **preview_namelists**
5. Verify that your changes do indeed appear in the resultant stream description file appear in `CaseDocs/datm.streams.txt.CLM_QIAN.Solar`. These changes will also appear in `$RUNDIR/datm.streams.txt.CLM_QIAN.Solar`.

The data elements found in the stream description file are:

`dataSource`

A comment about the source of the data -- always set to GENERIC in CESM1.2 and not used by the model. This is there only for backwards compatibility.

`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 namelists, tinterp_algo and taxMode [2].

`offset`

The 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.

The data models advance in time discretely. At a given time, they read/derive fields from input files. Those input files have data on a discrete time axis as well. Each data point in the input files are associated with a discrete time (as opposed to a time interval). Depending whether you pick lower, upper, nearest, linear, or coszen; the data in the input file will be "interpolated" to the time in the model.

The offset shifts the time axis of the input data the given number of seconds. so if the input data is at 0, 3600, 7200, 10800 seconds (hourly) and you set an offset of 1800, then the input data will be set at times 1800, 5400, 9000, and 12600. so a model at time 3600 using linear interpolation would have data at "n=2" with offset of 0 will have data at "n=(2+3)/2" with an offset of 1800. n=2 is the 2nd data in the time list 0, 3600, 7200, 10800 in this example. n=(2+3)/2 is the average of the 2nd and 3rd data in the time list 0, 3600, 7200, 10800. offset can be positive or negative.

`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>
<dataSource>
GENERIC
</dataSource>
<domainInfo>
<variableNames>
time   time
lon    lon
lat    lat
area   area
mask   mask
</variableNames>
<filePath>
/glade/proj3/cseg/inputdata/atm/datm7/NYF
</filePath>
<fileNames>
nyf.ncep.T62.050923.nc
</fileNames>
</domainInfo>
<fieldInfo>
<variableNames>
dn10  dens
slp_  pslv
q10   shnum
t_10  tbot
u_10  u
v_10  v
</variableNames>
<filePath>
/glade/proj3/cseg/inputdata/atm/datm7/NYF
</filePath>
<offset>
0
</offset>
<fileNames>
nyf.ncep.T62.050923.nc
</fileNames>
```

```
</fieldInfo>
</stream>
```

Notes

1. ../../cesm/doc/modelnl/nl_datm.html#stream
2. ../../cesm/doc/modelnl/nl_datm.html#stream

Chapter 3.

Data Model Science Modes

When a given data models run, the user must specify which *science mode* it 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.

The data model mode is a character string that is set in the namelist variable "datamode" in the namelist group "shr_strdata_nml". Although each data model, datm[1], dlnd[2], drof[3], docn[4], and dicn[5], has its own set of valid datamode values, 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.

Data Atmosphere Model

Namelists

DATM namelists can be separated into two groups, stream-independent namelist variables[6] that are specific to the DATM model and stream-specific namelist variables[7] that are contained in share code and whose names are common to all the data models.

For stream-independent input, the namelist input filename is hardwired in the data model code to "datm_in" (or datm_in_NNNN for multiple instances) and 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 ...).

For stream-dependent input, the namelist input file is datm_atm_in (or datm_atm_in_NNNN for multiple instances) and the namelist group is "shr_strdata_nml". One of the variables in shr_strdata_nml is the datamode value. The mode is selected by a character string set in the strdata namelist variable dataMode. Each data model has a unique set of datamode values that it supports. Those for DATM are listed in detail in the datamode[8] definition.

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            ", &
  "snowhl        ","lfrac         ","ifrac         ","ofrac         ", &
  "taux          ","tauy          ","lat           ","sen           ", &
  "lwup          ","evap          ","co2lnd        ","co2ocn        ", &
  "dms           "                                                    /)
```

Data Land Model

Namelists

The land model is unique because it supports land data and snow data (*lnd* and *sno*) almost as if they were two separate components, but they are in fact running in one component model through one interface. The lnd (land) data consist of fields sent to the atmosphere. This set of data is used when running dlnd with an active atmosphere. In general this is not a mode that is used or supported in CESM1.1. The sno (snow) data consist of fields sent to the glacier model. This set of data is used when running dlnd with an active glacier model (TG compsets). Both sets of data are assumed to be on the same grid.

DLND namelists can be separated into two groups, stream-independent namelist variables[9] that are specific to the DLND model and stream-specific namelist variables[10] that are contained in share code and whose names are common to all the data models.

For stream-independent input, the namelist input filename is hardwired in the data model code to "dlnd_in" (or dlnd_in_NNNN for multiple instances) and 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 ...).

For stream-dependent input, the namelist input file is `dlnd_lnd_in` and `dlnd_sno_in` (or `dlnd_lnd_in_NNNN` and `dlnd_sno_in_NNNN` for NNNN multiple instances) and the namelist group is "shr_strdata_nml". One of the variables in shr_strdata_nml is the datamode value. The mode is selected by a character string set in the strdata namelist variable dataMode. Each data model has a unique set of datamode values that it supports. Those for DLND are listed in detail in the datamode[11] definition.

If you want to change the namelist settings in `dlnd_lnd_in` or `dlnd_in` you should edit the file `user_nl_dlnd`. If you want to change the namelist settings in `dsno_lnd_in` or `dsno_in` you should edit the file `user_nl_dsno`.

Fields

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.

```
(/ "t           ","tref        ","qref        ","avsdr       ","anidr       ", &
   "avsdf       ","anidf       ","snowh       ","taux        ","tauy        ", &
   "lat         ","sen         ","lwup        ","evap        ","swnet       ", &
   "lfrac       ","fv          ","ram1        ",                              &
   "flddst1     ","flxdst2     ","flxdst3     ","flxdst4     ",               &
   "tsrf01      ","topo01      ","tsrf02      ","topo02      ","tsrf03      ", &
   "topo03      ","tsrf04      ","topo04      ","tsrf05      ","topo05      ", &
   "tsrf06      ","topo06      ","tsrf07      ","topo07      ","tsrf08      ", &
   "topo08      ","tsrf09      ","topo09      ","tsrf10      ","topo10      ", &
   "qice01      ","qice02      ","qice03      ","qice04      ","qice05      ", &
   "qice06      ","qice07      ","qice08      ","qice09      ","qice10      "  /)
```

Data River Runoff Model

Namelists

The data river runoff model is new and is effectively the runoff part of the dlnd model in CESM1.0 that has been made its own top level component.

DROF namelists can be separated into two groups, stream-independent namelist variables[12] that are specific to the DROF model and stream-specific namelist variables[13] that are contained in share code and whose names are common to all the data models.

For stream-independent input, the namelist input filename is hardwired in the data model code to "drof_in" (or drof_in_NNNN for multiple instances) and the namelist group is called "drof_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 ...).

For stream-dependent input, the namelist input file is "drof_lnd_in" (or drof_rof_in_NNNN for NNNN multiple instances) and the namelist group is "shr_strdata_nml". One of the variables in shr_strdata_nml is the datamode value. The mode is selected by a character string set in the strdata namelist variable dataMode. Each data model has a unique set of datamode values that it supports. Those for DROF are listed in detail in the datamode[14] definition.

Fields

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

```
(/ "roff        ","ioff        "/)
```

Data Ocean Model

Namelists

DOCN namelists can be separated into two groups, stream-independent namelist variables[15] that are specific to the DATM model and stream-specific namelist variables[16] that are contained in share code and whose names are common to all the data models.

For stream-independent input, the namelist input filename is hardwired in the data model code to "docn_in" (or docn_in_NNNN for multiple instances) and 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 ...).

For stream-dependent input, the namelist input file is docn_ocn_in (or docn_ocn_in_NNNN for multiple instances) and the namelist group is "shr_strdata_nml". One of the variables in shr_strdata_nml is the datamode value. The mode is selected by a character string set in the strdata namelist variable dataMode. Each data model has a unique set of datamode values that it supports. Those for DOCN are listed in detail in the datamode[17] definition. As part of the stream independent namelist input, DOCN supports two science modes, "SSTDATA" and "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 this mode no longer runs out of the box, the default testing 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 or use one of the standard SOM forcing files from the CESM control runs. Some of these are available in the inputdata repository[18]. The user then edits the DOCN_SOM_FILENAME variable in env_run.xml to point to the appropriate SOM forcing dataset. A tool is available to derive valid SOM forcing. More information on creating the SOM forcing[19] is also available.

Fields

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.

```
(/ "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        "                 /)
```

Data Ice Model

Namelists

DICE namelists can be separated into two groups, stream-independent namelist variables[20] that are specific to the DATM model and stream-specific namelist variables[21] that are contained in share code and whose names are common to all the data models.

For stream-independent input, the namelist input filename is hardwired in the data model code to "dice_in" (or dice_in_NNNN for multiple instances) and the namelist group is called "dice_nml".

Its important to point out that the only currently supported datamode that is not "NULL" or "COPYALL" is "SSTDATA", which is a prognostic mode and therefore 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.

Fields

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 ocean model.

```
(/"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          " /)
```

Data Land-Ice Model

This model does not yet exist.

Notes

1. ../../cesm/doc/modelnl/nl_datm.html#stream
2. ../../cesm/doc/modelnl/nl_dlnd.html#stream
3. ../../cesm/doc/modelnl/nl_drof.html#stream
4. ../../cesm/doc/modelnl/nl_docn.html#stream
5. ../../cesm/doc/modelnl/nl_dice.html#stream
6. ../../cesm/doc/modelnl/nl_datm.html#nonstream
7. ../../cesm/doc/modelnl/nl_datm.html#stream

8. ../../cesm/doc/modelnl/nl_datm.html#stream
9. ../../cesm/doc/modelnl/nl_dlnd.html#nonstream
10. ../../cesm/doc/modelnl/nl_dlnd.html#stream
11. ../../cesm/doc/modelnl/nl_dlnd.html#stream
12. ../../cesm/doc/modelnl/nl_drof.html#nonstream
13. ../../cesm/doc/modelnl/nl_drof.html#stream
14. ../../cesm/doc/modelnl/nl_drof.html#stream
15. ../../cesm/doc/modelnl/nl_docn.html#nonstream
16. ../../cesm/doc/modelnl/nl_docn.html#stream
17. ../../cesm/doc/modelnl/nl_docn.html#stream
18. https://svn-ccsm-inputdata.cgd.ucar.edu/trunk/inputdata/ocn/docn7/SOM/
19. ./SOM.pdf
20. ../../cesm/doc/modelnl/nl_dice.html#nonstream
21. ../../cesm/doc/modelnl/nl_dice.html#stream