
Climate Data Management System

Version 3.3

Robert Drach, Paul Dubois, Dean Williams

**Program for Climate Model Diagnosis and
Intercomparison**

Lawrence Livermore National Laboratory

September 2002

UCRL-JC-134897

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Table of Contents

CHAPTER 1 Introduction 9

Overview	9
Variables	9
File I/O	10
Domains and Axes	11
Attributes	11
Masked values	12
File Variables	13
Dataset Variables	15
Grids and Regridding	16
Time types	17
Plotting data	18
Databases	19

CHAPTER 2 CDMS Python Application Programming Interface 21

Overview	21
<i>Python types used in CDMS 22</i>	
A first example	23
cdms module	25
<i>cdms module functions 25</i>	
<i>Class Tags 32</i>	
CdmsObj	32
<i>Attributes common to all CDMS objects 32</i>	
<i>Getting and setting attributes 33</i>	
Axis	33
<i>Axis Internal Attributes 33</i>	
<i>partition attribute 34</i>	
<i>Axis Constructors 35</i>	
<i>Axis Methods 36</i>	

<i>Axis Slice Operators</i>	43
CdmsFile	44
<i>CdmsFile Internal Attributes</i>	44
<i>CdmsFile Constructors</i>	44
<i>CdmsFile Methods</i>	45
<i>CDMS Datatypes</i>	50
Database	50
<i>Overview</i>	51
<i>Database Internal Attributes</i>	52
<i>Database Constructors</i>	53
<i>Database Methods</i>	53
<i>Searching a database</i>	56
<i>SearchResult Methods</i>	58
<i>ResultEntry Attributes</i>	59
<i>ResultEntry Methods</i>	59
<i>Accessing data</i>	59
<i>Examples of database searches</i>	60
Dataset	61
<i>Dataset Internal Attributes</i>	61
<i>Dataset Constructors</i>	62
<i>Open Modes</i>	63
<i>Template Specifiers</i>	63
<i>Dataset Methods</i>	64
MV module	66
<i>Variable Constructors in module MV</i>	68
<i>MV functions</i>	69
RectGrid	72
<i>RectGrid Internal Attributes</i>	73
<i>RectGrid Constructors</i>	73
<i>RectGrid Methods</i>	74
Variable	80
<i>cu interface support</i>	81
<i>Variable Internal Attributes</i>	81
<i>Variable Constructors</i>	82
<i>Variable Methods</i>	85
<i>Variable Slice Operators</i>	94
<i>Index and Coordinate Intervals</i>	94
<i>Selectors</i>	95
<i>Selector keywords</i>	96
<i>Selector examples</i>	98

Examples **99**

CHAPTER 3 *cdtime Module* **105**

Time types **105**

Calendars **106**

Time Constructors **106**

Time Constructors 107

Relative Time **108**

Relative Time Members 108

Component Time **109**

Component Time Members 109

Time Methods **109**

Time Methods 110

CHAPTER 4 *Regridding Data* **113**

Overview **113**

Horizontal regridded 113

Pressure-level regridded 115

Cross-section regridded 115

regrid module **116**

Regridded Constructor 116

regridded functions **117**

Regridded function 120

Examples **121**

CHAPTER 5 *Plotting CDMS data in Python* **127**

Overview **127**

Examples **127**

Example: plotting a horizontal grid 127

Example: using plot keywords. 128

Example: plotting a time-latitude slice 129

Example: plotting subsetted data 129

plot method **130**
plot keywords 131

CHAPTER 6 *Climate Data Markup Language (CDML) 135*

Introduction **135**
Elements **136**
CDML Tags 136
Special Characters **137**
Special Character Encodings 137
Identifiers **138**
CF Metadata Standard **138**
CDML Syntax **138**
Dataset Element 139
Dataset Attributes 139
Axis Element 141
Axis Attributes 141
Grid Element 143
RectGrid Attributes 144
Variable Element 144
Variable Attributes 145
Attribute Element 146
A Sample CDML Document **147**

CHAPTER 7 *CDMS Utilities 149*

cdscan: Importing datasets into CDMS **149**
Overview 149
cdscan Syntax 150
cdscan command options 151
Examples 153
File Formats 153
Name Aliasing 154
Generating Metadata for a File 154

APPENDIX A *CDMS Classes* **155**

APPENDIX B *Version Notes* **157**

Version 3.0 Overview **157**

V3.0 Details **158**

AbstractVariable **158**

AbstractAxis **158**

AbstractDatabase **159**

Dataset **159**

cdms module **159**

CdmsFile **159**

CDMSError **159**

AbstractRectGrid **159**

InternalAttributes **159**

TransientVariable **159**

MV **160**

APPENDIX C *cu Module* **161**

Slab **161**

Slab Methods 162

cuDataset **163**

cuDataset Methods 163

1.1 Overview

The Climate Data Management System is an object-oriented data management system, specialized for organizing multidimensional, gridded data used in climate analysis and simulation.

CDMS is implemented as part of the Climate Data Analysis Tool (CDAT), which uses the Python language. The examples in this chapter assume some familiarity with the language and the Python Numeric module (<http://numpy.sf.net>). A number of excellent tutorials on Python are available in books or on the Internet. For example, see <http://python.org>.

1.2 Variables

The basic unit of computation in CDMS is the *variable*. A variable is essentially a multidimensional data array, augmented with a *domain* and a set of *attributes*. As a data array, a variable can be sliced to obtain a portion of the data, and can be used in arithmetic computations. For example, if u and v are variables representing the eastward and northward components of

wind speed, respectively, and both variables are functions of time, latitude, and longitude, then the velocity for time 0 (first index) can be calculated as

```
>>> from cdms import MV
>>> vel = MV.sqrt(u[0]**2 + v[0]**2)
```

This illustrates several points:

- Square brackets represent the slice operator. Indexing starts at 0, so `u[0]` selects from variable `u` for the first timepoint. The result of this slice operation is another variable.
- Variables can be used in computation. `**` is the Python exponentiation operator.
- Arithmetic functions are defined in the `cdms.MV` module.

1.3 File I/O

A variable can be obtained from a file or collection of files, or can be generated as the result of a computation. Files can be in any of the self-describing formats netCDF, HDF, GrADS/GRIB (GRIB with a GrADS control file), or PCMDI DRS. (Depending on your local installation, HDF and DRS may or may not be enabled.) For instance, to read data from file `sample.nc` into variable `u`:

```
>>> import cdms
>>> f = cdms.open('sample.nc')
>>> u = f('u')
```

Data can be read by index or by world coordinate values. The following reads the `n`-th timepoint of `u` (the syntax `slice(i,j)` refers to indices `k` such that `i <= k < j`):

```
>>> u0 = f('u',time=slice(n,n+1))
```

and this reads `u` at time 366.0:

```
>>> u1 = f('u',time=366.)
```

A variable can be written to a file with the `write` function:

```
>>> g = cdms.open('sample2.nc','w')
```

```
>>> g.write(u)
<Variable: u, file: sample2.nc, shape: (1, 16, 32)>
>>> g.close()
```

1.4 Domains and Axes

The spatial and temporal information associated with a variable is represented by the variable *domain*, an ordered tuple of axes and/or grids. In the above example, the domain of the variable `u` is the tuple (time, latitude, longitude). This indicates the order of the dimensions, with the slowest-varying dimension listed first (time).

Each element of the tuple is an *axis*. An axis is like a 1-D variable, in that it can be sliced, and has attributes. A number of functions are available to access axis information. For example, to see the list of time values associated with `u`:

```
>>> t = u.getTime()
>>> print t[:]
[  0., 366., 731.,]
```

1.5 Attributes

As mentioned above, variables can have associated *attributes*. In fact, nearly all CDMS objects can have associated attributes, which are accessed using the Python dot notation:

```
>>> u.units='m/s'
>>> print u.units
m/s
```

Attribute values can be strings, scalars, or 1-D Numeric arrays.

When a variable is written to a file, not all the attributes are written. Some attributes, called *internal* attributes, are used for bookkeeping, and are not intended to be part of the external file representation of the variable. In contrast, *external* attributes are written to an output file along with the variable.

By default, when an attribute is set, it is treated as external. To see the list of external attribute names:

```
>>> print u.attributes.keys()
['datatype', 'name', 'missing_value', 'units']
```

The Python **dir** command lists the internal attribute names:

```
>>> dir(u)
['_MaskedArray__data', '_MaskedArray__fill_value', ..., 'id',
 'parent']
```

In general internal attributes should not be modified directly. One exception is the **id** attribute, the name of the variable. It is used in plotting and I/O, and can be set directly.

1.6 Masked values

Variables can have an optional mask which represents a portion of data that is missing. If present, the mask of a variable is an array of ones and zeros, of the same shape as the data array. A mask value of one indicates that the corresponding data array element is missing or invalid.

Arithmetic operations in CDMS take missing data into account. The same is true of the functions defined in the **cdms.MV** module. For example:

```
>>> a = MV.array([1,2,3]) # Create array a, with no mask
>>> b = MV.array([4,5,6]) # Same for b
>>> a+b
variable_13
array([5,7,9,])
>>> a[1]=MV.masked      # Mask the second value of a
>>> a.mask()           # View the mask
[0,1,0,]
>>> a+b                # The sum is masked also
variable_14
array(
    data = [5,0,9,],
    mask = [0,1,0,],
    fill_value=[0,]
)
```

When data is read from a file, the result variable is masked if the file variable has a **missing_value** attribute. The mask is set to one for those elements equal to the missing value, zero elsewhere. If no such attribute is present in the file, the result variable is not masked.

When a variable with masked values is written to a file, data values with a corresponding mask value of one are set to the value of the variable's **missing_value** attribute. The data and **missing_value** attribute are then written to the file.

Masking is covered in Section 2.9. Also see the documentation on the Python **Numeric** and **MA** modules, on which **cdms.MV** is based, at <http://numpy.sourceforge.net> .

1.7 File Variables

A variable can be obtained either from a file, a collection of files, or as the result of computation. Correspondingly there are three types of variables in CDMS:

- A *file variable* is a variable associated with a single data file. Setting or referencing a file variable generates I/O operations.
- A *dataset variable* is a variable associated with a collection of files. Reference to a dataset variable reads data, possibly from multiple files. At present writing, dataset variables are read-only.
- A transient variable is not associated with a file or dataset. The examples to this point illustrate this type of variable. Transient variables result from a computation or I/O operation.

A typical use of file variables is to inquire information about variables in a file without actually reading the data for the variables. A file variable is obtained by applying the slice operator `[]` to a file, with the name of the variable, or with the **getVariable** function. Note that obtaining a file variable does not actually read the data array:

```
>>> f = cdms.open('sample.nc','r+')
>>> u = f.getVariable('u')      # or u=f['u']
```

```
>>> u.shape
(3, 16, 32)
```

File variables are also useful for fine-grained I/O. They behave like transient variables, but operations on them also affect the associated file. Specifically:

- slicing a file variable reads data,
- setting a slice writes data,
- referencing an attribute reads the attribute,
- setting an attribute writes the attribute,
- and calling a file variable like a function reads data associated with the variable:

```
>>> f = cdms.open('sample.nc','r+') # Open read/write
>>> uvar = f['u'] # Note square brackets
>>> uvar.shape
(3, 16, 32)
>>> u0 = uvar[0] # Reads data from sample.nc
>>> u0.shape
(16, 32)
>>> uvar[1]=u0 # Writes data to sample.nc
>>> uvar.units # Reads the attribute
'm/s'
>>> uvar.units='meters/second' # Writes the attribute
>>> u24 = uvar(time=24.0) # Reads data
>>> f.close() # Save changes to sample.nc (I/O may be buffered)
```

In an interactive application, the type of variable can be determined simply by printing the variable:

```
>>> rlsf # Transient variable
rls
array(
  array (4,48,96) , type = f, has 18432 elements)
>>> rlsg # Dataset variable
<Variable: rls, dataset: mri_perturb, shape: (4, 46, 72)>
>>> prc # File variable
<Variable: prc, file: testnc.nc, shape: (16, 32, 64)>
```

Note that the data values themselves are not printed. For transient variables, the data is printed only if the size of the array is less than the *print limit*. This value can be set with the function **MV.set_print_limit** to force the data to be printed:

```
>>> smallvar.size() # Number of elements
20
>>> MV.get_print_limit() # Current limit
300
>>> smallvar
small variable
array(
  [[ 0., 1., 2., 3.,]
   [ 4., 5., 6., 7.,]
   [ 8., 9., 10., 11.,]
   [ 12., 13., 14., 15.,]
   [ 16., 17., 18., 19.,]]

>>> largevar.size()
400
>>> largevar
large variable
array(
  array (20,20) , type = d, has 400 elements)

>>> MV.set_print_limit(500) # Reset the print limit
>>> largevar
large variable
array(
  [[ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.,
    10., 11., 12., 13., 14., 15., 16., 17., 18., 19.,]
   ... ])
```

The datatype of the variable is determined with the **typecode** function:

```
>>> x.typecode()
'd'
```

1.8 Dataset Variables

The third type of variable, a *dataset variable*, is associated with a *dataset*, a collection of files that is treated as a single file. A dataset is created with the **cdscan** utility. This generates an ASCII metafile that describes how the files are organized, and what metadata is contained in the files. In a climate simulation application, a dataset usually represents the data generated by one run of a general circulation or coupled ocean-atmosphere model.

For example, suppose data for variables `u` and `v` are stored in six files: `u_2000.nc`, `u_2001.nc`, `u_2002.nc`, `v_2000.nc`, `v_2001.nc`, and `v_2002.nc`. A metafile can be generated with the command:

```
% cdscan -x cdsample.xml [uv]*.nc
```

The metafile `cdsample.xml` is then used like an ordinary data file:

```
>>> f = cdms.open('cdsample.xml')
>>> u = f('u')
>>> u.shape
(3, 16, 32)
```

1.9 Grids and Regridding

Latitude-longitude *grids* are used for regridding variables. A grid encapsulates:

- latitude, longitude coordinates
- grid cell boundaries
- area weights
- data ordering

For example, to regrid variable `u` to a 96x192 Gaussian grid:

```
>>> u = f('u')
>>> u.shape
(3, 16, 32)
>>> t63_grid = cdms.createGaussianGrid(96)
>>> u63 = u.regrid(t63_grid)
>>> u63.shape
(3, 96, 192)
```

To regrid a variable `uold` to the same grid as variable `vnew`:

```
>>> uold.shape
(3, 16, 32)
>>> vnew.shape
(3, 96, 192)
>>> t63_grid = vnew.getGrid() # Obtain the grid for vnew
>>> u63 = u.regrid(t63_grid)
>>> u63.shape
```

(3, 96, 192)

Regridding is discussed in Chapter 4.

1.10 Time types

CDMS provides extensive support for time values in the **cdtime** module. **cdtime** also defines a set of *calendars*, specifying the number of days in a given month.

Two time types are available: *relative time* and *component time*. Relative time is time relative to a fixed base time. It consists of:

- a **units** string, of the form “**units since basetime**”, and
- a floating-point **value**

For example, the time “28.0 days since 1996-1-1” has **value=28.0**, and **units=“days since 1996-1-1”**. To create a relative time type:

```
>>> import cdtime
>>> rt = cdtime.reltime(28.0, "days since 1996-1-1")
>>> rt
28.00 days since 1996-1-1
>>> rt.value
28.0
>>> rt.units
'days since 1996-1-1'
```

A component time consists of the integer fields **year**, **month**, **day**, **hour**, **minute**, and the floating-point field **second**. For example:

```
>>> ct = cdtime.comptime(1996,2,28,12,10,30)
>>> ct
1996-2-28 12:10:30.0
>>> ct.year
1996
>>> ct.month
2
```

The conversion functions **tocomp** and **torel** convert between the two representations. For instance, suppose that the time axis of a variable is repre-

sented in units “days since 1979”. To find the coordinate value corresponding to January 1, 1990:

```
>>> ct = cdtime.comptime(1990,1)
>>> rt = ct.torel("days since 1979")
>>> rt.value
4018.0
```

Time values can be used to specify intervals of time to read. The syntax `time=(c1,c2)` specifies that data should be read for times `t` such that `c1<=t<=c2`:

```
>>> c1 = cdtime.comptime(1990,1)
>>> c2 = cdtime.comptime(1991,1)
>>> ua = f['ua']
>>> ua.shape
(480, 17, 73, 144)
>>> x = ua.subRegion(time=(c1,c2))
>>> x.shape
(12, 17, 73, 144)
```

or string representations can be used:

```
>>> x = ua.subRegion(time=('1990-1','1991-1'))
```

Time types are described in Chapter 3.

1.11 Plotting data

Data read via the CDMS Python interface can be plotted using the `vcs` module. This module, part of the Climate Data Analysis Tool (CDAT) is documented in the VCS reference manual. The `vcs` module provides access to the functionality of the VCS visualization program.

To generate a plot:

- Initialize a canvas with the `vcs init` routine.
- Plot the data using the canvas `plot` routine.

For example:

```
>>> import cdms, vcs
>>> f = cdms.open('sample.nc')
```

```
>>> f['time'][:] # Print the time coordinates
[ 0.,  6., 12., 18., 24., 30., 36., 42., 48., 54., 60., 66., 72.,
 78., 84., 90.,]
>>> precip = f('prc', time=24.0) # Read precip data
>>> precip.shape
(1, 32, 64)
>>> w = vcs.init() # Initialize a canvas
'Template' is currently set to P_default.
Graphics method 'Boxfill' is currently set to Gfb_default.
>>> w.plot(precip) # Generate a plot
(generates a boxfill plot)
```

By default a boxfill plot of the lat-lon slice is produced. Since variable `precip` includes information on time, latitude, and longitude, the continental outlines and time information are also plotted.

The **plot** routine has a number of options for producing different types of plots, such as isofill and x-y plots. See Chapter 5 for details.

1.12 Databases

Datasets can be aggregated together into hierarchical collections, called *databases*. In typical usage, a program:

- connects to a database
- searches for data
- opens a dataset
- accesses data

Databases add the ability to search for data and metadata in a distributed computing environment. At present CDMS supports one particular type of database, based on the Lightweight Directory Access Protocol (LDAP).

Here is an example of accessing data via a database:

```
>>> db = cdms.connect() # Connect to the default database.
>>> f = db.open('ncep_reanalysis_mo') # Open a dataset.
>>> f.variables.keys() # List the variables in the dataset.
['ua', 'evs', 'cvvta', 'tauv', 'wap', 'cvwhusa', 'rss', 'rls', ...
'prc', 'ts', 'va']
```

Databases are discussed further in Section 2.7.

CDMS Python Application Programming Interface

2.1 Overview

This chapter describes the CDMS Python application programming interface (API). Python is a popular public-domain, object-oriented language. Its features include support for object-oriented development, a rich set of programming constructs, and an extensible architecture. CDMS itself is implemented in a mixture of C and Python. In this chapter the assumption is made that the reader is familiar with the basic features of the Python language.

Python supports the notion of a **module**, which groups together associated classes and methods. The **import** command makes the module accessible to an application. This chapter documents the **cdms** module.

The chapter sections correspond to the CDMS classes. Each section contains tables describing the class internal (non-persistent) attributes, constructors (functions for creating an object), and class methods (functions). A method can return an instance of a CDMS class, or one of the Python types:

Table 2.1 Python types used in CDMS

Type	Description
Array	Numeric or masked multidimensional data array. All elements of the array are of the same type. Defined in the Numeric and MA modules.
Comptime	Absolute time value, a time with representation (year, month, day, hour, minute, second). Defined in the cdtime module. cf. reltime
Dictionary	An unordered collection of objects, indexed by key. All dictionaries in CDMS are indexed by strings, e.g.: <code>axes['time']</code>
Float	Floating-point value.
Integer	Integer value.
List	An ordered sequence of objects, which need not be of the same type. New members can be inserted or appended. Lists are denoted with square brackets, e.g., <code>[1, 2.0, 'x', 'y']</code>
None	No value returned.
Reltime	Relative time value, a time with representation (value, “units since basetime”). Defined in the cdtime module. cf. comptime
Tuple	An ordered sequence of objects, which need not be of the same type. Unlike lists, tuples elements cannot be inserted or appended. Tuples are denoted with parentheses, e.g., <code>(1, 2.0, 'x', 'y')</code>

2.2 A first example

The following Python script reads January and July monthly temperature data from an input dataset, averages over time, and writes the results to an output file. The input temperature data is ordered (time, latitude, longitude).

```
1 #!/usr/bin/env python
2 import cdms
3 from cdms import MV
4 jones = cdms.open('/pcmdi/cdms/obs/jones_mo.nc')
5 tasvar = jones['tas']
6 jans = tasvar[0::12]
7 july = tasvar[6::12]
8 janavg = MV.average(jans)
9 janavg.id = "tas_jan"
10 janavg.long_name = "mean January surface temperature"
11 julyavg = MV.average(july)
12 julyavg.id = "tas_jul"
13 julyavg.long_name = "mean July surface temperature"
14 out = cdms.open('janjuly.nc', 'w')
15 out.write(janavg)
16 out.write(julyavg)
17 out.comment = "Average January/July from Jones dataset"
18 jones.close()
19 out.close()
```

Line	Notes
2,3	Makes the CDMS and MV modules available. MV defines arithmetic functions.
4	Opens a netCDF file read-only. The result <code>jones</code> is a dataset object.
5	Gets the surface air temperature variable. <code>'tas'</code> is the name of the variable in the input dataset. This does not actually read the data.

Line	Notes
6	<p>Read all January monthly mean data into a variable <code>jans</code>. Variables can be sliced like arrays. The slice operator <code>[0::12]</code> means ‘take every 12th slice from dimension 0, starting at index 0 and ending at the last index.’ If the stride <code>12</code> were omitted, it would default to 1.</p> <p>Note that the variable is actually 3-dimensional. Since no slice is specified for the second or third dimensions, all values of those dimensions are retrieved. The slice operation could also have been written <code>[0::12, :, :]</code>.</p> <p>Also note that the same script works for multi-file datasets. CDMS opens the needed data files, extracts the appropriate slices, and concatenates them into the result array.</p>
7	<p>Reads all July data into a masked array <code>july</code>.</p>
8	<p>Calculate the average January value for each grid zone. Any missing data is handled automatically.</p>
9,10	<p>Set the variable <code>id</code> and <code>long_name</code> attributes. The <code>id</code> is used as the name of the variable when plotted or written to a file.</p>
14	<p>Create a new netCDF output file named ‘<code>janjuly.nc</code>’ to hold the results.</p>
15	<p>Write the January average values to the output file. The variable will have id “<code>tas_jan</code>” in the file.</p> <p><code>write</code> is a utility function which creates the variable in the file, then writes data to the variable. A more general method of data output is first to create a variable, then set a slice of the variable.</p> <p>Note that <code>janavg</code> and <code>julavg</code> have the same latitude and longitude information as <code>tasvar</code>. It is carried along with the computations.</p>
17	<p>Set the global attribute ‘<code>comment</code>’.</p>
18	<p>Close the output file.</p>

2.3 *cdms module*

The **cdms** module is the Python interface to CDMS. The objects and methods in this chapter are made accessible with the command:

```
import cdms
```

The functions described in this section are not associated with a class. Rather, they are called as module functions, e.g.,

```
file = cdms.open('sample.nc')
```

Table 2.2 cdms module functions

Type	Definition
Variable	<p>asVariable(s)</p> <p>Transform <i>s</i> into a transient variable.</p> <p><i>s</i> is a masked array, Numeric array, or Variable. If <i>s</i> is already a transient variable, <i>s</i> is returned.</p> <p>See also: isVariable.</p>
Axis	<p>createAxis(data, bounds=None)</p> <p>Create an Axis, which is not associated with a file or dataset. This is useful for creating a grid which is not contained in a file or dataset.</p> <p><i>data</i> is a one-dimensional, monotonic Numeric array.</p> <p><i>bounds</i> is an array of shape (len(<i>data</i>),2), such that for all <i>i</i>, <i>data</i>[<i>i</i>] is in the range [bounds[<i>i</i>,0],bounds[<i>i</i>,1]]. If <i>bounds</i> is not specified, the default boundaries are generated at the mid-points between the consecutive data values, provided that the autobounds mode is 'on' (the default). See setAutoBounds.</p> <p>Also see: CdmsFile.createAxis</p>

Table 2.2 `cdms` module functions

Type	Definition
Axis	<p>createEqualAreaAxis(<i>nlat</i>)</p> <p>Create an equal-area latitude axis. The latitude values range from north to south, and for all axis values $x[i]$, $\sin(x[i]) - \sin(x[i+1])$ is constant.</p> <p><i>nlat</i> is the axis length.</p> <p>The axis is not associated with a file or dataset.</p>
Axis	<p>createGaussianAxis(<i>nlat</i>)</p> <p>Create a Gaussian latitude axis. Axis values range from north to south.</p> <p><i>nlat</i> is the axis length.</p> <p>The axis is not associated with a file or dataset.</p>
RectGrid	<p>createGaussianGrid(<i>nlats</i>, <i>xorigin</i>=0.0, <i>order</i>="yx")</p> <p>Create a Gaussian grid, with shape (<i>nlats</i>, $2 * nlats$).</p> <p><i>nlats</i> is the number of latitudes.</p> <p><i>xorigin</i> is the origin of the longitude axis.</p> <p><i>order</i> is either "yx" (lat-lon, default) or "xy" (lon-lat)</p>

Table 2.2 cdms module functions

Type	Definition
RectGrid	<p data-bbox="654 366 1300 465">createGenericGrid(latArray, lonArray, latBounds=None, lonBounds=None, order="yx", mask=None)</p> <p data-bbox="654 482 1300 574">Create a generic grid, that is, a grid which is not typed as Gaussian, uniform, or equal-area. The grid is not associated with a file or dataset.</p> <p data-bbox="654 591 1129 618"><i>latArray</i> is a NumPy array of latitude values.</p> <p data-bbox="654 635 1153 661"><i>lonArray</i> is a NumPy array of longitude values</p> <p data-bbox="654 678 1300 739"><i>latBounds</i> is a NumPy array having shape (len(latArray),2), of latitude boundaries.</p> <p data-bbox="654 756 1300 817"><i>lonBounds</i> is a NumPy array having shape (len(lonArray),2), of longitude boundaries.</p> <p data-bbox="654 835 1300 895"><i>order</i> is a string specifying the order of the axes, either “yx” for (latitude, longitude), or “xy” for the reverse.</p> <p data-bbox="654 913 1300 960"><i>mask</i> (optional) is an integer-valued NumPy mask array, having the same shape and ordering as the grid.</p>
RectGrid	<p data-bbox="654 991 1015 1017">createGlobalMeanGrid(grid)</p> <p data-bbox="654 1034 1300 1130">Generate a grid for calculating the global mean via a regrid-ding operation. The return grid is a single zone covering the range of the input grid.</p> <p data-bbox="654 1147 853 1175"><i>grid</i> is a RectGrid.</p>

Table 2.2 cdms module functions

Type	Definition
RectGrid	<p>createRectGrid(lat, lon, order, type="generic", mask=None)</p> <p>Create a rectilinear grid, not associated with a file or dataset. This might be used as the target grid for a regridding operation.</p> <p><i>lat</i> is a latitude axis, created by <code>cdms.createAxis</code>.</p> <p><i>lon</i> is a longitude axis, created by <code>cdms.createAxis</code>.</p> <p><i>order</i> is a string with value “yx” (the first grid dimension is latitude) or “xy” (the first grid dimension is longitude).</p> <p><i>type</i> is one of ‘gaussian’, ‘uniform’, ‘equalarea’, or ‘generic’</p> <p>If specified, <i>mask</i> is a two-dimensional, logical Numeric array (all values are zero or one) with the same shape as the grid.</p>
RectGrid	<p>createUniformGrid(startLat, nlat, deltaLat, startLon, nlon, deltaLon, order="yx", mask=None)</p> <p>Create a uniform rectilinear grid. The grid is not associated with a file or dataset. The grid boundaries are at the midpoints of the axis values.</p> <p><i>startLat</i> is the starting latitude value.</p> <p><i>nlat</i> is the number of latitudes. If <i>nlat</i> is 1, the grid latitude boundaries will be $\text{startLat} \pm \text{deltaLat}/2$.</p> <p><i>deltaLat</i> is the increment between latitudes.</p> <p><i>startLon</i> is the starting longitude value.</p> <p><i>nlon</i> is the number of longitudes. If <i>nlon</i> is 1, the grid longitude boundaries will be $\text{startLon} \pm \text{deltaLon}/2$.</p> <p><i>deltaLon</i> is the increment between longitudes.</p> <p><i>order</i> is a string with value “yx” (the first grid dimension is latitude) or “xy” (the first grid dimension is longitude).</p> <p>If specified, <i>mask</i> is a two-dimensional, logical Numeric array (all values are zero or one) with the same shape as the grid.</p>

Table 2.2 cdms module functions

Type	Definition
Axis	<p>createUniformLatitudeAxis(startLat, nlat, deltaLat)</p> <p>Create a uniform latitude axis. The axis boundaries are at the midpoints of the axis values. The axis is designated as a circular latitude axis.</p> <p><i>startLat</i> is the starting latitude value.</p> <p><i>nlat</i> is the number of latitudes.</p> <p><i>deltaLat</i> is the increment between latitudes.</p>
RectGrid	<p>createZonalGrid(grid)</p> <p>Create a zonal grid. The output grid has the same latitude as the input grid, and a single longitude. This may be used to calculate zonal averages via a regridding operation.</p> <p><i>grid</i> is a RectGrid.</p>
Axis	<p>createUniformLongitudeAxis(startLon, nlon, deltaLon)</p> <p>Create a uniform longitude axis. The axis boundaries are at the midpoints of the axis values. The axis is designated as a circular longitude axis.</p> <p><i>startLon</i> is the starting longitude value.</p> <p><i>nlon</i> is the number of longitudes.</p> <p><i>deltaLon</i> is the increment between longitudes.</p>
Variable	<p>createVariable(array, typecode=None, copy=0, savespace=0, mask=None, fill_value=None, grid=None, axes=None, attributes=None, id=None)</p> <p>This function is documented in Table 2.31 on page 82.</p>
Integer	<p>getAutoBounds()</p> <p>Get the current autobounds mode. Returns 1 if the autobounds mode is on, 0 otherwise. See setAutoBounds.</p>

Table 2.2 cdms module functions

Type	Definition
Integer	<p>isVariable(s)</p> <p>Return 1 if <i>s</i> is a variable, 0 otherwise. See also: asVariable.</p>
Dataset or CdmsFile	<p>open(url,mode='r')</p> <p>Open or create a Dataset or CdmsFile.</p> <p><i>url</i> is a Uniform Resource Locator, referring to a cdunif or XML file. If the URL has the extension '.xml' or '.cdml', a Dataset is returned, otherwise a CdmsFile is returned. If the URL protocol is 'http', the file must be a '.xml' or '.cdml' file, and the mode must be 'r'. If the protocol is 'file' or is omitted, a local file or dataset is opened.</p> <p><i>mode</i> is the open mode. See Table 2.22 on page 63.</p> <p>Example: Open an existing dataset:</p> <pre>f = cdms.open("sampleset.xml")</pre> <p>Example: Create a netCDF file:</p> <pre>f = cdms.open("newfile.nc", 'w')</pre>
List	<p>order2index (axes, orderstring)</p> <p>Find the index permutation of axes to match order. Return a list of indices</p> <p><i>axes</i> is a list of axis objects.</p> <p><i>orderstring</i> is defined as in orderparse.</p>

Table 2.2 cdms module functions

Type	Definition
List	<p>orderparse(orderstring)</p> <p>Parse an order string. Returns a list of axes specifiers.</p> <p><i>orderstring</i> consists of:</p> <ul style="list-style-type: none"> • Letters t, x, y, z meaning time, longitude, latitude, level • Numbers 0-9 representing position in axes • Dash (-) meaning insert the next available axis here. • The ellipsis ... meaning fill these positions with any remaining axes. • (name) meaning an axis whose id is name
None	<p>setAutoBounds(mode)</p> <p>Set autobounds mode.</p> <p>If <i>mode</i> is 'on' or 1 (the default), the getBounds method will automatically generate boundary information for an axis or grid, if the boundaries are not explicitly defined.</p> <p>If <i>mode</i> is 'off' or 0, and no boundary data is explicitly defined, the bounds will NOT be generated; the getBounds method will return None for the boundaries.</p>
None	<p>setClassifyGrids(mode)</p> <p>Set the grid classification mode. This affects how grid type is determined, for the purpose of generating grid boundaries.</p> <p>If <i>mode</i> is 'on' (the default), grid type is determined by a grid classification method, regardless of the value of <code>grid.getType()</code>.</p> <p>If <i>mode</i> is 'off', the value of <code>grid.getType()</code> determines the grid type</p>

Table 2.3 Class Tags

Tag	Class
'axis'	Axis
'database'	Database
'dataset'	Dataset, CdmsFile
'grid'	RectGrid
'variable'	Variable
'xlink'	Xlink

2.4 CdmsObj

A CdmsObj is the base class for all CDMS database objects. At the application level, CdmsObj objects are never created and used directly. Rather the subclasses of CdmsObj (Dataset, Variable, Axis, etc.) are the basis of user application programming.

All objects derived from CdmsObj have a special attribute **.attributes**. This is a Python dictionary, which contains all the external (persistent) attributes associated with the object. This is in contrast to the internal, non-persistent attributes of an object, which are built-in and predefined. When a CDMS object is written to a file, the external attributes are written, but not the internal attributes.

Example: get a list of all external attributes of obj.

```
extatts = obj.attributes.keys()
```

Table 2.4 Attributes common to all CDMS objects

Type	Name	Definition
Dictionary	attributes	External attribute dictionary for this object.

All attributes may be accessed and set using the Python dot notation (‘.’)

Table 2.5 Getting and setting attributes

Type	Definition
Various	<p>value = obj.attname</p> <p>Get an internal or external attribute value. If the attribute is external, it is read from the database. If the attribute is not already in the database, it is created as an external attribute. Internal attributes cannot be created, only referenced.</p> <p>obj.attname = value</p> <p>Set an internal or external attribute value. If the attribute is external, it is written to the database.</p>

2.5 Axis

An Axis is a one-dimensional coordinate object.

An Axis is contained in a Dataset. Setting a slice of an Axis writes data to the Dataset, referencing an Axis slice reads data from the Dataset. Axis objects are also used to define the domain of a Variable.

An axis in a CdmsFile may be designated the ‘unlimited’ axis, meaning that it can be extended in length after the initial definition. There can be at most one unlimited axis associated with a CdmsFile.

Table 2.6 Axis Internal Attributes

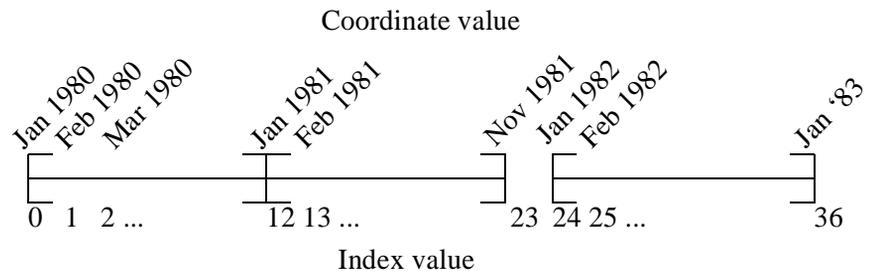
Type	Name	Definition
Dictionary	attributes	External attribute dictionary.

Table 2.6 Axis Internal Attributes

Type	Name	Definition
String	id	Axis identifier.
Dataset	parent	The dataset which contains the variable.
Tuple	shape	The length of each axis.

2.5.1 partition attribute

For an axis in a dataset, the **.partition** attribute describes how an axis is split across files. It is a list of the start and end indices of each axis partition.

FIGURE 1. Partitioned axis

For example, Figure 1 shows a time axis, representing the 36 months, January 1980 through December 1982, with December 1981 missing. The first partition interval is (0,12), the second is (12,23), and the third is (24,36), where the interval (i,j) represents all indices k such that $i \leq k < j$. The **.partition** attribute for this axis would be the list:

```
[0, 12, 23, 36]
```

Note that the end index of the second interval is strictly less than the start index of the following interval. This indicates that data for that period is missing.

Table 2.7 Axis Constructors

cdms.createAxis(data, bounds=None)

Create an axis which is not associated with a dataset or file. See Table 2.2 on page 25.

Dataset.createAxis(name,ar)

Create an Axis in a Dataset. (This function is not yet implemented.)

CdmsFile.createAxis(name,ar,unlimited=0)

Create an Axis in a CdmsFile.

name is the string name of the Axis.

ar is a 1-D data array which defines the Axis values. It may have the value None if an unlimited axis is being defined.

At most one Axis in a CdmsFile may be designated as being 'unlimited', meaning that it may be extended in length. To define an axis as unlimited, either:

- set *ar* to None, and leave *unlimited* undefined, or
- set *ar* to the initial 1-D array, and set *unlimited* to **cdms.Unlimited**

cdms.createEqualAreaAxis(nlat)

See Table 2.2 on page 25.

cdms.createGaussianAxis(nlat)

See Table 2.2 on page 18.

cdms.createUniformLatitudeAxis(startlat, nlat, deltalat)

See Table 2.2 on page 18.

cdms.createUniformLongitudeAxis(startlon, nlon, deltalon)

See Table 2.2 on page 18.

Table 2.8 Axis Methods

Type	Method Definition
Array	<p>array = axis[i:j]</p> <p>Read a slice of data from the external dataset. Data is returned in the physical ordering defined in the dataset. See Table 2.9 on page 43 for a description of slice operators.</p>
None	<p>axis[i:j] = array</p> <p>Write a slice of data to the external dataset. (axes in CdmsFiles only)</p>
List of component times	<p>asComponentTime(calendar=None)</p> <p>Array version of cftime tocomp. Returns a list of component times.</p>
List of relative times	<p>asRelativeTime()</p> <p>Array version of cftime torel. Returns a list of relative times.</p>
None	<p>assignValue(array)</p> <p>Set the entire value of the axis.</p> <p><i>array</i> is a one-dimensional, Numeric array.</p>
Axis	<p>clone(copyData=1)</p> <p>Return a copy of the axis, as a transient axis. If copyData is 1 (the default) the data itself is copied.</p>

Table 2.8 Axis Methods

Type	Method Definition
None	<p>designateCircular(modulo, persistent=0)</p> <p>Designate the axis to be circular.</p> <p><i>modulo</i> is the modulus value. Any given axis value <i>x</i> is treated as equivalent to <i>x+modulo</i></p> <p>If <i>persistent</i> is true, the external file or dataset (if any) is modified. By default, the designation is temporary.</p>
None	<p>designateLatitude(persistent=0):</p> <p>Designate the axis to be a latitude axis.</p> <p>If <i>persistent</i> is true, the external file or dataset (if any) is modified. By default, the designation is temporary.</p>
None	<p>designateLevel(persistent=0)</p> <p>Designate the axis to be a vertical level axis.</p> <p>If <i>persistent</i> is true, the external file or dataset (if any) is modified. By default, the designation is temporary.</p>
None	<p>designateLongitude(persistent=0, modulo=360.0)</p> <p>Designate the axis to be a longitude axis.</p> <p><i>modulo</i> is the modulus value. Any given axis value <i>x</i> is treated as equivalent to <i>x+modulo</i></p> <p>If <i>persistent</i> is true, the external file or dataset (if any) is modified. By default, the designation is temporary.</p>
None	<p>designateTime(persistent=0, calendar = cftime.MixedCalendar)</p> <p>Designate the axis to be a time axis.</p> <p>If <i>persistent</i> is true, the external file or dataset (if any) is modified. By default, the designation is temporary.</p> <p><i>calendar</i> is defined as in getCalendar().</p>

Table 2.8 Axis Methods

Type	Method Definition
Array	<p>getBounds()</p> <p>Get the associated boundary array. The boundary array has shape (n,2), where n is the length of the axis.</p> <p>If a boundary array is not explicitly defined and autoBounds mode is on, a default array is generated by calling genGenericBounds. Otherwise if autoBounds mode is off, the return value is None. See setAutoBounds.</p>
Integer	<p>getCalendar()</p> <p>Returns the calendar associated with the (time) axis. Possible return values, as defined in the cftime module, are:</p> <ul style="list-style-type: none"> • cftime.GregorianCalendar: the standard Gregorian calendar • cftime.MixedCalendar: mixed Julian/Gregorian calendar • cftime.JulianCalendar: years divisible by 4 are leap years • cftime.NoLeapCalendar: a year is 365 days • cftime.Calendar360: a year is 360 days • None: no calendar can be identified <p>Note: If the axis is not a time axis, the global, file-related calendar is returned.</p>
Array	<p>getValue()</p> <p>Get the entire axis vector.</p>
Integer	<p>isCircular()</p> <p>Returns true if the axis has circular topology.</p> <p>An axis is defined as circular if:</p> <ul style="list-style-type: none"> • axis.topology=='circular', or • axis.topology is undefined, and the axis is a longitude <p>The default cycle for circular axes is 360.0</p>

Table 2.8 Axis Methods

Type	Method Definition
Integer	isLatitude() Returns true iff the axis is a latitude axis.
Integer	isLevel() Returns true iff the axis is a level axis.
Integer	isLinear() Returns true iff the axis has a linear representation.
Integer	isLongitude() Returns true iff the axis is a longitude axis.
Integer	isTime() Returns true iff the axis is a time axis.
Integer	len(axis) The length of the axis.
Tuple	mapInterval(interval) Same as mapIntervalExt , but returns only the tuple (i,j), and wraparound is restricted to one cycle.

Table 2.8 Axis Methods

Type	Method Definition
(i,j,k)	<p data-bbox="606 347 921 373">mapIntervalExt(interval)</p> <p data-bbox="606 416 1149 442">Map a coordinate interval to an index interval.</p> <p data-bbox="606 486 1106 512"><i>interval</i> is a tuple having one of the forms:</p> <p data-bbox="614 555 849 685">(x,y) (x,y,indicator) (x,y,indicator,cycle) None or ':'</p> <p data-bbox="606 729 1220 789">where <i>x</i> and <i>y</i> are coordinates indicating the interval [x,y), and:</p> <p data-bbox="606 833 1263 1032"><i>indicator</i> is a two or three-character string, where the first character is 'c' if the interval is closed on the left, 'o' if open, and the second character has the same meaning for the right-hand point. If present, the third character specifies how the interval should be intersected with the axis:</p> <ul data-bbox="606 1058 1263 1275" style="list-style-type: none"> • 'n' - select node values which are contained in the interval • 'b' - select axis elements for which the corresponding cell boundary intersects the interval • 'e' - same as 'n', but include an extra node on either side • 's' - select axis elements for which the cell boundary is a subset of the interval <p data-bbox="606 1319 1206 1380">The default indicator is 'ccn', that is, the interval is closed, and nodes in the interval are selected.</p> <p data-bbox="606 1423 1263 1553">If <i>cycle</i> is specified, the axis is treated as circular with the given cycle value. By default, if <code>axis.isCircular()</code> is true, the axis is treated as circular with a default modulus of 360.0.</p> <p data-bbox="606 1597 1263 1657">An interval of None or ':' returns the full index interval of the axis.</p>

(continued)

Table 2.8 Axis Methods

Type	Method Definition
None	<p>(mapInterval, continued)</p> <p>The method returns the corresponding index interval as a 3-tuple (i,j,k), where k is the integer stride, and [i,j) is the half-open index interval $i \leq k < j$ ($i >= k > j$ if $k < 0$), or None if the intersection is empty.</p> <p>For an axis which is circular (<code>axis.topology == 'circular'</code>), [i,j) is interpreted as follows (where $N = \text{len}(\text{axis})$):</p> <ul style="list-style-type: none">• if $0 \leq i < N$ and $0 \leq j \leq N$, the interval does not wrap around the axis endpoint• otherwise the interval wraps around the axis endpoint. <p>See also: <code>mapInterval</code>, <code>Variable.subRegion()</code></p>

Table 2.8 Axis Methods

Type	Method Definition
None	<p>setBounds(bounds, validate=0, boundsid=None, persistent=0, index=None)</p> <p>Set the axis bounds.</p> <p><i>bounds</i> is a Numeric array, masked array (MA), or masked variable (MV) with shape (N,2) where N is the axis length.</p> <p>If <i>validate</i> is 1, an exception is raised if either of the tests fail:</p> <ul style="list-style-type: none"> • <i>bounds</i> must have shape (N,2) • for all indices <i>i</i>, <i>axis[i]</i> must be in the interval [<i>bounds[i,0]</i>, <i>bounds[i,1]</i>] <p>The remaining optional keywords apply only to FileAxis objects (axes in external files):</p> <p><i>boundsid</i> is the string name of the bounds array as written to the file. By default, the id is ‘bounds_<axisid>’ where ‘<axisid>’ is the axis identifier.</p> <p>If <i>persistent</i>=1, the bounds are written to the file.</p> <p>For <i>index</i> set to an integer <i>n</i>, the bounds array is written starting at index <i>n</i>. This is useful when the axis is the ‘unlimited’ dimension and is being written iteratively.</p> <p>Notes:</p> <ul style="list-style-type: none"> • When using the write call, it is not necessary to call setBounds. This is done automatically.
None	<p>setCalendar(calendar, persistent=1)</p> <p>Set the calendar for this (time) axis.</p> <p><i>calendar</i> is defined as in getCalendar().</p> <p>If <i>persistent</i> is true, the external file or dataset (if any) is modified. This is the default.</p>

Table 2.8 Axis Methods

Type	Method Definition
TransientAxis	subAxis(i,j,k=1) Create an axis associated with the integer range [i:j:k]. The stride k can be positive or negative. Wraparound is supported for longitude dimensions or those with a modulus attribute.
String	typecode() The Numeric datatype identifier.

Table 2.9 Axis Slice Operators

Slice	Definition
[i]	The ith element, starting with index 0
[i:j]	The ith element through, but not including, element j
[i:]	The ith element through and including the end
[:j]	The beginning element through, but not including, element j
[:]	The entire array
[i:j:k]	Every kth element, starting at i, through but not including j
[-i]	The ith element from the end. -1 is the last element.

Example: A longitude axis has value [0.0, 2.0, ..., 358.0], of length 180. Map the coordinate interval $-5.0 \leq x < 5.0$ to index interval(s), with wrap-around. The result index interval $-2 \leq n < 3$ wraps around, since $-2 < 0$, and has a stride of 1. This is equivalent to the two contiguous index intervals $-2 \leq n < 0$ and $0 \leq n < 3$

```
> axis.isCircular()
1
> axis.mapIntervalExt((-5.0,5.0,'co')
(-2,3,1)
>
```

2.6 *CdmsFile*

A *CdmsFile* is a physical file, accessible via the *cdunif* interface. *netCDF* files are accessible in read-write mode. All other formats (DRS, HDF, GrADS/GRIB, POP, QL) are accessible read-only.

As of CDMS V3, the legacy *cuDataset* interface is also supported by *CdmsFiles*. See “*cu Module*” on page 161.

Table 2.10 *CdmsFile* Internal Attributes

Type	Name	Definition
Dictionary	<i>attributes</i>	Global, external file attributes
Dictionary	<i>axes</i>	Axis objects contained in the file.
Dictionary	<i>grids</i>	Grids contained in the file.
String	<i>id</i>	File pathname.
Dictionary	<i>variables</i>	Variables contained in the file.

Table 2.11 *CdmsFile* Constructors

fileobj* = *cdms.open(path, mode)

Open the file specified by *path* returning a *CdmsFile* object.

path is the file pathname, a string.

mode is the open mode indicator, as listed in Table 2.22 on page 63.

fileobj* = *cdms.createDataset(path)

Create the file specified by *path*, a string.

Table 2.12 CdmsFile Methods

Type	Definition
Transient-Variable	<p><i>fileobj</i>(varname, selector)</p> <p>Calling a CdmsFile object as a function reads the region of data specified by the selector. The result is a transient variable, unless <code>raw=1</code> is specified. See “Selectors” on page 95.</p> <p>For example, the following reads data for variable ‘prc’, year 1980:</p> <pre>f = cdms.open('test.nc') x = f('prc', time=('1980-1','1981-1'))</pre>
Variable, Axis, or Grid	<p><i>fileobj</i>['id']</p> <p>Get the persistent variable, axis or grid object having the string identifier. This does not read the data for a variable.</p> <p>For example:</p> <pre>f = cdms.open('sample.nc') v = f['prc']</pre> <p>gets the persistent variable <code>v</code>, equivalent to <code>v=f.variables['prc']</code>.</p> <pre>t = f['time']</pre> <p>gets the axis named ‘time’, equivalent to <code>t=f.axes['time']</code>.</p>
None	<p><i>close</i>()</p> <p>Close the file.</p>

Table 2.12 CdmsFile Methods

Type	Definition
Axis	<p>copyAxis(axis, newname=None)</p> <p>Copy axis values and attributes to a new axis in the file. The returned object is persistent: it can be used to write axis data to or read axis data from the file. If an axis already exists in the file, having the same name and coordinate values, it is returned. It is an error if an axis of the same name exists, but with different coordinate values.</p> <p><i>axis</i> is the axis object to be copied.</p> <p><i>newname</i>, if specified, is the string identifier of the new axis object. If not specified, the identifier of the input axis is used.</p>
Grid	<p>copyGrid(grid, newname=None)</p> <p>Copy grid values and attributes to a new grid in the file. The returned grid is persistent. If a grid already exists in the file, having the same name and axes, it is returned. An error is raised if a grid of the same name exists, having different axes.</p> <p><i>grid</i> is the grid object to be copied.</p> <p><i>newname</i>, if specified is the string identifier of the new grid object. If unspecified, the identifier of the input grid is used.</p>
Axis	<p>createAxis(id, ar, unlimited=0)</p> <p>Create a new Axis. This is a persistent object which can be used to read or write axis data to the file.</p> <p><i>id</i> is an alphanumeric string identifier, containing no blanks.</p> <p><i>ar</i> is the one-dimensional axis array.</p> <p>Set <i>unlimited</i> to <code>cdms.Unlimited</code> to indicate that the axis is extensible.</p>

Table 2.12 CdmsFile Methods

Type	Definition
RectGrid	<p>createRectGrid(id, lat, lon, order, type="generic", mask=None)</p> <p>Create a RectGrid in the file. This is not a persistent object: the order, type, and mask are not written to the file. However, the grid may be used for regridding operations.</p> <p><i>lat</i> is a latitude axis in the file.</p> <p><i>lon</i> is a longitude axis in the file.</p> <p><i>order</i> is a string with value “yx” (the first grid dimension is latitude) or “xy” (the first grid dimension is longitude).</p> <p><i>type</i> is one of 'gaussian', 'uniform', 'equalarea', or 'generic'</p> <p>If specified, <i>mask</i> is a two-dimensional, logical Numeric array (all values are zero or one) with the same shape as the grid.</p>
Variable	<p>createVariable(String id, String datatype, List axes, fill_value=None)</p> <p>Create a new Variable. This is a persistent object which can be used to read or write variable data to the file.</p> <p><i>id</i> is a String name which is unique with respect to all other objects in the file.</p> <p><i>datatype</i> is an MA typecode, e.g., MA.Float, MA.Int.</p> <p><i>axes</i> is a list of Axis and/or Grid objects.</p> <p><i>fill_value</i> is the missing value (optional).</p>

Table 2.12 CdmsFile Methods

Type	Definition
Variable	createVariableCopy(var, newname=None) Create a new Variable, with the same name, axes, and attributes as the input variable. An error is raised if a variable of the same name exists in the file. <i>var</i> is the Variable to be copied. <i>newname</i> , if specified is the name of the new variable. If unspecified, the returned variable has the same name as <i>var</i> . Note: Unlike <code>copyAxis</code> , the actual data is not copied to the new variable.
None	sync() Writes any pending changes to the file.

Table 2.12 CdmsFile Methods

Type	Definition
Variable	<p data-bbox="655 366 1305 465">write(var, attributes=None, axes=None, extbounds=None, id=None, extend=None, fill_value=None, index=None, typecode=None)</p> <p data-bbox="655 482 1276 539">Write a variable or array to the file. The return value is the associated file variable.</p> <p data-bbox="655 557 1315 774">If the variable does not exist in the file, it is first defined and all attributes written, then the data is written. By default, the time dimension of the variable is defined as the 'unlimited' dimension of the file. If the data is already defined, then data is extended or overwritten depending on the value of keywords <i>extend</i> and <i>index</i>, and the unlimited dimension values associated with <i>var</i>.</p> <p data-bbox="655 791 1182 817"><i>var</i> is a Variable, masked array, or Numeric array.</p> <p data-bbox="655 835 1253 892"><i>attributes</i> is the attribute dictionary for the variable. The default is <i>var.attributes</i>.</p> <p data-bbox="655 909 1305 966"><i>axes</i> is the list of file axes comprising the domain of the variable. The default is to copy <i>var.getAxisList()</i>.</p> <p data-bbox="655 984 1268 1041"><i>extbounds</i> is the unlimited dimension bounds. Defaults to <i>var.getAxis(0).getBounds()</i></p> <p data-bbox="655 1058 1193 1085"><i>id</i> is the variable name in the file. Default is <i>var.id</i>.</p> <p data-bbox="655 1102 1290 1222"><i>extend=1</i> causes the first dimension to be 'unlimited': iteratively writeable. The default is <i>None</i>, in which case the first dimension is extensible if it is time. Set to 0 to turn off this behaviour.</p> <p data-bbox="655 1239 1029 1265"><i>fill_value</i> is the missing value flag.</p> <p data-bbox="655 1282 1315 1373"><i>index</i> is the extended dimension index to write to. The default index is determined by lookup relative to the existing extended dimension.</p> <p data-bbox="655 1390 1305 1480">Note: data can also be written by setting a slice of a file variable, and attributes can be written by setting an attribute of a file variable.</p>

Table 2.13 CDMS Datatypes

CDMS Datatype	Definition
CdChar	character
CdDouble	double-precision floating-point
CdFloat	floating-point
CdInt	integer
CdLong	long integer
CdShort	short integer

2.7 Database

A Database is a collection of datasets and other CDMS objects. It consists of a hierarchical collection of objects, with the database being at the root, or top of the hierarchy. A database is used to:

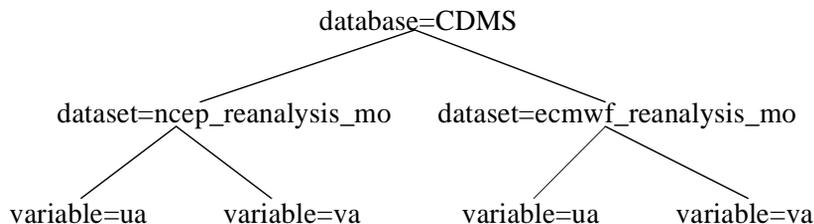
- search for metadata
- access data
- provide authentication and access control for data and metadata

The figure below illustrates several important points:

- Each object in the database has a *relative name* of the form *tag=id*. The id of an object is unique with respect to all objects contained in the parent.
- The *name* of the object consists of its relative name followed by the relative name(s) of its antecedent objects, up to and including the database name. In the figure below, one of the variables has name

`"variable=ua, dataset=ncep_reanalysis_mo,database=CDMS"`.

- Subordinate objects are thought of as being contained in the parent. In this example, the database 'CDMS' contains two datasets, each of which contain several variables.



2.7.1 Overview

To access a database:

1. Open a connection. The **connect** method opens a database connection. **connect** takes a database URI and returns a database object:

```
db = cdms.connect("ldap://dbhost.llnl.gov/  
database=CDMS,ou=PCMDI,o=LLNL,c=US")
```
2. Search the database, locating one or more datasets, variables, and/or other objects.

The database **searchFilter** method searches the database. A single database connection may be used for an arbitrary number of searches.

For example, to find all observed datasets:

```
result = db.searchFilter("category=observed",tag="dataset")
```

Searches can be restricted to a subhierarchy of the database. This example searches just the dataset 'ncep_reanalysis_mo':

```
result = db.searchFilter(rebase="dataset=ncep_reanalysis")
```

3. Refine the search results if necessary. The result of a search can be narrowed with the **searchPredicate** method.
4. Process the results. A search result consists of a sequence of entries. Each entry has a name, the name of the CDMS object, and an attribute dictionary, consisting of the attributes located by the search:


```
for entry in result:
    print entry.name, entry.attributes
```
5. Access the data. The CDMS object associated with an entry is obtained from the **getObject** method:


```
obj = entry.getObject()
```

If the id of a dataset is known, the dataset can be opened directly with the **open** method:

- ```
dset = db.open("ncep_reanalysis_mo")
```
6. Close the database connection:
 

```
db.close()
```

**Table 2.14 Database Internal Attributes**

---

| Type       | Name       | Summary                              |
|------------|------------|--------------------------------------|
| Dictionary | attributes | Database attribute dictionary        |
| LDAP       | db         | (LDAP only) LDAP database object     |
| String     | netloc     | Hostname, for server-based databases |
| String     | path       | path name                            |
| String     | uri        | Uniform Resource Identifier.         |

Table 2.15 Database Constructors

```
db = cdms.connect(uri=None, user="", password="")
```

Connect to the database.

*uri* is the Universal Resource Identifier of the database. The form of the URI depends on the implementation of the database. For a Lightweight Directory Access Protocol (LDAP) database, the form is:

```
ldap://host[:port]/dbname
```

For example, if the database is located on host ‘dbhost.llnl.gov’, and is named ‘database=CDMS,ou=PCMDI,o=LLNL,c=US’, the URI is:

```
ldap://dbhost.llnl.gov/database=CDMS,ou=PCMDI,o=LLNL,c=US
```

If unspecified, the URI defaults to the value of environment variable **CDMSROOT**.

*user* is the user ID. If unspecified, an anonymous connection is made.

*password* is the user password. A password is not required for an anonymous connection.

Table 2.16 Database Methods

| Type | Definition                                                                                                                         |
|------|------------------------------------------------------------------------------------------------------------------------------------|
| None | <b>close()</b><br>Close a database connection.                                                                                     |
| List | <b>listDatasets()</b><br>Return a list of the dataset IDs in this database. A dataset ID can be passed to the <b>open</b> command. |

**Table 2.16 Database Methods**

---

| Type    | Definition                                                                                                                                                                                                                                           |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dataset | <p><b>open(dsetid, mode='r')</b><br/>Open a dataset.<br/><i>dsetid</i> is the string dataset identifier<br/><i>mode</i> is the open mode, 'r' - read-only, 'r+' - read-write, 'w' - create.<br/><b>openDataset</b> is a synonym for <b>open</b>.</p> |

Table 2.16 Database Methods

| Type         | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SearchResult | <p data-bbox="654 366 1319 430"><b>searchFilter(filter=None, tag=None, rebase=None, scope=Subtree, attnames=None, timeout=None)</b></p> <p data-bbox="654 447 929 473">Search a CDMS database.</p> <p data-bbox="654 491 1319 678"><i>filter</i> is the string search filter. Simple filters have the form "tag = value". Simple filters can be combined using logical operators '&amp;', ' ', '!' in prefix notation. For example, the filter '(&amp;(objectclass=variable)(id=cli))' finds all variables named "cli". A formal definition of search filters is provided in the following section.</p> <p data-bbox="654 696 1265 751"><i>tag</i> restricts the search to objects with that tag ("dataset"   "variable"   "database"   "axis"   "grid").</p> <p data-bbox="654 769 1319 892"><i>rebase</i> is the relative name of the base object of the search. The search is restricted to the base object and all objects below it in the hierarchy. For example, to search only dataset 'ncep_reanalysis_mo', specify:</p> <pre data-bbox="701 909 1172 935">rebase="dataset=ncep_reanalysis_mo".</pre> <p data-bbox="654 970 1258 996">To search only variable 'ua' in ncep_reanalysis_mo, use:</p> <pre data-bbox="701 1013 1093 1060">rebase="variable=ua, dataset=ncep_reanalysis_mo"</pre> <p data-bbox="654 1095 1319 1150">If no base is specified, the entire database is searched. See the <i>scope</i> argument also.</p> <p data-bbox="654 1168 1319 1291"><i>scope</i> is the search scope (<b>Subtree</b>   <b>Onelevel</b>   <b>Base</b>). <b>Subtree</b> searches the base object and its descendants. <b>Onelevel</b> searches the base object and its immediate descendants. <b>Base</b> searches the base object alone. The default is <b>Subtree</b>.</p> <p data-bbox="654 1308 1279 1399"><i>attnames</i>: list of attribute names. Restricts the attributes returned. If None, all attributes are returned. Attributes 'id' and 'objectclass' are always included in the list.</p> <p data-bbox="654 1416 1248 1472"><i>timeout</i>: integer number of seconds before timeout. The default is no timeout.</p> |

## 2.7.2 Searching a database

The `searchFilter` method is used to search a database. The result is called a *search result*, and consists of a sequence of *result entries*.

In its simplest form, `searchFilter` takes an argument consisting of a string filter. The search returns a sequence of entries, corresponding to those objects having an attribute which matches the filter. Simple filters have the form (*tag = value*), where *value* can contain wildcards. For example:

```
'(id = ncep*)'
'(project = AMIP2)'
```

Simple filters can be combined with the logical operators '&', '|', '!'. For example,

```
'(&(id = bmrc*)(project = AMIP2))'
```

matches all objects with id starting with 'bmrc', and a 'project' attribute with value 'AMIP2'.

Formally, search filters are strings defined as follows:

```
filter ::= "(" filtercomp ")"
filtercomp ::= "&" filterlist | # and
 "|" filterlist | # or
 "!" filterlist | # not
 simple
filterlist ::= filter | filter filterlist
simple ::= tag op value
op ::= "=" | # equality
 "~=" | # approximate equality
 "<=" | # lexicographically less than or equal to
 ">=" | # lexicographically greater than or equal to
tag ::= string attribute name
value ::= string attribute value, may include '*' as a wild card
```

Attribute names are defined in the chapter on “Climate Data Markup Language (CDML)” on page 135. In addition, some special attributes are defined for convenience:

- **category** is either “experimental” or “observed”
- **parentid** is the ID of the parent dataset

- **project** is a project identifier, e.g., “AMIP2”
- **objectclass** is the list of tags associated with the object.

The set of objects searched is called the search *scope*. The top object in the hierarchy is the *base object*. By default, all objects in the database are searched, that is, the database is the base object. If the database is very large, this may result in an unnecessarily slow or inefficient search. To remedy this the search scope can be limited in several ways:

- The base object can be changed.
- The scope can be limited to the base object and one level below, or to just the base object.
- The search can be restricted to objects of a given class (dataset, variable, etc.)
- The search can be restricted to return only a subset of the object attributes
- The search can be restricted to the result of a previous search.

A search result is accessed sequentially within a for loop:

```
result = db.searchFilter('&(category=obs*)(id=ncep*)')
for entry in result:
 print entry.name
```

Search results can be narrowed using **searchPredicate**. In the following example, the result of one search is itself searched for all variables defined on a 94x192 grid:

```
>>> result = db.searchFilter('parentid=ncep*', tag="variable")
>>> len(result)
65
>>> result2 = result.searchPredicate(lambda x:
 x.getGrid().shape==(94,192))
>>> len(result2)
3
>>> for entry in result2: print entry.name
variable=rluscs,dataset=ncep_reanalysis_mo,database=CDMS,ou=PCMDI,
o=LLNL, c=US
variable=rlds,dataset=ncep_reanalysis_mo,database=CDMS,ou=PCMDI,
o=LLNL, c=US
variable=rhus,dataset=ncep_reanalysis_mo,database=CDMS,ou=PCMDI,
o=LLNL, c=US
>>>
```

Table 2.17 SearchResult Methods

| Type         | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ResultEntry  | <p><b>[i]</b></p> <p>Return the i-th search result. Results can also be returned in a <b>for</b> loop:</p> <pre>for entry in db.searchResult(tag="dataset"):     ...</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Integer      | <p><b>len()</b></p> <p>Number of entries in the result.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| SearchResult | <p><b>searchPredicate(predicate, tag=None)</b></p> <p>Refine a search result, with a predicate search.</p> <p><i>predicate</i> is a function which takes a single CDMS object and returns true (1) if the object satisfies the predicate, 0 if not.</p> <p><i>tag</i> restricts the search to objects of the class denoted by the tag.</p> <p>Note: In the current implementation, <b>searchPredicate</b> is much less efficient than <b>searchFilter</b>. For best performance, use <b>searchFilter</b> to narrow the scope of the search, then use <b>searchPredicate</b> for more general searches.</p> |

A search result is a sequence of result entries. Each entry has a string name, the name of the object in the database hierarchy, and an attribute dictionary. An entry corresponds to an object found by the search, but differs from the object, in that only the attributes requested are associated with the entry. In general, there will be much more information defined for the associated CDMS object, which is retrieved with the **getObject** method.

**Table 2.18 ResultEntry Attributes**

| Type       | Name       | Summary                                                                                                             |
|------------|------------|---------------------------------------------------------------------------------------------------------------------|
| String     | name       | The name of this entry in the database.                                                                             |
| Dictionary | attributes | The attributes returned from the search.<br>attributes[key] is a list of all string values associated with the key. |

**Table 2.19 ResultEntry Methods**

| Type    | Definition                                                                                                                                                                                                                                                                                              |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CdmsObj | <b>getObject()</b><br>Return the CDMS object associated with this entry.<br>Note: For many search applications it is unnecessary to access the associated CDMS object. For best performance this function should be used only when necessary, for example, to retrieve data associated with a variable. |

### 2.7.3 Accessing data

To access data via CDMS:

1. Locate the dataset ID. This may involve searching the metadata.
2. Open the dataset, using the open method.
3. Reference the portion of the variable to be read.

In the next example, a portion of variable 'ua' is read from dataset 'ncep\_reanalysis\_mo':

```
dset = db.open('ncep_reanalysis_mo')
ua = dset.variables['ua']
data = ua[0,0]
```

## 2.7.4 Examples of database searches

In the following examples, `db` is the database opened with

```
db = cdms.connect()
```

This defaults to the database defined in environment variable `CDMSROOT`.

List all variables in dataset 'ncep\_reanalysis\_mo':

```
for entry in db.searchFilter(filter="parentid=ncep_reanalysis_mo",
 tag="variable"):
 print entry.name
```

Find all axes with bounds defined:

```
for entry in db.searchFilter(filter="bounds=*", tag="axis"):
 print entry.name
```

Locate all GDT datasets:

```
for entry in
 db.searchFilter(filter="Conventions=GDT*", tag="dataset"):
 print entry.name
```

Find all variables with missing time values, in observed datasets:

```
def missingTime(obj):
 time = obj.getTime()
 return time.length != time.partition_length

result = db.searchFilter(filter="category=observed")
for entry in result.searchPredicate(missingTime):
 print entry.name
```

Find all CMIP2 datasets having a variable with id "hfss":

```
for entry in
 db.searchFilter(filter="(&(project=CMIP2)(id=hfss))", tag="variable"):
 print entry.getObject().parent.id
```

Find all observed variables on 73x144 grids:

```
result = db.searchFilter('category=obs*')
for entry in result.searchPredicate(lambda x:
 x.getGrid().shape==(73,144),tag="variable"):
 print entry.name
```

Find all observed variables with more than 1000 timepoints:

```
result = db.searchFilter('category=obs*')
for entry in result.searchPredicate(lambda x: len(x.getTime())>1000,
 tag="variable"):
 print entry.name, len(entry.getObject().getTime())
```

Find the total number of each type of object in the database

```
print len(db.searchFilter(tag="database")), "database"
print len(db.searchFilter(tag="dataset")), "datasets"
print len(db.searchFilter(tag="variable")), "variables"
print len(db.searchFilter(tag="axis")), "axes"
```

---

## 2.8 Dataset

A Dataset is a virtual file. It consists of a metafile, in CDML/XML representation, and one or more data files.

As of CDMS V3, the legacy cuDataset interface is supported by Datasets. See “cu Module” on page 161.

**Table 2.20 Dataset Internal Attributes**

---

| Type       | Name       | Summary                                                                                      |
|------------|------------|----------------------------------------------------------------------------------------------|
| Dictionary | attributes | Dataset external attributes.                                                                 |
| Dictionary | axes       | Axes contained in the dataset.                                                               |
| String     | datapath   | Path of data files, relative to the parent database. If no parent, the datapath is absolute. |

Table 2.20 Dataset Internal Attributes

| Type       | Name      | Summary                                                                                            |
|------------|-----------|----------------------------------------------------------------------------------------------------|
| Dictionary | grids     | Grids contained in the dataset.                                                                    |
| String     | mode      | Open mode.                                                                                         |
| Database   | parent    | Database which contains this dataset. If the dataset is not part of a database, the value is None. |
| String     | uri       | Uniform Resource Identifier of this dataset.                                                       |
| Dictionary | variables | Variables contained in the dataset.                                                                |
| Dictionary | xlinks    | External links contained in the dataset.                                                           |

Table 2.21 Dataset Constructors

***datasetobj* = cdms.open(String uri, String mode='r')**

Open the dataset specified by the Universal Resource Indicator, a CDML file. Returns a Dataset object. mode is one of the indicators listed in Table 2.22 on page 63.

**openDataset** is a synonym for **open**.

***datasetobj* = cdms.createDataset(String path, String directory, String fileTemplate)**

**(Note: this function is not yet implemented)**

Create a new dataset, returning a Dataset object. *path* is the filepath of a CDML file. *fileTemplate* describes how the dataset is to be partitioned. It is a pathname, relative to the directory, which contains zero or more template specifiers (see Table 2.23 on page 63). A template may contain directory names as well as file names. A template specifier is a string of the form '%X' or '%eX', where X is one of the characters listed Table 2.23 on page 63. The form '%eX' may be used to specify the end time or level value. A specifier may appear more than once in a template.

Table 2.22 Open Modes

| Mode | Definition                                                          |
|------|---------------------------------------------------------------------|
| 'r'  | read-only                                                           |
| 'r+' | read-write                                                          |
| 'a'  | read-write. Open the file if it exists, otherwise create a new file |
| 'w'  | Create a new file, read-write                                       |

Table 2.23 Template Specifiers

| Specifier | Definition                           | Example           |
|-----------|--------------------------------------|-------------------|
| d         | day number                           | 1 .. 31           |
| f         | day number, two-digit, zero-filled   | 01 .. 31          |
| g         | month, lower case, three characters  | 'jan', 'feb', ... |
| G         | month, upper case, three characters  | 'JAN', 'FEB', ... |
| H         | hour                                 | 0 .. 23           |
| L         | vertical level                       | integer           |
| m         | month number, not zero filled        | 1 .. 12           |
| M         | minute                               | 0 .. 59           |
| n         | month number, two-digit, zero-filled | 01, 02, ..., 12   |
| S         | second                               | 0 .. 59           |
| v         | variable ID                          | character         |
| y         | year, two-digit, zero-filled         | integer           |
| Y         | year                                 | integer           |

Table 2.23 Template Specifiers

| Specifier | Definition   | Example          |
|-----------|--------------|------------------|
| z         | Zulu time    | ex: '6Z19990201' |
| %         | percent sign | '%'              |

For example, the file template

```
ccsr-a/mo/%v/ccsr-a/%v_ccsr-a_%Y.%n-%eY.%en.nc
```

contains the specifiers `%v` (variable name), `%Y` (year), `%eY` (end year), and `%en` (end month). One of the files in the dataset might have the path (relative to the parent directory)

```
ccsr-a/mo/ta/ccsr-a/ta_ccsr-a_1979.01-1979.12.nc
```

Table 2.24 Dataset Methods

| Type               | Definition                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Transient-Variable | <p><b><i>datasetobj</i>(varname, selector)</b></p> <p>Calling a Dataset object as a function reads the region of data defined by the selector. The result is a transient variable, unless <code>raw=1</code> is specified. See “Selectors” on page 95.</p> <p>For example, the following reads data for variable ‘prc’, year 1980:</p> <pre>f = cdms.open('test.xml') x = f('prc', time=('1980-1','1981-1'))</pre> |

Table 2.24 Dataset Methods

| Type                    | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Variable, Axis, or Grid | <p><b><i>datasetobj</i>['id']</b></p> <p>The square bracket operator applied to a dataset gets the persistent variable, axis or grid object having the string identifier. This does not read the data for a variable. Returns None if not found.</p> <p>For example:</p> <pre>f = cdms.open('sample.xml') v = f['prc']</pre> <p>gets the persistent variable <i>v</i>, equivalent to <code>v=f.variables['prc']</code>.</p> <pre>t = f['time']</pre> <p>gets the axis named 'time', equivalent to <code>t=f.axes['time']</code>.</p>                                                                                                                                                                                                                  |
| None                    | <p><b>close()</b></p> <p>Close the dataset.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| RectGrid                | <p><b>createRectGrid(id, lat, lon, order, type="generic", mask=None)</b></p> <p>Create a RectGrid in the dataset. This is not a persistent object: the order, type, and mask are not written to the dataset. However, the grid may be used for regridding operations.</p> <p><i>lat</i> is a latitude axis in the dataset.</p> <p><i>lon</i> is a longitude axis in the dataset.</p> <p><i>order</i> is a string with value "yx" (the first grid dimension is latitude) or "xy" (the first grid dimension is longitude).</p> <p><i>type</i> is one of 'gaussian', 'uniform', 'equalarea', or 'generic'</p> <p>If specified, <i>mask</i> is a two-dimensional, logical Numeric array (all values are zero or one) with the same shape as the grid.</p> |
| Axis                    | <p><b>getAxis(id)</b></p> <p>Get an axis object from the file or dataset.</p> <p><i>id</i> is the string axis identifier.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

**Table 2.24 Dataset Methods**

| Type     | Definition                                                                                                                                                                                    |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Grid     | <b>getGrid(id)</b><br>Get a grid object from a file or dataset.<br><i>id</i> is the string grid identifier.                                                                                   |
| List     | <b>getPaths()</b><br>Get a sorted list of pathnames of datafiles which comprise the dataset. This does not include the XML metafile path, which is stored in the <code>.uri</code> attribute. |
| Variable | <b>getVariable(id)</b><br>Get a variable object from a file or dataset.<br><i>id</i> is the string variable identifier.                                                                       |
| None     | <b>sync()</b><br>Write any pending changes to the dataset.                                                                                                                                    |

---

## 2.9 *MV module*

The fundamental CDMS data object is the variable. A variable is comprised of:

- a masked data array, as defined in the NumPy MA module.
- a domain: an ordered list of axes and/or grids.
- an attribute dictionary.

The MV module is a work-alike replacement for the MA module, that carries along the domain and attribute information where appropriate. MV pro-

vides the same set of functions as MA. However, MV functions generate transient variables as results. Often this simplifies scripts that perform computation. MA is part of the Python Numeric package, documented at <http://numpy.sourceforge.net>.

MV can be imported with the command:

```
import MV
```

The command

```
from MV import *
```

allows use of MV commands without any prefix.

Table 2.25 on page 68 lists the constructors in **MV**. All functions return a transient variable. In most cases the keywords *axes*, *attributes*, and *id* are available. *axes* is a list of axis objects which specifies the domain of the variable. *attributes* is a dictionary. *id* is a special attribute string that serves as the identifier of the variable, and should not contain blanks or non-printing characters. It is used when the variable is plotted or written to a file. Since the *id* is just an attribute, it can also be set like any attribute:

```
var.id = 'temperature'
```

For completeness MV provides access to all the MA functions. The functions not listed in the following tables are identical to the corresponding MA function: **allclose**, **allequal**, **common\_fill\_value**, **compress**, **create\_mask**, **dot**, **e**, **fill\_value**, **filled**, **get\_print\_limit**, **getmask**, **getmaskarray**, **identity**, **indices**, **innerproduct**, **isMA**, **isMaskedArray**, **is\_mask**, **isarray**, **make\_mask**, **make\_mask\_none**, **mask\_or**, **masked**, **pi**, **put**, **putmask**, **rank**, **ravel**, **set\_fill\_value**, **set\_print\_limit**, **shape**, **size**. See the documentation at <http://numpy.sourceforge.net> for a description of these functions.

Table 2.25 Variable Constructors in module MV

**arrayrange(start, stop=None, step=1, typecode=None, axis=None, attributes=None, id=None)**

Just like MA.arange() except it returns a variable whose type can be specified by the keyword argument typecode. The axis, attribute dictionary, and string identifier of the result variable may be specified.

Synonym: **arange**

**masked\_array(a, mask=None, fill\_value=None, axes=None, attributes=None, id=None)**

Same as MA.masked\_array but creates a variable instead. If no axes are specified, the result has default axes, otherwise *axes* is a list of axis objects matching a.shape.

**masked\_object(data, value, copy=1, savespace=0, axes=None, attributes=None, id=None)**

Create variable masked where exactly data equal to value. Create the variable with the given list of axis objects, attribute dictionary, and string id.

**masked\_values(data, value, rtol=1e-05, atol=1e-08, copy=1, savespace=0, axes=None, attributes=None, id=None)**

Constructs a variable with the given list of axes and attribute dictionary, whose mask is set at those places where

$\text{abs}(\text{data} - \text{value}) < \text{atol} + \text{rtol} * \text{abs}(\text{data})$ .

This is a careful way of saying that those elements of the data that have value = value (to within a tolerance) are to be treated as invalid. If data is not of a floating point type, calls masked\_object instead.

**ones(shape, typecode='f', savespace=0, axes=None, attributes=None, id=None)**

Return an array of all ones of the given length or shape.

Table 2.25 Variable Constructors in module MV

**reshape(a, newshape, axes=None, attributes=None, id=None)**

Copy of a with a new shape.

**resize(a, new\_shape, axes=None, attributes=None, id=None)**

Return a new array with the specified shape. The original array's total size can be any size.

**zeros(shape, typecode='f', savespace=0, axes=None, attributes=None, id=None)**

An array of all zeros of the given length or shape.

The following table describes the MV non-constructor functions. With the exception of **argsort**, all functions return a transient variable.

Table 2.26 MV functions

#### Definition

**argsort(x, axis=-1, fill\_value=None)**

Return a Numeric array of indices for sorting along a given axis.

**asarray(data, typecode=None)**

Same as `cdms.createVariable(data, typecode, copy=0)`. This is a short way of ensuring that something is an instance of a variable of a given type before proceeding, as in

```
data = asarray(data)
```

Also see the variable **astype()** function.

Table 2.26 MV functions

**Definition****average(a, axis=0, weights=None)**

computes the average value of the non-masked elements of *x* along the selected axis. If *weights* is given, it must match the size and shape of *x*, and the value returned is:

$$\text{sum}(a * \text{weights}) / \text{sum}(\text{weights})$$

In computing these sums, elements that correspond to those that are masked in *x* or *weights* are ignored.

**choose(condition, t)**

has a result shaped like array *condition*. *t* must be a tuple of two arrays *t1* and *t2*. Each element of the result is the corresponding element of *t1* where *condition* is true, and the corresponding element of *t2* where *condition* is false. The result is masked where *condition* is masked or where the selected element is masked.

**concatenate(arrays, axis=0, axisid=None, axisattributes=None)**

Concatenate the arrays along the given axis. Give the extended axis the id and attributes provided - by default, those of the first array.

**count(a, axis=None)**

Count of the non-masked elements in *a*, or along a certain axis.

**isMaskedVariable(x)**

Return true if *x* is an instance of a variable.

**masked\_equal(x, value)**

*x* masked where *x* equals the scalar value For floating point value consider `masked_values(x, value)` instead.

**masked\_greater(x, value)**

*x* masked where  $x > \text{value}$

**masked\_greater\_equal(x, value)**

*x* masked where  $x \geq \text{value}$

Table 2.26 MV functions

**Definition****masked\_less(x, value)**

x masked where  $x < \text{value}$

**masked\_less\_equal(x, value)**

x masked where  $x \leq \text{value}$

**masked\_not\_equal(x, value)**

x masked where  $x \neq \text{value}$

**masked\_outside(x, v1, v2)**

x with mask of all values of x that are outside  $[\text{v1}, \text{v2}]$

**masked\_where(condition, x, copy=1)**

Return x as a variable masked where *condition* is true. Also masked where x or condition masked. *condition* is a masked array having the same shape as x.

**maximum(a, b=None)**

Compute the maximum valid values of x if y is None; with two arguments, return the element-wise larger of valid values, and mask the result where either x or y is masked.

**minimum(a, b=None)**

Compute the minimum valid values of x if y is None; with two arguments, return the element-wise smaller of valid values, and mask the result where either x or y is masked.

**outerproduct(a, b)**

Return a variable such that  $\text{result}[i, j] = a[i] * b[j]$ . The result will be masked where  $a[i]$  or  $b[j]$  is masked.

**power(a, b)**

$a^{*}b$

**product(a, axis=0, fill\_value=1)**

Product of elements along axis using *fill\_value* for missing elements.

Table 2.26 MV functions

**Definition****repeat(ar, repeats, axis=0)**

Return *ar* repeated *repeats* times along axis. *repeats* is a sequence of length *ar.shape[axis]* telling how many times to repeat each element.

**set\_default\_fill\_value(value\_type, value)**

Set the default fill value for *value\_type* to *value*. *value\_type* is a string: 'real', 'complex', 'character', 'integer', or 'object'. *value* should be a scalar or single-element array.

**sort(ar, axis=-1)**

Sort array *ar* elementwise along the specified axis. The corresponding axis in the result has dummy values.

**sum(a, axis=0, fill\_value=0)**

Sum of elements along a certain axis using *fill\_value* for missing.

**take(a, indices, axis=0)**

Return a selection of items from *a*. See the documentation in the Numeric manual.

**transpose(ar, axes=None)**

Perform a reordering of the axes of array *ar* depending on the tuple of indices *axes*; the default is to reverse the order of the axes.

**where(condition, x, y)**

*x* where *condition* is true, *y* otherwise.

---

## 2.10 RectGrid

A RectGrid is a two-dimensional, horizontal, rectilinear grid. A rectGrid can be defined in terms of a pair of axes, one longitude and one latitude. A two-dimensional, logical mask array may optionally be associated with a rectGrid.

Table 2.27 RectGrid Internal Attributes

| Type                | Name       | Definition                                   |
|---------------------|------------|----------------------------------------------|
| Dictionary          | attributes | External attribute dictionary.               |
| String              | id         | The grid identifier.                         |
| Dataset or CdmsFile | parent     | The dataset or file which contains the grid. |
| Tuple               | shape      | The shape of the grid, a 2-tuple.            |

Table 2.28 RectGrid Constructors

**cdms.createRectGrid(lat, lon, order, type="generic", mask=None)**

Create a grid not associated with a file or dataset.

See Table 2.2 on page 25.

**CdmsFile.createRectGrid(id, lat, lon, order, type="generic", mask=None)**

Create a grid associated with a file. See Table 2.12 on page 45.

**Dataset.createRectGrid(id, lat, lon, order, type="generic", mask=None)**

Create a grid associated with a dataset. See Table 2.24 on page 64.

**cdms.createGaussianGrid(nlats, xorigin=0.0, order="yx")**

See Table 2.2 on page 25.

**cdms.createGenericGrid(latArray, lonArray, latBounds=None, lonBounds=None, order="yx", mask=None)**

See Table 2.2 on page 18.

**Table 2.28 RectGrid Constructors**

---

**cdms.createGlobalMeanGrid(grid)**

See Table 2.2 on page 18.

**cdms.createRectGrid(lat, lon, order, type="generic", mask=None)**

See Table 2.2 on page 18.

**cdms.createUniformGrid(startLat, nlat, deltaLat, startLon, nlon, deltaLon, order="yx", mask=None)**

See Table 2.2 on page 18.

**cdms.createZonalGrid(grid)**

See Table 2.2 on page 18.

**Table 2.29 RectGrid Methods**

---

| Type | Definition                                                             |
|------|------------------------------------------------------------------------|
| Axis | <b>getAxis(Integer n)</b><br>Get the n-th axis.<br>n is either 0 or 1. |

Table 2.29 RectGrid Methods

| Type   | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tuple  | <p><b>getBounds()</b></p> <p>Get the grid boundary arrays.</p> <p>Returns a tuple (latitudeArray, longitudeArray), where latitudeArray is a Numeric array of latitude bounds, with shape (n,2), and longitudeArray is a similar array for longitude bounds.</p> <p>If no boundary arrays are explicitly defined (in the file or dataset), the result depends on the autoBounds mode (see <b>cdms.setAutoBounds</b>) and the grid classification mode (see <b>cdms.setClassifyGrids</b>). By default, autoBounds mode is enabled, in which case the boundary arrays are generated based on the type of grid. If disabled, the return value is (None,None).</p> <p>The grid classification mode specifies how the grid type is to be determined. By default, the grid type (Gaussian, uniform, etc.) is determined by calling <b>grid.classifyInFamily</b>. If the mode is 'off' <b>grid.getType</b> is used instead</p> |
| Axis   | <p><b>getLatitude()</b></p> <p>Get the latitude axis of this grid.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Axis   | <p><b>getLongitude()</b></p> <p>Get the longitude axis of this grid.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Array  | <p><b>getMask()</b></p> <p>Get the mask array of this grid, if any.</p> <p>Returns a 2-D Numeric array, having the same shape as the grid. If the mask is not explicitly defined, the return value is None.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| String | <p><b>getOrder()</b></p> <p>Get the grid ordering, either "yx" if latitude is the first axis, or "xy" if longitude is the first axis.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

Table 2.29 RectGrid Methods

| Type          | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String        | <p><b>getType()</b></p> <p>Get the grid type, either “gaussian”, “uniform”, “equalarea”, or “generic”.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| (Array,Array) | <p><b>getWeights()</b></p> <p>Get the normalized area weight arrays, as a tuple (latWeights, lonWeights). It is assumed that the latitude and longitude axes are defined in degrees.</p> <p>The latitude weights are defined as:</p> $\text{latWeights}[i] = 0.5 * \text{abs}(\text{sin}(\text{latBounds}[i+1]) - \text{sin}(\text{latBounds}[i]))$ <p>The longitude weights are defined as:</p> $\text{lonWeights}[i] = \text{abs}(\text{lonBounds}[i+1] - \text{lonBounds}[i]) / 360.0$ <p>For a global grid, the weight arrays are normalized such that the sum of the weights is 1.0</p> <p><b>Example:</b> Generate the 2-D weights array, such that weights[i,j] is the fractional area of grid zone [i,j].</p> <pre>from cdms import MV latwts, lonwts = grid.getWeights() weights = MV.outerproduct(latwts, lonwts)</pre> <p>Also see the function <b>area_weights</b> in module <b>pcmdi.weighting</b>.</p> |

Table 2.29 RectGrid Methods

| Type | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| None | <p><b>setBounds(latBounds, lonBounds, persistent=0)</b></p> <p>Set the grid boundaries.</p> <p><i>latBounds</i> is a NumPy array of shape (n,2), such that the boundaries of the kth axis value are [<i>latBounds</i>[k,0],<i>latBounds</i>[k,1]].</p> <p><i>lonBounds</i> is defined similarly for the longitude array.</p> <p>Note: By default, the boundaries are not written to the file or dataset containing the grid (if any). This allows bounds to be set on read-only files, for regridding. If the optional argument <i>persistent</i> is set to 1, the boundary array is written to the file.</p> |
| None | <p><b>setMask(mask, persistent=0)</b></p> <p>Set the grid mask. If <i>persistent</i>==1, the mask values are written to the associated file, if any. Otherwise, the mask is associated with the grid, but no I/O is generated.</p> <p><i>mask</i> is a two-dimensional, Boolean-valued Numeric array, having the same shape as the grid.</p>                                                                                                                                                                                                                                                                  |
| None | <p><b>setType(gridtype)</b></p> <p>Set the grid type.</p> <p><i>gridtype</i> is one of “gaussian”, “uniform”, “equalarea”, or “generic”.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**Table 2.29 RectGrid Methods**

---

| Type     | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RectGrid | <p><b>subGrid((latStart,latStop),(lonStart,lonStop))</b></p> <p>Create a new grid, with latitude index range [latStart : latStop] and longitude index range [lonStart : lonStop]. Either index range can also be specified as None, indicating that the entire range of the latitude or longitude is used. For example,</p> <pre>newgrid = oldgrid.subGrid(None, (lonStart, lonStop))</pre> <p>creates <code>newgrid</code> corresponding to all latitudes, and index range [lonStart:lonStop] of <code>oldgrid</code>.</p> <p>If a mask is defined, the subgrid also has a mask corresponding to the index ranges.</p> <p>Note: The result grid is not associated with any file or dataset.</p> |

Table 2.29 RectGrid Methods

| Type     | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RectGrid | <p data-bbox="654 366 1153 392"><b>subGridRegion(latInterval, lonInterval)</b></p> <p data-bbox="654 413 1260 470">Create a new grid corresponding to the coordinate region defined by <i>latInterval</i>, <i>lonInterval</i>.</p> <p data-bbox="654 513 1296 574"><i>latInterval</i> and <i>lonInterval</i> are the coordinate intervals for latitude and longitude, respectively.</p> <p data-bbox="654 618 1215 644">Each interval is a tuple having one of the forms:</p> <p data-bbox="662 687 896 817">(x,y)<br/>(x,y,indicator)<br/>(x,y,indicator,cycle)<br/>None</p> <p data-bbox="654 861 1265 921">where <i>x</i> and <i>y</i> are coordinates indicating the interval [x,y], and:</p> <p data-bbox="654 965 1303 1095"><i>indicator</i> is a two-character string, where the first character is 'c' if the interval is closed on the left, 'o' if open, and the second character has the same meaning for the right-hand point. (Default: 'co')</p> <p data-bbox="654 1138 1315 1269">If <i>cycle</i> is specified, the axis is treated as circular with the given cycle value. By default, if <code>grid.isCircular()</code> is true, the axis is treated as circular with a default value of 360.0.</p> <p data-bbox="654 1312 1310 1373">An interval of None returns the full index interval of the axis.</p> <p data-bbox="654 1399 1300 1459">If a mask is defined, the subgrid also has a mask corresponding to the index ranges.</p> <p data-bbox="654 1468 1315 1494">Note: The result grid is not associated with any file or dataset.</p> |

**Table 2.29 RectGrid Methods**

---

| Type     | Definition                                                                                                                                                                |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RectGrid | <b>transpose()</b><br>Create a new grid, with axis order reversed. The grid mask is also transposed.<br>Note: The result grid is not associated with any file or dataset. |

---

## 2.11 Variable

A Variable is a multidimensional data object, consisting of:

- a multidimensional data array, possibly masked,
- a collection of attributes
- a domain, an ordered tuple of Axis and/or Grid objects.

A Variable which is contained in a Dataset or CdmsFile is called a *persistent* variable. Setting a slice of a persistent Variable writes data to the Dataset or file, and referencing a Variable slice reads data from the Dataset. Variables may also be *transient*, not associated with a Dataset or CdmsFile.

Variables support arithmetic operations. The basic Python operators are +, -, \*, /, \*\*, abs, and sqrt, together with the operations defined in the **MV** module. The result of an arithmetic operation is a transient variable, that is, the axis information is transferred to the result.

The methods subRegion and subSlice return transient variables. In addition, a transient variable may be created with the **cdms.createVariable** method. The vcs and regrid module methods take advantage of the attribute, domain, and mask information in a transient variable.

### 2.11.1 cu interface support

The cu module defines the original CDAT I/O interface. It is maintained for backward compatibility. As of CDMS V3, CDMS variables support the cu Slab interface defined in “cu Module” on page 161.

**Table 2.30 Variable Internal Attributes**

| Type                | Name         | Definition                                                                                                                          |
|---------------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Dictionary          | attributes   | External attribute dictionary.                                                                                                      |
| String              | id           | Variable identifier.                                                                                                                |
| String              | name_in_file | The name of the variable in the file or files which represent the dataset. If different from <b>id</b> , the variable is ‘aliased’. |
| Dataset or CdmsFile | parent       | The dataset or file which contains the variable.                                                                                    |
| Tuple               | shape        | The length of each axis of the variable.                                                                                            |

**Table 2.31 Variable Constructors**

---

**Dataset.createVariable(String id, String datatype, List axes)**

Create a Variable in a Dataset. **This function is not yet implemented.**

**Table 2.31 Variable Constructors**

---

**CdmsFile.createVariable(String id, String datatype, List axesOrGrids)**

Create a Variable in a CdmsFile.

*id* is the name of the variable.

*datatype* is the MA or Numeric typecode, for example, MA.Float.

*axesOrGrids* is a list of Axis and/or Grid objects, on which the variable is defined. Specifying a rectilinear grid is equivalent to listing the grid latitude and longitude axes, in the order defined for the grid. Note: this argument can either be a list or a tuple. If the tuple form is used, and there is only one element, it must have a following comma, e.g.: (axisobj,).

Table 2.31 Variable Constructors

**`cdms.createVariable(array, typecode=None, copy=0, savespace=0, mask=None, fill_value=None, grid=None, axes=None, attributes=None, id=None)`**

Create a transient variable, not associated with a file or dataset.

*array* is the data values: a Variable, masked array, or Numeric array.

*typecode* is the MA typecode of the array. Defaults to the typecode of *array*.

*copy* is an integer flag: if 1, the variable is created with a copy of the array, if 0 the variable data is shared with *array*.

*savespace* is an integer flag: if set to 1, internal Numeric operations will attempt to avoid silent upcasting.

*mask* is an array of integers with value 0 or 1, having the same shape as *array*. *array* elements with a corresponding mask value of 1 are considered 'invalid', and are not used for subsequent Numeric operations. The default mask is obtained from array if present, otherwise is None.

*fill\_value* is the missing value flag. The default is obtained from *array* if possible, otherwise is set to 1.0e20 for floating point variables, 0 for integer-valued variables.

*grid* is a rectilinear grid object.

*axes* is a tuple of axis objects. By default the axes are obtained from *array* if present. Otherwise for a dimension of length *n*, the default axis has values [0., 1., ..., double(*n*)].

*attributes* is a dictionary of attribute values. The dictionary keys must be strings. By default the dictionary is obtained from *array* if present, otherwise is empty.

*id* is the string identifier of the variable. By default the *id* is obtained from array if possible, otherwise is set to 'variable\_n' for some integer *n*.

Table 2.32 Variable Methods

| Type     | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Variable | <p><b>tvar = var[ i:j, m:n]</b></p> <p>Read a slice of data from the file or dataset, resulting in a transient variable. Singleton dimensions are ‘squeezed’ out. Data is returned in the physical ordering defined in the dataset. The forms of the slice operator are listed in Table 2.33 on page 94.</p> <p><b>var[ i:j, m:n] = array</b></p> <p>Write a slice of data to the external dataset. The forms of the slice operator are listed in Table 2.21 on page 32. (Variables in CdmsFiles only)</p> |
| Variable | <p><b>tvar = var(selector)</b></p> <p>Calling a variable as a function reads the region of data defined by the <i>selector</i>. The result is a transient variable, unless <b>raw=1</b> keyword is specified. ‘See “Selectors” on page 95.</p>                                                                                                                                                                                                                                                             |
| None     | <p><b>assignValue(Array ar)</b></p> <p>Write the entire data array. Equivalent to <code>var[:] = ar</code>. (Variables in CdmsFiles only).</p>                                                                                                                                                                                                                                                                                                                                                             |
| Variable | <p><b>astype(typecode)</b></p> <p>Cast the variable to a new datatype. Typecodes are as for MV, MA, and Numeric modules.</p>                                                                                                                                                                                                                                                                                                                                                                               |
| Variable | <p><b>clone(copyData=1)</b></p> <p>Return a copy of a transient variable.</p> <p>If <code>copyData</code> is 1 (the default) the variable data is copied as well. If <code>copyData</code> is 0, the result transient variable shares the original transient variable’s data array.</p>                                                                                                                                                                                                                    |

Table 2.32 Variable Methods

| Type               | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Transient Variable | <p><b>crossSectionRegrid(newLevel, newLatitude, method="log", missing=None, order=None)</b></p> <p>Return a lat/level vertical cross-section regridded to a new set of latitudes <i>newLatitude</i> and levels <i>newLevel</i>. The variable should be a function of latitude, level, and (optionally) time. <i>newLevel</i> is an axis of the result pressure levels. <i>newLatitude</i> is an axis of the result latitudes.</p> <p><i>method</i> is optional, either "log" to interpolate in the log of pressure (default), or "linear" for linear interpolation.</p> <p><i>missing</i> is a missing data value. The default is <code>var.getMissing()</code></p> <p><i>order</i> is an order string such as "tzy" or "zy". The default is <code>var.getOrder()</code></p> <p>See also: <b>regrid</b>, <b>pressureRegrid</b>.</p> |
| Axis               | <p><b>getAxis(n)</b></p> <p>Get the n-th axis.</p> <p><i>n</i> is an integer.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| List               | <p><b>getAxisIds()</b></p> <p>Get a list of axis identifiers.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Integer            | <p><b>getAxisIndex(axis_spec)</b></p> <p>Return the index of the axis specified by <i>axis_spec</i>. Return -1 if no match.</p> <p><i>axis_spec</i> is a specification as defined for <b>getAxisList</b></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

Table 2.32 Variable Methods

| Type | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| List | <p data-bbox="654 366 1272 392"><b>getAxisList(axes=None, omit=None, order=None)</b></p> <p data-bbox="654 413 1286 465">Get an ordered list of axis objects in the domain of the variable..</p> <p data-bbox="654 487 1315 574">If <i>axes</i> is not None, include only certain axes. Otherwise <i>axes</i> is a list of specifications as described below. Axes are returned in the order specified unless the <i>order</i> keyword is given.</p> <p data-bbox="654 597 1308 683">If <i>omit</i> is not None, omit those specified by an integer dimension number. Otherwise <i>omit</i> is a list of specifications as described below.</p> <p data-bbox="654 706 1243 732"><i>order</i> is an optional string determining the output order.</p> <p data-bbox="654 755 1286 807">Specifications for the axes or omit keywords are a list, each element having one of the following forms:</p> <ul data-bbox="654 829 1300 1194" style="list-style-type: none"> <li data-bbox="654 829 1125 855">• an integer dimension index, starting at 0.</li> <li data-bbox="654 878 1272 947">• a string representing an axis id or one of the strings 'time', 'latitude', 'lat', 'longitude', 'lon', 'lev' or 'level'.</li> <li data-bbox="654 970 1300 1022">• a function that takes an axis as an argument and returns a value. If the value returned is true, the axis matches.</li> <li data-bbox="654 1045 1286 1194">• an axis object; will match if it is the same object as axis.<br/> <i>order</i> can be a string containing the characters <i>t,x,y,z</i>, or -. If a dash ('-') is given, any elements of the result not chosen otherwise are filled in from left to right with remaining candidates.</li> </ul> |
| List | <p data-bbox="654 1229 1172 1289"><b>getAxisListIndex(axes=None, omit=None, order=None)</b></p> <p data-bbox="654 1312 1293 1364">Return a list of indices of axis objects. Arguments are as for <b>getAxisList</b>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

Table 2.32 Variable Methods

| Type    | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| List    | <p><b>getDomain()</b></p> <p>Get the domain. Each element of the list is itself a tuple of the form</p> <p><code>(axis, start, length, true_length)</code></p> <p>where <i>axis</i> is an axis object, <i>start</i> is the start index of the domain relative to the axis object, <i>length</i> is the length of the axis, and <i>true_length</i> is the actual number of (defined) points in the domain.</p> <p>See also: <b>getAxisList</b>.</p> |
| Grid    | <p><b>getGrid()</b></p> <p>Return the associated grid, or None if the variable is not gridded.</p>                                                                                                                                                                                                                                                                                                                                                 |
| Axis    | <p><b>getLatitude()</b></p> <p>Get the latitude axis, or None if not found.</p>                                                                                                                                                                                                                                                                                                                                                                    |
| Axis    | <p><b>getLevel()</b></p> <p>Get the vertical level axis, or None if not found.</p>                                                                                                                                                                                                                                                                                                                                                                 |
| Axis    | <p><b>getLongitude()</b></p> <p>Get the longitude axis, or None if not found.</p>                                                                                                                                                                                                                                                                                                                                                                  |
| Various | <p><b>getMissing()</b></p> <p>Get the missing data value, or None if not found.</p>                                                                                                                                                                                                                                                                                                                                                                |

Table 2.32 Variable Methods

| Type   | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String | <p data-bbox="654 366 791 392"><b>getOrder()</b></p> <p data-bbox="654 413 1319 565">Get the order string of a spatio-temporal variable. The order string specifies the physical ordering of the data. It is a string of characters with length equal to the rank of the variable, indicating the order of the variable’s time, level, latitude, and/or longitude axes. Each character is one of:</p> <p data-bbox="654 586 1022 739">           ‘t’: time<br/>           ‘z’: vertical level<br/>           ‘y’: latitude<br/>           ‘x’: longitude<br/>           ‘-’: the axis is not spatio-temporal.         </p> <p data-bbox="654 760 1319 812"><b>Example:</b> A variable with ordering “tzyx” is 4-dimensional, where the ordering of axes is (time, level, latitude, longitude).</p> <p data-bbox="654 833 1319 887">Note: The order string is of the form required for the <i>order</i> argument of a regridding function.</p> |
| List   | <p data-bbox="654 921 905 947"><b>getPaths(*intervals)</b></p> <p data-bbox="654 968 1319 1020">Get the file paths associated with the index region specified by intervals.</p> <p data-bbox="654 1041 1319 1133"><i>intervals</i> is a list of scalars, 2-tuples representing [i,j], slices, and/or Ellipses. If no argument(s) are present, all file paths associated with the variable are returned.</p> <p data-bbox="654 1154 1319 1307">Returns a list of tuples of the form (path,slicetuple), where <i>path</i> is the path of a file, and <i>slicetuple</i> is itself a tuple of slices, of the same length as the rank of the variable, representing the portion of the variable in the file corresponding to <i>intervals</i>.</p> <p data-bbox="654 1328 1258 1352">Note: This function is not defined for transient variables.</p>                                                                                               |

Table 2.32 Variable Methods

| Type               | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String             | <p><b>getTemplate()</b></p> <p>Get the file template associated with this variable. If no template is associated with the variable, the dataset template is returned.</p> <p>Note: This function is not defined for transient variables.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Axis               | <p><b>getTime()</b></p> <p>Get the time axis, or None if not found.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Integer            | <p><b>len(var)</b></p> <p>The length of the first dimension of the variable. If the variable is zero-dimensional (scalar), a length of 0 is returned.</p> <p>Note: <b>size()</b> returns the total number of elements.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Transient Variable | <p><b>pressureRegrid (newLevel, method="log", missing=None, order=None)</b></p> <p>Return the variable regridded to a new set of pressure levels <i>newLevel</i>. The variable must be a function of latitude, longitude, pressure level, and (optionally) time.</p> <p><i>newLevel</i> is an axis of the result pressure levels.</p> <p><i>method</i> is optional, either "log" to interpolate in the log of pressure (default), or "linear" for linear interpolation.</p> <p><i>missing</i> is a missing data value. The default is <code>var.getMissing()</code></p> <p><i>order</i> is an order string such as "tzyx" or "zyx". The default is <code>var.getOrder()</code></p> <p>See also: <b>regrid</b>, <b>crossSectionRegrid</b>.</p> |
| Integer            | <p><b>rank()</b></p> <p>The number of dimensions of the variable.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Table 2.32 Variable Methods

| Type               | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Transient Variable | <p><b>regrid (togrid, missing=None, order=None, mask=None)</b></p> <p>Return the variable regridded to the horizontal grid <i>togrid</i>. <i>missing</i> is a Float specifying the missing data value. The default is 1.0e20.</p> <p><i>order</i> is a string indicating the order of dimensions of the array. It has the form returned from <b>variable.getOrder()</b>. For example, the string “tzyx” indicates that the dimension order of <i>array</i> is (time, level, latitude, longitude). If unspecified, the function assumes that the last two dimensions of <i>array</i> match the input grid.</p> <p><i>mask</i> is a Numeric array, of datatype Integer or Float, consisting of ones and zeros. A value of 0 or 0.0 indicates that the corresponding data value is to be ignored for purposes of regridding. If <i>mask</i> is two-dimensional of the same shape as the input grid, it overrides the mask of the input grid. If the mask has more than two dimensions, it must have the same shape as <i>array</i>. In this case, the <i>missing</i> data value is also ignored. Such an n-dimensional mask is useful if the pattern of missing data varies with level (e.g., ocean data) or time. Note: If neither <i>missing</i> or <i>mask</i> is set, the default mask is obtained from the mask of the array if any.</p> <p>See also: <b>crossSectionRegrid</b>, <b>pressureRegrid</b>.</p> |
| None               | <p><b>setAxis(n, axis)</b></p> <p>Set the n-th axis (0-origin index) of to a copy of <i>axis</i>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| None               | <p><b>setAxisList(axislist)</b></p> <p>Set all axes of the variable. <i>axislist</i> is a list of axis objects.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| None               | <p><b>setMissing(value)</b></p> <p>Set the missing value.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

Table 2.32 Variable Methods

| Type     | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Integer  | <p><b>size()</b></p> <p>Number of elements of the variable.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Variable | <p><b>subRegion(*region, time=None, level=None, latitude=None, longitude=None, squeeze=0, raw=0)</b></p> <p>Read a coordinate region of data, returning a transient variable. A region is a hyperrectangle in coordinate space.</p> <p><i>region</i> is an argument list, each item of which specifies an interval of a coordinate axis. The intervals are listed in the order of the variable axes. If trailing dimensions are omitted, all values of those dimensions are retrieved. If an axis is circular (<b>axis.isCircular()</b> is true) or cycle is specified (see below), then data will be read with wraparound in that dimension. Only one axis may be read with wraparound. A coordinate interval has one of the forms listed in Table 2.34 on page 94. Also see <b>axis.mapIntervalExt</b>.</p> <p>The optional keyword arguments <i>time</i>, <i>level</i>, <i>latitude</i>, and <i>longitude</i> may also be used to specify the dimension for which the interval applies. This is particularly useful if the order of dimensions is not known in advance. An exception is raised if a keyword argument conflicts with a positional <i>region</i> argument.</p> <p>The optional keyword argument <i>squeeze</i> determines whether or not the shape of the returned array contains dimensions whose length is 1; by default this argument is 0, and such dimensions are not 'squeezed out'.</p> <p>The optional keyword argument <i>raw</i> specifies whether the return object is a variable or a masked array. By default, a transient variable is returned, having the axes and attributes corresponding to the region read. If <i>raw</i>=1, an MA masked array is returned, equivalent to the transient variable without the axis and attribute information.</p> |

Table 2.32 Variable Methods

| Type     | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Variable | <p><b>subSlice(*specs, time=None, level=None, latitude=None, longitude=None, squeeze=0, raw=0)</b></p> <p>Read a slice of data, returning a transient variable. This is a functional form of the slice operator [] with the squeeze option turned off.</p> <p><i>specs</i> is an argument list, each element of which specifies a slice of the corresponding dimension. There can be zero or more positional arguments, each of the form:</p> <ul style="list-style-type: none"> <li>(a) a single integer <i>n</i>, meaning slice(<i>n</i>, <i>n</i>+1)</li> <li>(b) an instance of the slice class</li> <li>(c) a tuple, which will be used as arguments to create a slice</li> <li>(d) ':', which means a slice covering that entire dimension</li> <li>(e) Ellipsis (...), which means to fill the slice list with ':' leaving only enough room at the end for the remaining positional arguments</li> <li>(f) a Python slice object, of the form slice(<i>i</i>,<i>j</i>,<i>k</i>)</li> </ul> <p>If there are fewer slices than corresponding dimensions, all values of the trailing dimensions are read.</p> <p>The keyword arguments are defined as in <b>subRegion</b>.</p> <p>There must be no conflict between the positional arguments and the keywords.</p> <p>In (a)-(c) and (f), negative numbers are treated as offsets from the end of that dimension, as in normal Python indexing.</p> |
| String   | <p><b>typecode()</b></p> <p>The Numeric datatype identifier.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Example:** Get a region of data.

Variable *ta* is a function of (time, latitude, longitude). Read data corresponding to all times, latitudes -45.0 up to but not including +45.0, longitudes 0.0 through and including longitude 180.0:

```
data = ta.subRegion(':', (-45.0,45.0,'co'), (0.0, 180.0))
```

or equivalently:

```
data = ta.subRegion(latitude=(-45.0,45.0,'co'), longitude=(0.0,
180.0))
```

Read all data for March, 1980:

```
data = ta.subRegion(time=('1980-3','1980-4','co'))
```

**Table 2.33 Variable Slice Operators**

|             |                                                             |
|-------------|-------------------------------------------------------------|
| [i]         | The ith element, zero-origin indexing.                      |
| [i:j]       | The ith element through, but not including, element j       |
| [i:]        | The ith element through the end                             |
| [:j]        | The beginning element through, but not including, element j |
| [:]         | The entire array                                            |
| [i:j:k]     | Every kth element                                           |
| [i:j, m:n]  | Multidimensional slice                                      |
| [i, ..., m] | (Ellipsis) All dimensions between those specified.          |
| [-1]        | Negative indices 'wrap around'. -1 is the last element.     |

**Table 2.34 Index and Coordinate Intervals**

| Interval | Definition                                                                                                                                                                                                                                                       | Example                                                                 |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| x        | single point, such that axis[i]==x<br><br>In general x is a scalar. If the axis is a time axis, x may also be a cftime relative time type, component time type, or string of the form 'yyyy-mm-dd hh:mi:ss' (where trailing fields of the string may be omitted. | 180.0<br><br>cdtime.rel-time(48,"hours since 1980-1")<br><br>'1980-1-3' |
| (x,y)    | indices i such that x <= axis[i] <= y                                                                                                                                                                                                                            | (-180,180)                                                              |

Table 2.34 Index and Coordinate Intervals

| Interval         | Definition                                                                                                                                                                                                                                                                                                | Example                                                          |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| (x,y,'co')       | $x \leq \text{axis}[i] < y$<br>The third item is defined as in mapInterval.                                                                                                                                                                                                                               | (-90,90,'cc')                                                    |
| (x,y,'co',cycle) | $x \leq \text{axis}[i] < y$ , with wraparound<br>Note: It is not necessary to specify the cycle of a circular longitude axis, that is, for which axis.isCircular() is true.                                                                                                                               | ( 180, 180, 'co', 360.0)                                         |
| slice(i,j,k)     | slice object, equivalent to i:j:k in a slice operator. Refers to the indices i, i+k, i+2k, ... up to but not including index j. If i is not specified or is None it defaults to 0. If j is not specified or is None it defaults to the length of the axis. The stride k defaults to 1. k may be negative. | slice(1,10)<br>slice(,-1)<br>reverses the direction of the axis. |
| ':'              | all axis values of one dimension                                                                                                                                                                                                                                                                          |                                                                  |
| Ellipsis         | all values of all intermediate axes                                                                                                                                                                                                                                                                       |                                                                  |

### 2.11.2 Selectors

A *selector* is a specification of a region of data to be selected from a variable. For example, the statement

```
x = v(time='1979-1-1', level=(1000.0,100.0))
```

means 'select the values of variable v for time '1979-1-1' and levels 1000.0 to 100.0 inclusive, setting x to the result.' Selectors are generally used to represent regions of space and time.

The form for using a selector is

```
result = v(s)
```

where *v* is a variable and *s* is the selector. An equivalent form is

```
result = f('varid', s)
```

where *f* is a file or dataset, and 'varid' is the string ID of a variable.

A selector consists of a list of *selector components*. For example, the selector

```
time='1979-1-1', level=(1000.0,100.0)
```

has two components: `time='1979-1-1'`, and `level=(1000.0,100.0)`. This illustrates that selector components can be defined with keywords, using the form:

```
keyword=value
```

Note that for the keywords `time`, `level`, `latitude`, and `longitude`, the selector can be used with any variable. If the corresponding axis is not found, the selector component is ignored. This is very useful for writing general purpose scripts. The `required` keyword overrides this behavior. These keywords take values that are coordinate ranges or index ranges as defined in Table 2.34 on page 94.

The following keywords are available:

**Table 2.35 Selector keywords**

| Keyword                | Description                                                                  | Value                                                                |
|------------------------|------------------------------------------------------------------------------|----------------------------------------------------------------------|
| <code>axisid</code>    | Restrict the axis with ID <code>axisid</code> to a value or range of values. | See Table 2.34 on page 94                                            |
| <code>grid</code>      | Regrid the result to the grid.                                               | Grid object                                                          |
| <code>latitude</code>  | Restrict latitude values to a value or range. Short form: <code>lat</code>   | See Table 2.34 on page 94                                            |
| <code>level</code>     | Restrict vertical levels to a value or range. Short form: <code>lev</code>   | See Table 2.34 on page 94                                            |
| <code>longitude</code> | Restrict longitude values to a value or range. Short form: <code>lon</code>  | See Table 2.34 on page 94                                            |
| <code>order</code>     | Reorder the result.                                                          | Order string, e.g., “tzyx”                                           |
| <code>raw</code>       | Return a masked array (MA.array) rather than a transient variable.           | 0: return a transient variable (default); =1: return a masked array. |

Table 2.35 Selector keywords

| Keyword               | Description                                  | Value                                                                    |
|-----------------------|----------------------------------------------|--------------------------------------------------------------------------|
| <code>required</code> | Require that the axis IDs be present.        | List of axis identifiers.                                                |
| <code>squeeze</code>  | Remove singleton dimensions from the result. | 0: leave singleton dimensions (default); 1: remove singleton dimensions. |
| <code>time</code>     | Restrict time values to a value or range.    | See Table 2.34 on page 94                                                |

Another form of selector components is the *positional* form, where the component order corresponds to the axis order of a variable. For example:

```
x9 = hus(('1979-1-1', '1979-2-1'), 1000.0)
```

reads data for the range ('1979-1-1', '1979-2-1') of the first axis, and coordinate value 1000.0 of the second axis. Non-keyword arguments of the form(s) listed in Table 2.34 on page 94 are treated as positional. Such selectors are more concise, but not as general or flexible as the other types described in this section.

Selectors are objects in their own right. This means that a selector can be defined and reused, independent of a particular variable. Selectors are constructed using the `cdms.selectors.Selector` class. The constructor takes an argument list of selector components. For example:

```
from cdms.selectors import Selector
sel = Selector(time=('1979-1-1', '1979-2-1'), level=1000.)
x1 = v1(sel)
x2 = v2(sel)
```

For convenience CDMS provides several predefined selectors, which can be used directly or can be combined into more complex selectors. The selectors `time`, `level`, `latitude`, `longitude`, and `required` are equivalent to their keyword counterparts. For example:

```
from cdms import time, level
x = hus(time('1979-1-1', '1979-2-1'), level(1000.))
```

and

```
x = hus(time=('1979-1-1','1979-2-1'), level=1000.)
```

are equivalent. Additionally, the predefined selectors `latitudeslice`, `longitudeslice`, `levelslice`, and `timeslice` take arguments (startindex, stopindex[, stride]):

```
from cdms import timeslice, levelslice
x = v(timeslice(0,2), levelslice(16,17))
```

Finally, a collection of selectors is defined in module `cdutil.region`:

```
from cdutil.region import *
NH=NorthernHemisphere=domain(latitude=(0.,90.))
SH=SouthernHemisphere=domain(latitude=(-90.,0.))
Tropics=domain(latitude=(-23.4,23.4))
NPZ=AZ=ArcticZone=domain(latitude=(66.6,90.))
SPZ=AAZ=AntarcticZone=domain(latitude=(-90.,-66.6))
```

Selectors can be combined using the `&` operator, or by refining them in the call:

```
from cdms.selectors import Selector
from cdms import level
sel2 = Selector(time=('1979-1-1','1979-2-1'))
sel3 = sel2 & level(1000.0)
x1 = hus(sel3)
x2 = hus(sel2, level=1000.0)
```

### 2.11.3 Selector examples

CDMS provides a variety of ways to select or slice data. In the following examples, variable ‘hus’ is contained in file `sample.nc`, and is a function of (time, level, latitude, longitude). Time values are monthly starting at 1979-1-1. There are 17 levels, the last level being 1000.0. The name of the vertical level axis is ‘plev’. All the examples select the first two times and the last level. The last two examples remove the singleton level dimension from the result array.

```
import cdms

f = cdms.open('sample.nc')
hus = f.variables['hus']

Keyword selection
x = hus(time=('1979-1-1','1979-2-1'), level=1000.)
```

```
Interval indicator (see mapIntervalExt)
x = hus(time=('1979-1-1','1979-3-1','co'), level=1000.)

Axis ID (plev) as a keyword
x = hus(time=('1979-1-1','1979-2-1'), plev=1000.)

Positional
x9 = hus(('1979-1-1','1979-2-1'),1000.0)

Predefined selectors
from cdms import time, level
x = hus(time('1979-1-1','1979-2-1'), level(1000.))

from cdms import timeslice, levelslice
x = hus(timeslice(0,2), levelslice(16,17))

Call file as a function
x = f('hus', time=('1979-1-1','1979-2-1'), level=1000.)

Python slices
x = hus(time=slice(0,2), level=slice(16,17))

Selector objects
from cdms.selectors import Selector
sel = Selector(time=('1979-1-1','1979-2-1'), level=1000.)
x = hus(sel)

sel2 = Selector(time=('1979-1-1','1979-2-1'))
sel3 = sel2 & level(1000.0)
x = hus(sel3)
x = hus(sel2, level=1000.0)

Squeeze singleton dimension (level)
x = hus[0:2,16]
x = hus(time=('1979-1-1','1979-2-1'), level=1000., squeeze=1)

f.close()
```

---

## 2.12 Examples

In this example, two datasets are opened, containing surface air temperature ('tas') and upper-air temperature ('ta') respectively. Surface air temperature is a function of (time, latitude, longitude). Upper-air temperature is a function of (time, level, latitude, longitude). Time is assumed to

have a relative representation in the datasets (e.g., with units “months since basetime”).

Data is extracted from both datasets for January of the first input year through December of the second input year. For each time and level, three quantities are calculated: slope, variance, and correlation. The results are written to a netCDF file. For brevity, the functions `corrCoefSlope` and `removeSeasonalCycle` are omitted.

```

1 import cdms
 from cdms import MV

 # Calculate variance, slope, and correlation of
 # surface air temperature with upper air temperature
 # by level, and save to a netCDF file. 'pathTa' is the location of
 # the file containing ta, 'pathTas' is the file with contains tas.
 # Data is extracted from January of year1 through December of year2.
 def ccSlopeVarianceBySeasonFiltNet(pathTa,pathTas,month1,month2):

 # Open the files for ta and tas
 fta = cdms.open(pathTa)
 ftas = cdms.open(pathTas)

2 # Get upper air temperature
 taObj = fta['ta']
 levs = taObj.getLevel()

3 # Get the surface temperature for the closed interval [time1,time2]
 tas = ftas('tas', time=(month1,month2,'cc'))

 # Allocate result arrays
 newaxes = taObj.getAxisList(omit='time')
 newshape = tuple([len(a) for a in newaxes])

4 cc = MV.zeros(newshape, typecode=MV.Float, axes=newaxes, id='correlation')
 b = MV.zeros(newshape, typecode=MV.Float, axes=newaxes, id='slope')
 v = MV.zeros(newshape, typecode=MV.Float, axes=newaxes, id='variance')

 # Remove seasonal cycle from surface air temperature
 tas = removeSeasonalCycle(tas)

 # For each level of air temperature, remove seasonal cycle
 # from upper air temperature, and calculate statistics
5 for ilev in range(len(levs)):
 ta = taObj.subRegion(time=(month1,month2,'cc'), \
 level=slice(ilev, ilev+1), squeeze=1)
 ta = removeSeasonalCycle(ta)
 cc[ilev], b[ilev] = corrCoefSlope(tas ,ta)
 v[ilev] = MV.sum(ta**2)/(1.0*ta.shape[0])

6 # Write slope, correlation, and variance variables
 f = cdms.open('CC_B_V_ALL.nc','w')
 f.title = 'filtered'
 f.write(b)
 f.write(cc)
 f.write(v)
 f.close()

```

7

```
if __name__ == '__main__':
 pathTa = '/pcmdi/cdms/sample/ccmSample_ta.xml'
 pathTas = '/pcmdi/cdms/sample/ccmSample_tas.xml'
 # Process Jan80 through Dec81
 ccSlopeVarianceBySeasonFiltNet(pathTa,pathTas,'80-1','81-12')
```

Notes:

1. Two modules are imported, **cdms**, and **MV**. **MV** implements arithmetic functions.
2. **taObj** is a file (persistent) variable. At this point, no data has actually been read. This happens when the file variable is sliced, or when the **subRegion** function is called. **levs** is an axis.
3. Calling the file like a function reads data for the given variable and time range. Note that **month1** and **month2** are time strings.
4. In contrast to **taObj**, the variables **cc**, **b**, and **v** are transient variables, not associated with a file. The assigned names are used when the variables are written.
5. Another way to read data is to call **subRegion** on the variable. The **squeeze** option removes singleton axes, in this case the level axis.
6. Write the data. Axis information is written automatically.
7. This is the main routine of the script. **pathTa** and **pathTas** pathnames. Data is processed from January 1980 through December 1981.

In the next example, the pointwise variance of a variable over time is calculated, for all times in a dataset. The name of the dataset and variable are entered, then the variance is calculated and plotted via the **vcs** module.

8

```
#!/usr/bin/env python
#
Calculates gridpoint total variance
from an array of interest
#
import cdms
from MV import *

Wait for return in an interactive window
def pause():
 print 'Hit return to continue: ',
 line = sys.stdin.readline()

Calculate pointwise variance of variable over time
Returns the variance and the number of points
for which the data is defined, for each grid point
def calcVar(x):
```

```

Check that the first axis is a time axis
firstaxis = x.getAxis(0)
if not firstaxis.isTime():
 raise 'First axis is not time, variable:', x.id

n = count(x,0)
sumxx = sum(x*x)
sumx = sum(x)
variance = (n*sumxx - (sumx * sumx))/(n * (n-1.))

return variance,n

if __name__=='__main__':
 import vcs, sys

 print 'Enter dataset path [/pcmdi/cdms/obs/erbs_mo.xml]: ',
 path = string.strip(sys.stdin.readline())
 if path=='': path='/pcmdi/cdms/obs/erbs_mo.xml'

 # Open the dataset
 dataset = cdms.open(path)

 # Select a variable from the dataset
 print 'Variables in file:',path
 varnames = dataset.variables.keys()
 varnames.sort()
 for varname in varnames:
 var = dataset.variables[varname]
 if hasattr(var,'long_name'):
 long_name = var.long_name
 elif hasattr(var,'title'):
 long_name = var.title
 else:
 long_name = '?'
 print '%-10s: %s'%(varname,long_name)
 print 'Select a variable: ',
 varname = string.strip(sys.stdin.readline())
 var = dataset(varname)
 dataset.close()

 # Calculate variance, count, and set attributes
 variance,n = calcVar(var)
 variance.id = 'variance_%s'%var.id
 n.id = 'count_%s'%var.id
 if hasattr(var,'units'):
 variance.units = '%s^2'%var.units

 # Plot variance
 w=vcs.init()
 w.plot(variance)
 pause()
 w.clear()
 w.plot(n)
 pause()
 w.clear()

```

The result of running this script is as follows:

```
% calcVar.py
Enter dataset path [/pcmdi/cdms/sample/obs/erbs_mo.xml]:
Variables in file: /pcmdi/cdms/sample/obs/erbs_mo.xml
albt : Albedo TOA [%]
albtcs : Albedo TOA clear sky [%]
rlcrft : LW Cloud Radiation Forcing TOA [W/m^2]
rlut : LW radiation TOA (OLR) [W/m^2]
rlutcs : LW radiation upward TOA clear sky [W/m^2]
rscrft : SW Cloud Radiation Forcing TOA [W/m^2]
rsdt : SW radiation downward TOA [W/m^2]
rsut : SW radiation upward TOA [W/m^2]
rsutcs : SW radiation upward TOA clear sky [W/m^2]
Select a variable: albt

<The variance is plotted>

Hit return to continue:

<The number of points is plotted>
```

### Notes:

8. `n = count(x, 0)` returns the pointwise number of valid values, summing across axis 0, the first axis. `count` is an MV function.
9. `dataset` is a `Dataset` or `CdmsFile` object, depending on whether a `.xml` or `.nc` pathname is entered. `dataset.variables` is a dictionary mapping variable name to file variable.
10. `var` is a transient variable.
11. Plot the variance and count variables. Spatial longitude and latitude information are carried with the computations, so the continents are plotted correctly.



---

### 3.1 Time types

The `cdtime` module implements the CDMS time types, methods, and calendars. These are made available with the command

```
import cdtime
```

Two time types are available: *relative time* and *component time*. Relative time is time relative to a fixed base time. It consists of:

- a `units` string, of the form “`units since basetime`”, and
- a floating-point `value`

For example, the time “28.0 days since 1996-1-1” has `value=28.0`, and `units="days since 1996-1-1"`

Component time consists of the integer fields `year`, `month`, `day`, `hour`, `minute`, and the floating-point field `second`. A sample component time is 1996-2-28 12:10:30.0

The **cdtime** module contains functions for converting between these forms, based on the common calendars used in climate simulation. Basic arithmetic and comparison operators are also available.

---

### 3.2 *Calendars*

A calendar specifies the number of days in each month, for a given year. **cdtime** supports these calendars:

- **cdtime.GregorianCalendar**: years evenly divisible by four are leap years, except century years not evenly divisible by 400. This is sometimes called the *proleptic* Gregorian calendar, meaning that the algorithm for leap years applies for all years.
- **cdtime.MixedCalendar**: mixed Julian/Gregorian calendar. Dates before 1582-10-15 are encoded with the Julian calendar, otherwise are encoded with the Gregorian calendar. The day immediately following 1582-10-4 is 1582-10-15. This is the default calendar.
- **cdtime.JulianCalendar**: years evenly divisible by four are leap years,
- **cdtime.NoLeapCalendar**: all years have 365 days,
- **cdtime.Calendar360**: all months have 30 days.

Several **cdtime** functions have an optional calendar argument. The default calendar is the **MixedCalendar**. The default calendar may be changed with the command:

```
cdtime.DefaultCalendar = newCalendar
```

---

### 3.3 *Time Constructors*

The following table describes the methods for creating time types.

Table 3.1 Time Constructors

Type	Definition
Reltime	<p><b>cdtime.reltime(value, relunits)</b></p> <p>Create a relative time type.</p> <p><i>value</i> is an integer or floating point value.</p> <p><i>relunits</i> is a string of the form "<i>unit(s) [since basetime]</i>" where</p> <p><i>unit</i> = [second   minute   hour   day   week   month   season   year]</p> <p><i>basetime</i> has the form <i>yyyy-mm-dd hh:mi:ss</i>. The default basetime is 1979-1-1, if no since clause is specified.</p> <p>Example:</p> <pre>r = cdtime.reltime(28, "days since 1996-1-1")</pre>
Comptime	<p><b>cdtime.comptime(year, month=1, day=1, hour=0, minute=0, second=0.0)</b></p> <p>Create a component time type.</p> <p><i>year</i> is an integer.</p> <p><i>month</i> is an integer in the range 1 .. 12</p> <p><i>day</i> is an integer in the range 1 .. 31</p> <p><i>hour</i> is an integer in the range 0 .. 59</p> <p><i>minute</i> is an integer in the range 0 .. 59</p> <p><i>second</i> is a floating point number in the range 0.0 .. 60.0</p> <p>Example: <code>c = cdtime.comptime(1996, 2, 28)</code></p>

Table 3.1 Time Constructors

Type	Definition
Comptime	<p><b>cdtime.abstime(absvalue, absunits)</b></p> <p>Create a component time from an absolute time representation. <i>absvalue</i> is a floating-point encoding of an absolute time. <i>absunits</i> is the units template, a string of the form "<i>unit as format</i>", where <i>unit</i> is one of <code>second</code>, <code>minute</code>, <code>hour</code>, <code>day</code>, <code>calendar_month</code>, or <code>calendar_year</code>. <i>format</i> is a string of the form "<code>%x[%x[...]][. %f]</code>", where <i>x</i> is one of the format letters 'Y' (year, including century), 'm' (two digit month, 01=January), 'd' (two-digit day within month), 'H' (hours since midnight), 'M' (minutes), or 's' (seconds). The optional '.%f' denotes a floating-point fraction of the unit.</p> <p>Example: <code>c = cdtime.abstime(19960228.0, "day as %Y%m%d. %f")</code></p>

### 3.4 Relative Time

A relative time type has two members, `value` and `units`. Both can be set.

Table 3.2 Relative Time Members

Type	Name	Summary
Float	<code>value</code>	Number of units
String	<code>units</code>	Relative units, of the form " <code>unit(s) since basetime</code> "

---

### 3.5 *Component Time*

A component time type has six members, all of which are settable.

**Table 3.3 Component Time Members**

---

Type	Name	Summary
Integer	year	Year value
Integer	month	Month, in the range 1..12
Integer	day	Day of month, in the range 1 .. 31
Integer	hour	Hour, in the range 0 .. 59
Integer	minute	Minute, in the range 0 .. 59
Float	second	Seconds, in the range 0.0 .. 60.0

---

### 3.6 *Time Methods*

The following methods apply both to relative and component times.

Table 3.4 Time Methods

Type	Definition
Comptime or Reltime	<p><b>t.add(value, intervalUnits, calendar=cdtime.DefaultCalendar)</b></p> <p>Add an interval of time to a time type t. Returns the same type of time.</p> <p><i>value</i> is the Float number of interval units.</p> <p><i>intervalUnits</i> is <code>cdtime.[Second(s)   Minute(s)   Hour(s)   Day(s)   Week(s)   Month(s)   Season(s)   Year(s)]</code></p> <p><i>calendar</i> is the calendar type.</p> <p>Example:</p> <pre>&gt;&gt;&gt; from cdtime import * &gt;&gt;&gt; c = comptime(1996,2,28) &gt;&gt;&gt; r = reltime(28,"days since 1996-1-1") &gt;&gt;&gt; print r.add(1,Days) 29.00 days since 1996-1-1 &gt;&gt;&gt; print c.add(36,Hour) 1996-2-29 12:0:0.0</pre>
Integer	<p><b>t.cmp(t2, calendar=cdtime.DefaultCalendar)</b></p> <p>Compare time values t and t2. Returns -1, 0, 1 as t is less than, equal to, or greater than t2 respectively.</p> <p><i>t2</i> is the time to compare.</p> <p><i>calendar</i> is the calendar type.</p> <p>Example:</p> <pre>&gt;&gt;&gt; from cdtime import * &gt;&gt;&gt; r = cdtime.reltime(28,"days since 1996-1-1") &gt;&gt;&gt; c = comptime(1996,2,28) &gt;&gt;&gt; print r.cmp(c) -1 &gt;&gt;&gt; print c.cmp(r) 1 &gt;&gt;&gt; print r.cmp(r) 0</pre>

Table 3.4 Time Methods

Type	Definition
Comptime or Reltime	<p><b>t.sub(value, intervalUnits, calendar=cdtime.DefaultCalendar)</b></p> <p>Subtract an interval of time from a time type t. Returns the same type of time.</p> <p><i>value</i> is the Float number of interval units.</p> <p><i>intervalUnits</i> is <code>cdtime.[Second(s)   Minute(s)   Hour(s)   Day(s)   Week(s)   Month(s)   Season(s)   Year(s)]</code></p> <p><i>calendar</i> is the calendar type.</p> <p>Example:</p> <pre>&gt;&gt;&gt; from cdtime import * &gt;&gt;&gt; r = cdtime.reftime(28,"days since 1996-1-1") &gt;&gt;&gt; c = comptime(1996,2,28) &gt;&gt;&gt; print r.sub(10,Days) 18.00 days since 1996-1-1 &gt;&gt;&gt; print c.sub(30,Days) 1996-1-29 0:0:0.0</pre>
Comptime	<p><b>t.tocomp(calendar = cdtime.DefaultCalendar)</b></p> <p>Convert to component time. Returns the equivalent component time.</p> <p><i>calendar</i> is the calendar type.</p> <p>Example:</p> <pre>&gt;&gt;&gt; r = cdtime.reftime(28,"days since 1996-1-1") &gt;&gt;&gt; r.tocomp() 1996-1-29 0:0:0.0</pre>
Reltime	<p><b>t.torel(units, calendar=cdtime.DefaultCalendar)</b></p> <p>Convert to relative time. Returns the equivalent relative time.</p> <p>Example:</p> <pre>&gt;&gt;&gt; c = comptime(1996,2,28) &gt;&gt;&gt; print c.torel("days since 1996-1-1") 58.00 days since 1996-1-1 &gt;&gt;&gt; r = reltime(28,"days since 1996-1-1") &gt;&gt;&gt; print r.torel("days since 1995") 393.00 days since 1995 &gt;&gt;&gt; print r.torel("days since 1995").value 393.0</pre>



---

## 4.1 Overview

CDMS has functions to interpolate gridded data:

- from one horizontal (lat/lon) grid to another
- from one set of pressure levels to another
- from one vertical (lat/level) cross-section to another vertical cross-section.

### 4.1.1 Horizontal regrider

The simplest method to regrid a variable from one horizontal, lat/lon grid to another is to use the **regrid** function defined for variables. This function takes the target grid as an argument, and returns the variable regridded to the target grid:

```
>>> import cdms
>>> f = cdms.open('/pcmdi/cdms/exp/cmip2/ccc/perturb.xml')
>>> rlsf = f('rls') # Read the data
>>> rlsf.shape
(4, 48, 96)
>>> g = cdms.open('/pcmdi/cdms/exp/cmip2/mri/perturb.xml')
>>> rls_g = g['rls'] # Get the file variable (no data read)
```

```
>>> outgrid = rlsf.getGrid() # Get the target grid
>>> rlsnew = rlsf.regrid(outgrid)
>>> rlsnew.shape
(4, 46, 72)
>>> outgrid.shape
(46, 72)
```

A somewhat more efficient method is to create a regridding function. This has the advantage that the mapping is created only once and can be used for multiple arrays. Also, this method can be used with data in the form of an `MA.MaskedArray` or `Numeric` array. The steps in this process are:

- Given an input grid and output grid, generate a regridding function.
- Call the regridding function on a `Numeric` array, resulting in an array defined on the output grid. The regridding function can be called with any array or variable defined on the input grid.

The following example illustrates this process. The regridding function is generated at line 9, and the regridding is performed at line 10:

```
1 #!/usr/bin/env python
2 import cdms
3 from regrid import Regridder
4 f = cdms.open('/pcmdi/cdms/exp/cmip2/ccc/perturb.xml')
5 rlsf = f['rls']
6 ingrid = rlsf.getGrid()
7 g = cdms.open('/pcmdi/cdms/exp/cmip2/mri/perturb.xml')
8 outgrid = g['rls'].getGrid()
9 regridfunc = Regridder(ingrid, outgrid)
10 rlsnew = regridfunc(rlsf)
11 f.close()
12 g.close()
```

Line	Notes
2	Makes the <code>CDMS</code> module available.
3	Makes the <code>Regridder</code> class available from the <code>regrid</code> module.
4	Opens the input dataset.

<b>Line</b>	<b>Notes</b>
5	Gets the variable object named <code>'r1s'</code> . No data is read.
6	Gets the input grid.
7	Opens a dataset to retrieve the output grid.
8	The output grid is the grid associated with the variable named <code>'r1s'</code> in dataset <code>g</code> . Just the grid is retrieved, not the data.
9	Generates a regridding function <code>regridfunc</code> .
10	Reads all data for variable <code>r1sf</code> , and calls the regridding function on that data, resulting in a transient variable <code>r1snew</code> .

### 4.1.2 Pressure-level regridding

To regrid a variable which is a function of latitude, longitude, pressure level, and (optionally) time to a new set of pressure levels, use the **pressureRegrid** function defined for variables. This function takes an axis representing the target set of pressure levels, and returns a new variable `d` regridded to that dimension.

```
>>> var.shape
(3, 16, 32)
>>> var.getAxisIds()
['level', 'latitude', 'longitude']
>>> len(levout)
2
>>> result = var.pressureRegrid(levout)
>>> result.shape
(2, 16, 32)
```

### 4.1.3 Cross-section regridding

To regrid a variable which is a function of latitude, height, and (optionally) time to a new latitude/height cross-section, use the **crossSec-**

**tionRegridder** defined for variables. This function takes as arguments the new latitudes and heights, and returns the variable regridded to those axes.

```
>>> varin.shape
(11, 46)
>>> varin.getAxisIds()
['level', 'latitude']
>>> levOut[:]
[10., 30., 50., 70., 100., 200., 300., 400., 500.,
 700., 850.,
 1000.,]
>>> varout = varin.crossSectionRegrid(levOut, latOut)
>>> varout.shape
(12, 64)
```

---

## 4.2 *regrid module*

The `regrid` module implements the regridding functionality. Although this module is not strictly a part of CDMS, it is designed to work with CDMS objects. The Python command

```
from regrid import Regridder
```

makes the `Regridder` class available within a Python program. An instance of `Regridder` is a function which regrids data from input to output grid.

---

**Table 4.1 Regridder Constructor**

---

### **regridFunction = Regridder(inputGrid, outputGrid)**

Create a `regridder` function which interpolates a data array from input to output grid. Table 4.2 on page 120 describes the calling sequence of this function.

*inputGrid* and *outputGrid* are CDMS grid objects.

Note: To set the mask associated with *inputGrid* or *outputGrid*, use the grid `setMask` function.

### 4.3 regridder functions

A regridder function is an instance of the `Regridder` class. The function is associated with an input and output grid. Typically its use is straightforward: the function is passed an input array and returns the regridded array. However, when the array has missing data, or the input and/or output grids are masked, the logic becomes more complicated.

**Step 1:** The regridder function first forms an *input mask*. This mask is either two-dimensional or ‘n-dimensional’, depending on the rank of the user-supplied mask. If no mask or missing value is specified, the mask is obtained from the data array mask if present.

**Two-dimensional case:**

- Let *mask\_1* be the two-dimensional user mask supplied via the `mask` argument, or the mask of the input grid if no user mask is specified.
- If a missing-data value is specified via the `missing` argument, let the *implicit\_mask* be the two-dimensional mask defined as 0 where the first horizontal slice of the input array is missing, 1 elsewhere.
- The input mask is the logical `AND(mask_1, implicit_mask)`

**N-dimensional case:** If the user mask is 3 or 4-dimensional with the same shape as the input array, it is used as the input mask.

**Step 2:** The data is then regridded. In the two-dimensional case, the input mask is ‘broadcast’ across the other dimensions of the array. In other words, it assumes that all horizontal slices of the array have the same mask. The result is a new array, defined on the output grid. Optionally, the regridder function can also return an array having the same shape as the output array, defining the fractional area of the output array which overlaps a non-missing input grid cell. This is useful for calculating area-weighted means of masked data.

**Step 3:** Finally, if the output grid has a mask, it is applied to the result array. Where the output mask is 0, data values are set to the missing data value, or 1.0e20 if undefined. The result array or transient variable will have a mask

value of 1 (invalid value) for those output grid cells which completely overlap input grid cells with missing values.



Table 4.2 Regridder function

## Type

Array or  
Transient-  
Variable

***regridFunction*(array, missing=None, order=None, mask=None)**

Interpolate a gridded data array to a new grid. The interpolation preserves the area-weighted mean on each horizontal slice. If array is a Variable, a TransientVariable of the same rank as the input array is returned, otherwise a masked array is returned.

*array* is a Variable, masked array, or Numeric array of rank 2, 3, or 4.

*missing* is a Float specifying the missing data value. The default is 1.0e20.

*order* is a string indicating the order of dimensions of the array. It has the form returned from **variable.getOrder()**. For example, the string “tzyx” indicates that the dimension order of *array* is (time, level, latitude, longitude). If unspecified, the function assumes that the last two dimensions of *array* match the input grid.

*mask* is a Numeric array, of datatype Integer or Float, consisting of a fractional number between 0 and 1. A value of 1 or 1.0 indicates that the corresponding data value is to be ignored for purposes of regridding. A value of 0 or 0.0 indicates that the corresponding data value is valid. This is consistent with the convention for masks used by the MA module. A fractional value between 0.0 and 1.0 indicates the fraction of the data value (e.g., the corresponding cell) to be ignored when regridding. This is useful if a variable is regridded first to grid A and then to another grid B; the mask when regridding from A to B would be  $(1.0 - f)$  where  $f$  is the maskArray returned from the initial grid operation using the returnTuple argument.

If *mask* is two-dimensional of the same shape as the input grid, it overrides the mask of the input grid. If the mask has more than two dimensions, it must have the same shape as *array*. In this case, the *missing* data value is also ignored. Such an  $n$ -dimensional mask is useful if the pattern of missing data varies with level (e.g., ocean data) or time.

Note: If neither *missing* or *mask* is set, the default mask is obtained from the mask of the array if any.

Table 4.2 Regridder function

## Type

Array, Array ***regridFunction*(ar, missing=None, order=None, mask=None, returnTuple=1)**

If called with the optional returnTuple argument equal to 1, the function returns a tuple (*dataArray*, *maskArray*). *dataArray* is the result data array. *maskArray* is a Float32 array of the same shape as *dataArray*, such that *maskArray*[*i,j*] is fraction of the output grid cell [*i,j*] overlapping a non-missing cell of the input grid.

## 4.4 Examples

**Example:** Regrid data to a uniform output grid.

```

1 #!/usr/local/bin/python
2 import cdms
3 from regrid import Regridder
4 f = cdms.open('rls_ccc_per.nc')
5 rlsf = f.variables['rls']
6 ingrid = rlsf.getGrid()
7 outgrid = cdms.createUniformGrid(90.0, 46, -4.0, 0.0, 72, 5.0)
8 regridFunc = Regridder(ingrid, outgrid)
9 newrls = regridFunc(rlsf)
10 f.close()

```

Line	Notes
------	-------

4	Open a netCDF file for input.
---	-------------------------------

<b>Line</b>	<b>Notes</b>
7	Create a 4 x 5 degree output grid. Note that this grid is not associated with a file or dataset
8	Create the regridding function
9	Read all data and regrid. The missing data value is obtained from variable rlsf.

**Example:** Get a mask from a separate file, and set as the input grid mask.

```
1 import cdms
2 from regrid import Regridder
3 #
4 f = cdms.open('so_ccc_per.nc')
5 sof = f.variables['so']
6 ingrid = sof.getGrid()
7 g = cdms.open('rls_mri_per.nc')
8 rlsf = g.variables['rls']
9 outgrid = rlsf.getGrid()
10 regridFunc = Regridder(ingrid,outgrid)
11 h = cdms.open('sft_ccc.nc')
12 sfmaskvar = h.variables['sfmask']
13 sfmask = sfmaskvar[:]
14 outArray = regridFunc(sof.subSlice(time=0),mask=sfmask)
15 f.close()
16 g.close()
17 h.close()
```

<b>Line</b>	<b>Notes</b>
6	Get the input grid.
9	Get the output grid
10	Create the regridding function.

<b>Line</b>	<b>Notes</b>
13	Get the mask.
14	Regrid with a user mask. The subslice call returns a transient variable corresponding to variable <code>sof</code> at time 0.  Note: Although it cannot be determined from the code, both <code>mask</code> and the input array <code>sof</code> are four-dimensional. This is the ‘n-dimensional’ case.

**Example:** Generate an array of zonal mean values.

```
1 f = cdms.open('rls_ccc_per.nc')
2 rlsf = f.variables['rls']
3 ingrid = rlsf.getGrid()
4 outgrid = cdms.createZonalGrid(ingrid)
5 regridFunc = Regridder(ingrid,outgrid)
6 mean = regridFunc(rlsf)
7 f.close()
```

<b>Line</b>	<b>Notes</b>
3	Get the input grid.
4	Create a zonal grid. <code>outgrid</code> has the same latitudes as <code>ingrid</code> , and a singleton longitude dimension. <code>createGlobalMeanGrid</code> could be used here to generate a global mean array.
5	Generate the regridder function.
6	Generate the zonal mean array.

**Example:** Regrid an array with missing data, and calculate the area-weighted mean of the result.

```
1 from cdms.MV import *
 ...
2 outgrid = cdms.createUniformGrid(90.0, 46, -4.0, 0.0, 72, 5.0)
3 outlatw, outlonw = outgrid.getWeights()
4 outweights = outerproduct(outlatw, outlonw)
5 grid = var.getGrid()
6 sample = var[0,0]
7 latw, lonw = grid.getWeights()
8 weights = outerproduct(latw, lonw)
9 inmask = where(greater(absolut(sample),1.e15),0,1)
10 mean = add.reduce(ravel(inmask*weights*sample))/
 add.reduce(ravel(inmask*weights))
11 regridFunc = Regridder(grid, outgrid)
12 outsample, outmask = regridFunc(sample, mask=inmask,
 returnTuple=1)
13 outmean = add.reduce(ravel(outmask*outweights*outsample))/
 add.reduce(ravel(outmask*outweights))
```

Line	Notes
2	Create a uniform target grid.
3	Get the latitude and longitude weights.
4	Generate a 2-D weights array.
5	Get the input grid. <code>var</code> is a 4-D variable.
6	Get the first horizontal slice from <code>var</code> .
7-8	Get the input weights, and generate a 2-D weights array.
9	Set the 2-D input mask.
10	Calculate the input array area-weighted mean.
11	Create the regridding function.

<b>Line</b>	<b>Notes</b>
12	Regrid. Because <code>returnTuple</code> is set to 1, the result is a tuple (dataArray, maskArray).
13	Calculate the area-weighted mean of the regridded data. <code>mean</code> and <code>outmean</code> should be approximately equal.



# *Plotting CDMS data in Python*

---

## *5.1 Overview*

Data read via the CDMS Python interface can be plotted using the **vcs** module. This module, part of the Climate Data Analysis Tool (CDAT) is documented in the CDAT reference manual. The **vcs** module provides access to the functionality of the VCS visualization program.

Examples of plotting data accessed from CDMS are given below, as well as documentation for the **plot** routine keywords.

---

## *5.2 Examples*

In the following examples, it is assumed that variable **ps1** is dimensioned (time, latitude, longitude). **ps1** is contained in the dataset named `'sample.xml'`.

### **5.2.1 Example: plotting a horizontal grid**

```
1 import cdms, vcs
2 #
```

```
3 f = cdms.open('sample.xml')
4 psl = f.variables['psl']
5 sample = psl[0]
6 w=vcs.init()
7 #
8 w.plot(sample)
9 f.close()
```

Notes:

Line	Notes
5	Get a horizontal slice, for the first timepoint.
6	Create a VCS Canvas <code>w</code> .
8	Plot the data. Because <code>sample</code> is a transient variable, it encapsulates all the time, latitude, longitude, and attribute information.
9	Close the file. This must be done after the reference to the persistent variable <code>psl</code> .

That's it! The axis coordinates, variable name, description, units, etc. are obtained from variable `sample`.

What if the units are not explicitly defined for `psl`, or a different description is desired? `plot` has a number of other keywords which 'fill in' the extra plot information.

### 5.2.2 Example: using plot keywords.

```
w.plot(array, xaxis=lon, yaxis=lat, units='mm/day',
 file_comment='High-frequency reanalysis', long_name="Sea level
 pressure", comment1="Sample plot", hms="18:00:00", ymd="1978/
 01/01")
```

Notes:

- Keyword arguments can be listed in any order.
- Specific keywords take precedence over general keywords.

### 5.2.3 Example: plotting a time-latitude slice

Assuming that variable `psl` has domain (time,latitude,longitude), this example selects and plots a time-latitude slice:

```
1 samp = psl[:, :, 0]
2 w = vcs.init()
3 w.plot(samp, name='sea level pressure')
```

Notes:

Line	Notes
1	<code>samp</code> is a slice of <code>psl</code> , at index 0 of the last dimension. Since <code>samp</code> was obtained from the slice operator, it is a transient variable, which includes the latitude and time information.
3	The <b>name</b> keyword defines the identifier, by default the name in the file.

### 5.2.4 Example: plotting subsetted data

The **subRegion** and **subSlice** methods return a transient variable, which contains axis and attribute values. These functions are especially useful for reading data to be plotted.

```
...
1 samp = psl.subRegion(time=(0.0,100.0), longitude=180.0)
2 w = vcs.init()
3 w.plot(samp)
```

---

### 5.3 *plot method*

The **plot** method is documented in the CDAT Reference Manual. This section augments the documentation with a description of the optional keyword arguments.

The general form of the plot command is:

```
canvas.plot(array [, args] [,key=value [, key=value [, ...]])
```

where:

- *canvas* is a VCS Canvas object, created with the **vcs.init** method.
- *array* is a variable, masked array, or Numeric array having between two and five dimensions. The last dimensions of the array is termed the 'x' dimension, the next-to-last the 'y' dimension, then 'z', 't', and 'w'. For example, if *array* is three-dimensional, the axes are (z,y,x), and if *array* is four-dimensional, the axes are (t,z,y,x). (Note that the 't' dimension need have no connection with time; any spatial axis can be mapped to any plot dimension. For a graphics method which is two-dimensional, such as boxfill, the y-axis is plotted on the horizontal, and the x-axis on the vertical.
- *args* are optional positional arguments:

```
args := template_name, graphics_method, graphics_name
template_name: the name of the VCS template (e.g., 'AMIP')
graphics_method: the VCS graphics method ('boxfill')
graphics_name: the name of the specific graphics method ('default')
```

See the CDAT Reference Manual and VCS Reference Manual for a detailed description of these arguments.

- *key=value*, ... are optional keyword/value pairs, listed in any order. These are defined in Table 5.1 on page 131.

Table 5.1 plot keywords

key	type	value
<b>comment1</b>	string	Comment plotted above file_comment
<b>comment2</b>	string	Comment plotted above comment1
<b>comment3</b>	string	Comment plotted above comment2
<b>continents</b>	0 or 1	if ==1, plot continental outlines (default: plot if xaxis is longitude, yaxis is latitude -or- xname is 'longitude' and yname is 'latitude')
<b>file_comment</b>	string	Comment, defaults to variable.parent.comment)
<b>grid</b>	CDMS grid object	Grid associated with the data. Defaults to variable.getGrid()
<b>hms</b>	string	Hour, minute, second
<b>long_name</b>	string	Descriptive variable name, defaults to variable.long_name.
<b>missing_value</b>	same type as array	Missing data value, defaults to variable.getMissing()
<b>name</b>	string	Variable name, defaults to variable.id
<b>time</b>	cdtime relative or absolute time	time associated with the data. Example: cdttime.reltime(30.0, "days since 1978-1-1")
<b>units</b>	string	Data units. Defaults to variable.units

Table 5.1 plot keywords

key	type	value
<b>variable</b>	CDMS variable object	Variable associated with the data. The variable grid must have the same shape as the data array.
<b>xarray</b> ([y z t w]array)	1-D Numeric array	Array of coordinate values, having the same length as the corresponding dimension. Defaults to <code>xaxis[:]</code> ( <code>y z t waxis[:]</code> )
<b>xaxis</b> ([y z t w]axis)	CDMS axis object	Axis object. <b>xaxis</b> defaults to <code>grid.getAxis(0)</code> , <b>yaxis</b> defaults to <code>grid.getAxis(1)</code>
<b>xbounds</b> ( <b>ybounds</b> )	2-D Numeric array	Boundary array of shape (n,2) where n is the axis length. Defaults to <code>xaxis.getBounds()</code> , or <code>xaxis.genGenericBounds()</code> if None, similarly for <b>ybounds</b> .
<b>xname</b> ([y z t w]name)	string	Axis name. Defaults to <code>xaxis.id</code> ( <code>[y z t w]axis.id</code> )
<b>xrev</b> ( <b>yrev</b> )	0 or 1	If <b>xrev</b> ( <b>yrev</b> ) is 1, reverse the direction of the x-axis (y-axis). Defaults to 0, with the following exceptions: <ul style="list-style-type: none"> <li>• If the y-axis is latitude, and has decreasing values, <b>yrev</b> defaults to 1</li> <li>• If the y-axis is a vertical level, and has increasing pressure levels, <b>yrev</b> defaults to 1.</li> </ul>
<b>xunits</b> ([y z t w]units)	string	Axis units. Defaults to <code>xaxis.units</code> ( <code>[y z t w]axis.units</code> ).





# *Climate Data Markup Language (CDML)*

---

## *6.1 Introduction*

The Climate Data Markup Language (CDML) is the markup language used to represent metadata in CDMS. CDML is based on the W3C XML standard (<http://www.w3.org>). This chapter defines the syntax of CDML. Read this section if you will be building or maintaining a CDMS database.

XML, the eXtensible Markup Language, makes it possible to define interoperable dialects of markup languages. The most recent version of HTML, the Web hypertext markup language, is an XML dialect. CDML is also an XML dialect, geared toward the representation of gridded climate datasets. XML provides rigor to the metadata representation, ensuring that applications can access it correctly. XML also deals with internationalization issues, and holds forth the promise that utilities for browsing, editing, and other common tasks will be available in the future.

CDML files have the file extension **.xml** or **.cdml**.

## 6.2 Elements

A CDML document consists of a nested collection of *elements*. An *element* is a description of the metadata associated with a CDMS object. The form of an element is:

```
<tag attribute-list> element-content </tag>
```

or

```
<tag attribute-list />
```

where

- `tag` is a string which defines the type of element
- `attribute-list` is a blank-separated list of attribute-value pairs, of the form:

```
attribute = "value"
```

- `element-content` depends on the type of element. It is either a list of elements, or text which defines the element values. For example, the content of an axis element either is a list of axis values, or is a **linear** element. For datasets, the content is the blank-separated list of elements corresponding to the axes, grids, and variables contained in the dataset.

The CDML elements are:

**Table 6.1 CDML Tags**

Tag	Description
attr	Extra attribute
axis	Coordinate axis
domain	Axes on which a variable is defined
domElem	Element of a variable domain
linear	Linearly-spaced axis values

**Table 6.1 CDML Tags**

Tag	Description
rectGrid	Rectilinear Grid
variable	Variable

---

### 6.3 *Special Characters*

XML reserves certain characters for markup. If they appear as content, they must be encoded to avoid confusion with markup:

**Table 6.2 Special Character Encodings**

Character	Encoding
<	&lt;
>	&gt;
&	&amp;
“	&quot;
‘	&apos;

For example, the comment

```
Certain "special characters", such as <, >, and ` , must
be encoded.
```

would appear in an attribute string as:

```
comment = "Certain "special characters", such
as <, >, and ', must be encoded."
```

---

## 6.4 Identifiers

In CDMS, all objects in a dataset have a unique string *identifier*. The **id** attribute holds the value of this identifier. If the variable, axis, or grid has a string name within a data file, then the **id** attribute ordinarily has this value. Alternatively, the name of the object in a data file can be stored in the **name\_in\_file** attribute, which can differ from the **id**. Datasets also have IDs, which can be used within a larger context (databases).

An identifier must start with an alphabetic character (upper or lower case), an underscore (`_`), or a colon (`:`). Characters after the first must be alphanumeric, an underscore, or colon. There is no restriction on the length of an identifier.

---

## 6.5 CF Metadata Standard

The CF metadata standard (<http://www.cgd.ucar.edu/cms/eaton/netcdf/CF-current.htm>) defines a set of conventions for usage of netCDF. This standard is supported by CDML. The document defines names and usage for metadata attributes. CF supersedes the GDT 1.3 standard.

---

## 6.6 CDML Syntax

The following notation is used in this section:

- Courier font is used for a syntax specification. **Bold font** highlights literals.
- $(R|S)$  denotes ‘either R or S’.
- $R^*$  denotes ‘zero or more R’.
- $R^+$  denotes ‘one or more R’.

A CDML document consists of a prolog followed by a single dataset element.

1. `CDML-document ::= prolog dataset-element`

The prolog defines the XML version, and the Document Type Definition (DTD), a formal specification of the document syntax. See <http://www.w3.org/TR/1998/REC-xml-19980210> for a formal definition of XML Version 1.0.

- ```
2. prolog ::=
  <?xml version="1.0"?>
  <!DOCTYPE dataset SYSTEM "http://www-pcmdi.llnl.gov/
  ~drach/cdms/cdml.dtd">
```

6.6.1 Dataset Element

A dataset element describes a single dataset. The content is a list of elements corresponding to the axes, grids, and variables contained in the dataset. Axis, variable, and grid elements can be listed in any order, and an element ID can be used before the element is actually defined.

- ```
3. dataset-element ::= <dataset dataset-attributes>
 dataset-content </dataset>
4. dataset-content ::= (axis-element | grid-element |
 variable-element)* extra-attribute-element+
```

**Table 6.3 Dataset Attributes**

Attribute	Required	CF	GDT	Notes
appendices	N	N	Y	Version number
calendar	N	N	Y	Calendar used for encoding time axes. “gregorian”   “julian”   “no leap”   “360_day”   “proleptic_gregorian”   “standard” Note: for the CF convention, the calendar attribute is placed on the time axis.
comment	N	Y	Y	Additional dataset information

**Table 6.3 Dataset Attributes**

Attribute	Required	CF	GDT	Notes
Conventions	Y	Y	Y	The netCDF metadata standard. Example: “CF-1.0”
cdms_file_map	Y	N	N	Map of partitioned axes to files. See note below.
directory	N	N	N	Root directory of the dataset
frequency	N	N	N	Temporal frequency
history	N	Y	Y	Evolution of the data
id	Y	N	N	Dataset identifier
institution	N	Y	Y	Who made or supplied the data
production	N	N	Y	How the data was produced (see source)
project	N	N	N	Project associated with the data Example: “CMIP 2”
references	N	Y	N	Published or web-based references that describe the data or methods used to produce it.
source	N	Y	N	The method of production of the original data.
title	N	Y	N	A succinct description of the data.

**Notes:**

- The `cdms_filemap` attribute describes how the dataset is partitioned into files. The format is:

```

filemap ::= [varmap, varmap, ...]
varmap ::= [namelist, slicelist]
namelist ::= [name, name, ...]
slicelist ::= [indexlist, indexlist, ...,]
indexlist ::= [time0, time1, lev0, lev1, path]
name ::= variable name
time0 ::= first index of time in the file, or ‘-’ if not split on time
time1 ::= last index of time + 1, in the file, or ‘-’ if not split on time

```

lev0 ::= first index of vertical levels in the file, or ‘-’ if not split on level  
 lev1 ::= last index +1 of vertical levels in the file, or ‘-’ if not split on level  
 path ::= pathname of the file containing data for this time/level range.

The pathname is appended to the value of the directory attribute, to obtain an absolute pathname.

## 6.6.2 Axis Element

An axis element describes a single coordinate axis. The content can be a blank-separated list of axis values or a linear element. A linear element is a representation of a linearly-spaced axis as (start, delta, length).

5. axis-element ::= **<axis** axis-attributes> axis-content>  
 </axis>
6. axis-content ::= (axis-values | linear-element)  
 extra-attribute-element\*
7. axis-values ::= [value\*]
8. linear-element ::= **<linear** delta="value"  
 length="Integer" start="value"> </linear>

**Table 6.4 Axis Attributes**

Attribute	Required?	CF	GDT	Notes
associate	N	N	Y	IDs of variables containing alternative sets of coordinates.
axis	N	Y	Y	The spatial type of the axis: “T” - time “X” - longitude “Y” - latitude “Z” - vertical level “-” - not spatiotemporal
bounds	N	Y	Y	ID of the boundary variable

Table 6.4 Axis Attributes

Attribute	Required?	CF	GDT	Notes
calendar	N	Y	N	See dataset.calendar
climatology	N	Y	N	Range of dates to which climatological statistics apply.
comment	N	Y	N	String comment
compress	N	Y	Y	Dimensions which have been compressed by gathering
datatype	Y	N	N	Char, Short, Long, Float, Double, or String
dates	N	Y	N	Range of dates to which statistics for a typical diurnal cycle apply.
expand	N	N	Y	Coordinates prior to contraction
formula_terms	N	Y	N	Variables that correspond to the terms in a formula.
id	Y	N	N	Axis identifier. Also the name of the axis in the underlying file(s), if <b>name_in_file</b> is undefined.
isvar	N	N	N	“true”   “false” “false” if the axis does not have coordinate values explicitly defined in the underlying file(s). Default: “true”
leap_month	N	Y	N	For a user-defined calendar, the month which is lengthened by a day in leap years.
leap_year	N	Y	N	An example of a leap year for a user-defined calendar. All years that differ from this year by a multiple of four are leap years.
length	N	N	N	Number of axis values, including values for which no data is defined. Cf. <b>partition_length</b> .

Table 6.4 Axis Attributes

Attribute	Required?	CF	GDT	Notes
long_name	N	Y	Y	Long description of a physical quantity
modulo	N	N	Y	Arithmetic modulo of an axis with circular topology.
month_lengths	N	Y	N	Length of each month in a non-leap year for a user-defined calendar.
name_in_file	N	N	N	Name of the axis in the underlying file(s). See id.
partition	N	N	N	How the axis is split across files.
partition_length	N	N	N	Number of axis points for which data is actually defined. If data is missing for some values, this will be smaller than the <b>length</b> .
positive	N	Y	Y	Direction of positive for a vertical axis
standard_name	N	Y	N	Reference to an entry in the standard name table.
topology	N	N	Y	Axis topology. “circular”   “linear”
units	Y	Y	Y	Units of a physical quantity
weights	N	N	N	Name of the weights array

### 6.6.3 Grid Element

A grid element describes a horizontal, latitude-longitude grid which is rectangular in topology,

```
9. grid-element ::= <rectGrid grid-attributes> extra-attribute-element* </rectGrid>
```

Table 6.5 RectGrid Attributes

Attribute	Required?	GDT?	Notes
id	Y	N	Grid identifier
type	Y	N	Grid classification “gaussian”   “uniform”   “equalarea”   “generic” Default: “generic”
latitude	Y	N	Latitude axis name
longitude	Y	N	Longitude axis name
mask	N	N	Name of associated mask variable
order	Y	N	Grid ordering “yx”   “xy” Default: “yx”, axis order is latitude, longitude

### 6.6.4 Variable Element

A variable element describes a data variable. The domain of the variable is an ordered list of *domain elements* naming the axes on which the variable is defined. A domain element is a reference to an axis or grid in the dataset.

The **length** of a domain element is the number of axis points for which data can be retrieved. The **partition\_length** is the number of points for which data is actually defined. If data is missing, this is less than the **length**.

10. variable-element ::= <variable variable-attributes>  
variable-content </variable>
11. variable-content ::= variable-domain extra-attribute-element\*
12. variable-domain ::= <domain> domain-element\* </domain>

```
13. domain-element ::= <domElem name="axis-name"
 start="Integer" length="Integer"
 partition_length="Integer"/>
```

Table 6.6 Variable Attributes

Attribute	Required?	CF	GDT	Notes
id	Y	N	N	Variable identifier. Also, the name of the variable in the underlying file(s), if <b>name_in_file</b> is undefined.
add_offset	N	Y	Y	Additive offset for packing data. See <b>scale_factor</b> .
associate	N	N	Y	IDs of variables containing alternative sets of coordinates
axis	N	N	Y	Spatio-temporal dimensions. Ex: "TYX" for a variable with domain (time, latitude, longitude) Note: for CF, applies to axes only.
cell_methods	N	Y	N	The method used to derive data that represents cell values, e.g., "maximum", "mean", "variance", etc.
comments	N	N	N	Comment string
coordinates	N	Y	N	IDs of variables containing coordinate data.
datatype	Y	N	N	Char, Short, Long, Float, Double, or String
grid_name	N	N	N	Id of the grid
grid_type	N	N	N	"gaussian"   "uniform"   "equalarea"   "generic"
long_name	N	Y	Y	Long description of a physical quantity.

Table 6.6 Variable Attributes

Attribute	Required?	CF	GDT	Notes
missing_value	N	Y	Y	Value used for data that are unknown or missint.
name_in_file	N	N	N	Name of the variable in the underlying file(s). See <i>id</i> .
scale_factor	N	Y	Y	Multiplicative factor for packing data. See <b>add_offset</b> .
standard_name	N	Y	N	Reference to an entry in the standard name table.
subgrid	N	N	Y	Records how data values represent subgrid variation.
template	N	N	N	Name of the file template to use for this variable. Overrides the dataset value.
units	N	Y	Y	Units of a physical quantity.
valid_max	N	Y	Y	Largest valid value of a variable
valid_min	N	Y	Y	Smallest valid value of a variable
valid_range	N	Y	Y	Largest and smallest valid values of a variable

### 6.6.5 Attribute Element

Attributes which are not explicitly defined by the GDT convention are represented as extra attribute elements. Any dataset, axis, grid, or variable element can have an extra attribute as part of its content. This representation is also useful if the attribute value has non-blank whitespace characters (carriage returns, tabs, linefeeds) which are significant.

The datatype is one of: Char, Short, Long, Float, Double, or String.

```
14. extra-attribute-element ::= <attr name=attribute-name
 datatype="attribute-datatype"> attribute-value </
 attr>
```

## 6.7 A Sample CDML Document

Dataset 'sample' has two variables, and six axes.

Note:

- The file is indented for readability. This is not required; the added whitespace is ignored.
- The dataset contains three axes and two variables. Variables *u* and *v* are functions of time, latitude, and longitude.
- The global attribute `cdms_filemap` describes the mapping between variables and files. The entry `[[u],[[0,1,-,-,u_2000.nc],[1,2,-,-,u_2001.nc],[2,3,-,-,u_2002.nc]]]` indicates that variable *u* is contained in file `u_2000.nc` for time index 0, `u_2001.nc` for time index 1, etc.

```
<?xml version="1.0"?>
<?xml version="1.0"?>
<!DOCTYPE dataset SYSTEM "http://www-pcmdi.llnl.gov/software/cdms/cdml.dtd">
<dataset
 Conventions="CF-1.0"
 id ="sample"
 calendar="gregorian"
 directory=""
 cdms_filemap="[[u],[[0,1,-,-,u_2000.nc],[1,2,-,-,u_2001.nc],[2,3,-,-,
 u_2002.nc]]],[[v],[[0,1,-,-,v_2000.nc],[1,2,-,-,v_2001.nc],[2,3,-,-,
 v_2002.nc]]]"
 history="
[2002-1-7 18:21:41] /idoru/cdat/3.1/bin/cdscan -d sample -x sample.xml u_2000.nc
 u_2001.nc u_2002.nc v_2000.nc v_2001.nc v_2002.nc"
>
 <axis
 id ="latitude"
 length="16"
 units="degrees_north"
 datatype="Double"
 >
 [-90. -78. -66. -54. -42. -30. -18. -6. 6. 18. 30. 42. 54. 66.
 78.
 90.]
 </axis>
 <axis
 id ="longitude"
 length="32"
 units="degrees_east"
 datatype="Double"
 >
 [0. 11.25 22.5 33.75 45. 56.25 67.5 78.75 90.
 101.25 112.5 123.75 135. 146.25 157.5 168.75 180. 191.25
 202.5 213.75 225. 236.25 247.5 258.75 270. 281.25 292.5
 303.75 315. 326.25 337.5 348.75]
 </axis>
</axis>
```

```
id ="time"
partition="[0 1 1 2 2 3]"
calendar="gregorian"
units="days since 2000-1-1"
datatype="Double"
length="3"
name_in_file="time"
>
[0. 366. 731.]
</axis>
<variable
id ="u"
missing_value="-99.9"
units="m/s"
datatype="Double"
>
<domain
>
<domElem name="time" length="3" start="0"/>
<domElem name="latitude" length="16" start="0"/>
<domElem name="longitude" length="32" start="0"/>
</domain>
</variable>
<variable
id ="v"
missing_value="-99.9"
units="m/s"
datatype="Double"
>
<domain
>
<domElem name="time" length="3" start="0"/>
<domElem name="latitude" length="16" start="0"/>
<domElem name="longitude" length="32" start="0"/>
</domain>
</variable>
</dataset>
```

---

***7.1 cdscan: Importing datasets into CDMS*****7.1.1 Overview**

A dataset is a partitioned collection of files. To create a dataset, the files must be scanned to produce a text representation of the dataset. CDMS represents datasets as an ASCII metafile in the CDML markup language. The file contains all metadata, together with information describing how the dataset is partitioned into files. (Note: CDMS provides a direct interface to individual files as well. It is not necessary to scan an individual file in order to access it.)

For CDMS applications to work correctly, it is important that the CDML metafile be valid. The **cdscan** utility generates a metafile from a collection of data files.

CDMS assumes that there is some regularity in how datasets are partitioned:

- A variable can be partitioned (split across files) in at most two dimensions. The partitioned dimension(s) must be either time or vertical level dimensions; variables may not be partitioned across longitude or latitude. Datasets can be parti-

tioned by variable as well. For example, one set of files might contain heat fluxes, while another set contains wind speeds.

Otherwise, there is considerable flexibility in how a dataset can be partitioned:

- Files can contain a single variable or all variables in the dataset.
- The time axis can have gaps.
- Horizontal grid boundary information and related information can be duplicated across files.
- Variables can be on different grids.
- Files may be in any of the self-describing formats supported by CDMS, including netCDF, HDF, GrADS/GRIB, and DRS.

### **7.1.2 cdscan Syntax**

The syntax of the `cdscan` command is

```
cdscan [options] file1 file2 ...
```

or

```
cdscan [options] -f file_list
```

where

- *file1 file2 ..* is a blank-separated list of files to scan
- *file\_list* is the name of a file containing a list of files to scan, one pathname per line.

Output is written to standard output by default. Use the `-x` option to specify an output filename.

Table 7.1 cdscan command options

Option	Description
-a <i>alias_file</i>	Change variable names to the aliases defined in an alias file. Each line of the alias file consists of two blank separated fields: <code>variable_id alias</code> . <code>variable_id</code> is the ID of the variable in the file, and <code>alias</code> is the name that will be substituted for it in the output dataset. Only variables with entries in the <code>alias_file</code> are renamed.
-c <i>calendar</i>	Specify the dataset calendar attribute. One of "gregorian" (default), "julian", "noleap", "proleptic_gregorian", "standard", or "360_day".
-d <i>dataset_id</i>	String identifier of the dataset. Should not contain blanks or non-printing characters. Default: "none"
-f <i>file_list</i>	File containing a list of absolute data file names, one per line.
-h	Print a help message.
-i <i>time_delta</i>	Causes the time dimension to be represented as linear, producing a more compact representation. This is useful if the time dimension is very long. <i>time_delta</i> is a float or integer. For example, if the time delta is 6 hours, and the reference units are 'hours since xxxx', set the time delta to 6. See the -r option. See Note 2.
-j	scan time as a vector dimension. Time values are listed individually. Turns off the -i option.
-l <i>levels</i>	Specify that the files are partitioned by vertical level. That is, data for different vertical levels may appear in different files. <i>levels</i> is a comma-separated list of levels containing no blanks. See Note 3.
-m <i>levelid</i>	name of the vertical level dimension. The default is the vertical dimension as determined by CDMS. See Note 3.

Table 7.1 cdscan command options

Option	Description
-p <i>template</i>	Add a file template string, for compatibility with pre-V3.0 datasets. 'cdimport -h' describes template strings.
-q	Quiet mode.
-r <i>time_units</i>	time units of the form " <i>units since yyyy-mm-dd hh:mi:ss</i> ", where <i>units</i> is one of "year", "month", "day", "hour", "minute", "second".
-s <i>suffix_file</i>	Append a suffix to variable names, depending on the directory containing the data file. This can be used to distinguish variables having the same name but generated by different models or ensemble runs. 'suffix_file' is the name of a file describing a mapping between directories and suffixes. Each line consists of two blank-separated fields: <b>directory suffix</b> . Each file path is compared to the directories in the suffix file. If the file path is in that directory or a subdirectory, the corresponding suffix is appended to the variable IDs in the file. If more than one such directory is found, the first directory found is used. If no match is made, the variable ids are not altered. Regular expressions can be used: see the example in the Notes section.
-t <i>timeid</i>	id of the partitioned time dimension. The default is the name of the time dimension as determined by CDMS. See Note 1.
-x <i>xmlfile</i>	Output file name. By default, output is written to standard output.

## Notes:

- Files can be in netCDF, GrADS/GRIB, HDF, or DRS format, and can be listed in any order. Most commonly, the files are the result of a single experiment, and the 'partitioned' dimension is time. The time dimension of a variable is the coordinate variable having a name that starts with 'time' or having an attribute axis='T'. If this is not the case, specify the time dimension with the -t option. The time dimension should be in the form supported by cftime. If this is not the case (or to override them) use the -r option.

2. By default, the time values are listed explicitly in the output XML. This can cause a problem if the time dimension is very long, say for 6-hourly data. To handle this the form `'cdscan -i delta <files>'` may be used. This generates a compact time representation of the form `<start, length, delta>`. An exception is raised if the time dimension for a given file is not linear.
3. Another form of the command is `'cdscan -l lev1,lev2,...,levn <files>'`. This asserts that the dataset is partitioned in both time and vertical level dimensions. The level dimension of a variable is the dimension having a name that starts with "lev", or having an attribute "axis=Z". If this is not the case, set the level name with the `-m` option.
4. An example of a suffix file:

```
/exp/pr/ncar-a _ncar-a
/exp/pr/ecm-a _ecm-a
/exp/ta/ncar-a _ncar-a
/exp/ta/ecm-a _ecm-a
```

For all files in directory `/exp/pr/ncar-a` or a subdirectory, the corresponding variable ids will be appended with the suffix `'_ncar-a'`. Regular expressions can be used, as defined in the Python `'re'` module. For example, The previous example can be replaced with the single line:

```
/exp/[^\/*]*/([^\/*]*) _\g<1>
```

Note the use of parentheses to delimit a group. The syntax `\g<n>` refers to the `n`-th group matched in the regular expression, with the first group being `n=1`. The string `[^\/*]*` matches any sequence of characters other than a forward slash.

### 7.1.3 Examples

```
cdscan -c noleap -d test -x test.xml [uv]*.nc
cdscan -d pcmdi_6h -i 0.25 -r 'days since 1979-1-1' *6h*.ctl
```

### 7.1.4 File Formats

Data may be represented in a variety of self-describing binary file formats, including

- netCDF, the Unidata Network Common Data Format
- HDF, the NCSA Hierarchical Data Format

- GrADS/GRIB, WMO GRIB plus a GrADS control file (.ctl)  
The first non-comment line of the control file must be a **dset** specification.
- DRS, the PCMDI legacy format.

### 7.1.5 Name Aliasing

A problem can occur if variables in different files are defined on different grids. What if the axis names are the same? CDMS requires that within a dataset, axis and variable IDs (names) be unique. What should the longitude axes be named in CDMS to ensure uniqueness? The answer is to allow CDMS IDs to differ from file names.

If a variable or axis has a CDMS ID which differs from its name in the file, it is said to have an *alias*. The actual name of the object in the file is stored in the attribute **name\_in\_file**. **cdscan** uses this mechanism (with the **-a** and **-s** options) to resolve name conflicts; a new axis or variable ID is generated, and the **name\_in\_file** is set to the axis name in the file.

Name aliases also can be used to enforce naming standards. For data received from an outside organization, variable names may not be recognized by existing applications. Often it is simpler and safer to add an alias to the metafile rather than rewrite the data.

### 7.1.6 Generating Metadata for a File

A single file can be accessed directly in CDMS, without ingesting. However, frequently it is useful to generate an ASCII description of the metadata in the file. To do this, use the filename as the template argument:

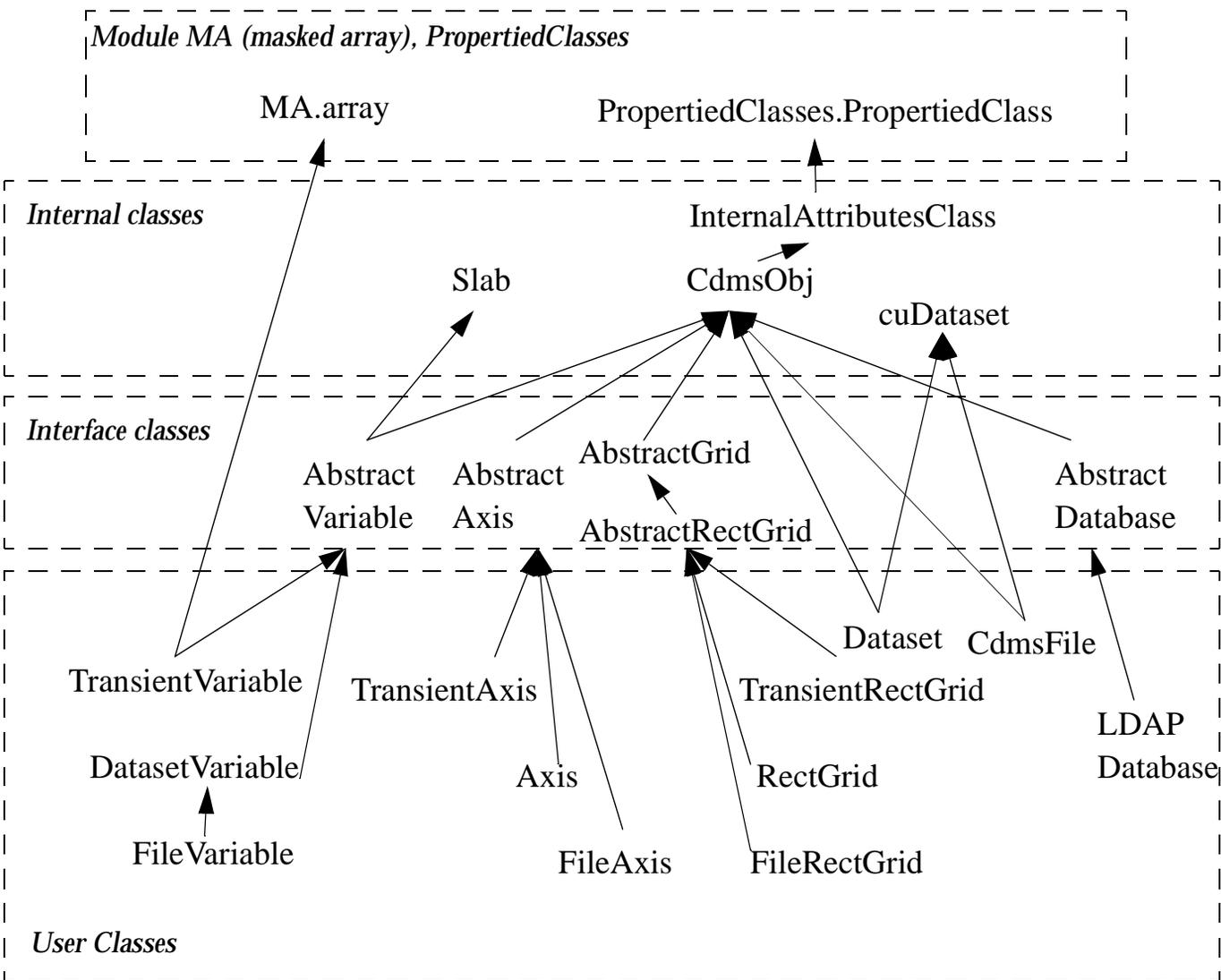
```
cdimport . clt.nc sample
```

---

CDMS classes are grouped into internal, interface, and user classes. Only user classes are meant to be used directly. An interface class defines the common interface of the classes derived from it. For simplicity this document describes the interface classes, and only discusses the derived user classes in the few cases where the interface differs.

Transient classes are not associated with a container such as a Dataset or CdmsFile. They are ‘in-memory’ classes. In contrast, persistent classes such as Axis and DatasetVariable are associated with a container. Modification of an instance of the class persists after the program ends, that is, I/O is generated to a file.

Note that the TransientVariable class is derived from class array in module MA, the masked array class. Masked arrays may be thought of as containing: an array of data, a mask of ones and zeros, and a “fill value”. Complete documentation is available in the Numerical Python package at <http://sourceforge.net/projects/numpy>.



CDMS Classes

---

### *B.1 Version 3.0 Overview*

CDMS version 3.0 is a significant enhancement of previous versions. The major changes were:

1. CDAT/CDMS was integrated with the Numerical Python masked array class `MA.MaskedVariable`. The `MV` submodule was added as a wrapper around `MA`.
2. Methods that read data, such as `subRegion`, `subSlice`, and the slice operations, return instances of class `TransientVariable`. The `plot` and `regrid` modules were modified to handle masked array input. The specifiers `time=...`, `latitude=...`, etc. were added to the I/O routines.
3. The class `TransientVariable` was added.
4. A number of new functions were added, notably `subRegion` and `subSlice`, which return instances of `TransientVariable`.
5. When a masked array is returned from a method, it is “squeezed”: singleton dimensions are removed. In contrast, transient variables are not squeezed. I/O functions have a *squeeze* option. The method `setAutoReshapeMode` was removed.
6. Internal attributes are handled in the `InternalAttributes` class. This allows CDMS classes to be subclassed more readily.
7. The class `Variable` was renamed `DatasetVariable`.

8. The cu module was emulated in cdms. cu and cdms methods can be mixed.
9. The code was modularized, so that Python, CDMS, and Numerical Python can be built and installed separately. This significantly enhances the portability of the code.

---

## *B.2 V3.0 Details*

### **B.2.1 AbstractVariable**

- Functions `getDomain`, `getSlice`, `rank`, `regrid`, `setMissing`, `size`, `subRegion`, and `subSlice` were added.
- The functions `getRegion`, `getSlice`, `getValue`, and the slice operators all return an instance of `MA`, a masked array. Singleton dimensions are squeezed.
- The functions `subRegion` and `subSlice` return an instance of `TransientVariable`. Singleton dimensions are not squeezed.
- The `xxSlice` and `xxRegion` functions have keywords *time*, *level*, *latitude*, and *longitude*.
- The input functions have the keyword *squeeze*.
- `AbstractVariable` inherits from class `Slab`. The following functions previously available in module `cu` are `Slab` methods: `getattribute`, `setattribute`, `listdimattributes`, `getdimattribute`, `listall`, and `info`.
- `AbstractVariable` implements arithmetic functions, `astype`.
- The `write` function was added.

### **B.2.2 AbstractAxis**

- The functions `asComponentTime`, `asRelativeTime`, `clone`, `getAxisIds`, `getAxisIndex`, `getAxisList`, `getAxisListIndex`, `mapIntervalExt` were added.
- `subaxis` was renamed `subAxis` for consistency.
- Generalized wraparound was implemented, to handle multiple cycles, reversing, and negative strides. By default, coordinate intervals are closed. The intersection options `'n'`, `'e'`, `'b'`, and `'s'` were added to the interval indicator - see `mapIntervalExt`.

### **B.2.3 AbstractDatabase**

- The function open is synonymous with openDataset.

### **B.2.4 Dataset**

- The function open is synonymous with openDataset.

### **B.2.5 cdms module**

- The functions asVariable, isVariable, and createVariable were added.
- The function setAutoReshapeMode was removed. It is replaced by the squeeze option for all I/O functions.

### **B.2.6 CdmsFile**

- The function createVariable has a keyword fill\_value. The datatype may be a Numeric/MA typecode.
- The function write was added.

### **B.2.7 CDMSError**

- All errors are an instance of the class CDMSError.

### **B.2.8 AbstractRectGrid**

- The function createGaussianGrid was added.

### **B.2.9 InternalAttributes**

- The class InternalAttributes was added. It has methods add\_internal\_attribute, is\_internal\_attribute, and replace\_external\_attributes.

### **B.2.10 TransientVariable**

- The class TransientVariable was added. It inherits from both AbstractVariable and MA.
- The cdms module function createVariable returns a transient variable.

- This class does not implement the functions `getPaths` or `getTemplate`.

### **B.2.11 MV**

- The MV submodule of `cdms` was added.

---

The *cu* module is the original CDAT I/O interface. As of version 3 it is emulated in the *cdms* module. It is maintained for backward compatibility.

The *cu* classes are *Slab*, corresponding to *TransientVariable* in CDMS, and *cuDataset*, corresponding to *Dataset* in CDMS.

---

*C.1 Slab*

Table C.1 Slab Methods

Type	Definition
Various	<p><b>getattribute(name)</b></p> <p>Get the value of an attribute.</p> <p><i>name</i> is the string name of the attribute. The following special names can always be used: 'filename', 'comments', 'grid_name', 'grid_type', 'time_statistic', 'long_name', 'units'.</p>
Various	<p><b>getdimattribute(dim, field)</b></p> <p>Get the value of a dimension attribute.</p> <p><i>dim</i> is the dimension number, an integer in the range 0..rank-1.</p> <p><i>field</i> is a string, one of: "name", "values", "length", "units", "weights", "bounds".</p>
None	<p><b>info(flag=None, device=sys.stdout)</b></p> <p>Print slab information.</p> <p>If <i>flag</i> is nonzero, dimension values, weights, and bounds are also printed.</p> <p>Output is sent to <i>device</i>.</p>
List	<p><b>listall(all=None)</b></p> <p>Print slab information.</p> <p>If <i>all</i> is nonzero, dimension values, weights, and bounds are also printed.</p>

Table C.1 Slab Methods

Type	Definition
List	<p><b>listdimattributes(dim, field)</b></p> <p>List dimension attributes.</p> <p>Returns a list of string attribute names which can be input to <b>getdimattribute</b>.</p> <p><i>dim</i> is the dimension number, an integer in the range 0..rank-1.</p> <p><i>field</i> is a string, one of: "name", "values", "length", "units", "weights", "bounds".</p>
None	<p><b>setattribute(name, value)</b></p> <p>Set an attribute.</p> <p><i>name</i> is the string name of the attribute.</p> <p><i>value</i> is the value of the attribute.</p>

## C.2 cuDataset

Table C.2 cuDataset Methods

Type	Definition
None	<p><b>cleardefault()</b></p> <p>Clear the default variable name.</p>
None	<p><b>default_variable(vname )</b></p> <p>Set the default variable name.</p> <p><i>vname</i> is the string variable name.</p>

Table C.2 cuDataset Methods

Type	Definition
Array	<p><b>dimensionarray(dname, vname=None)</b></p> <p>Values of the axis named dname.</p> <p><i>dname</i> is the string axis name.</p> <p><i>vname</i> is the string variable name. The default is the variable name set by <b>default_variable</b>.</p>
Axis	<p><b>dimensionobject(dname, vname=None)</b></p> <p>Get an axis.</p> <p><i>dname</i> is the string name of an axis.</p> <p><i>vname</i> is a string variable name. The default is the variable name set by <b>default_variable</b>.</p>
Various	<p><b>getattribute (vname, attribute)</b></p> <p>Get an attribute value.</p> <p><i>vname</i> is a string variable name.</p> <p><i>attribute</i> is the string attribute name.</p>
String	<p><b>getdimensionunits (dname,vname=None)</b></p> <p>Get the units for the given dimension.</p> <p><i>dname</i> is the string name of an axis.</p> <p><i>vname</i> is a string variable name. The default is the variable name set by <b>default_variable</b>.</p>
Various	<p><b>getglobal (attribute)</b></p> <p>Get the value of the global attribute.</p> <p><i>attribute</i> is the string attribute name.</p>

Table C.2 cuDataset Methods

Type	Definition
Variable	<p><b>getslab (vname, *args)</b></p> <p>Read data for a variable.</p> <p><i>vname</i> is the string name of the variable.</p> <p><i>args</i> is an argument list corresponding to the dimensions of the variable. Arguments for each dimension can be:</p> <ol style="list-style-type: none"> <li>(1) ‘:’ or None -- select the entire dimension</li> <li>(2) Ellipsis -- select entire dimensions between the ones given.</li> <li>(3) a pair of successive arguments giving an interval in world coordinates.</li> <li>(4) a CDMS-style tuple of world coordinates e.g. (start, stop, ‘cc’)</li> </ol>
List	<p><b>listall (vname=None, all=None)</b></p> <p>Get info about data from the file.</p> <p><i>vname</i> is the string name of the variable.</p> <p>If <i>all</i> is non-zero, dimension values, weights, and bounds are returned as well.</p>
List	<p><b>listattribute (vname=None )</b></p> <p>Return a list of attribute names.</p> <p><i>vname</i> is the name of the variable. The default is the variable name set by <b>default_variable</b>.</p>
List	<p><b>listdimension (vname=None)</b></p> <p>Return a list of the dimension names associated with a variable.</p> <p><i>vname</i> is the name of the variable. The default is the variable name set by <b>default_variable</b>.</p>
List	<p><b>listglobal ()</b></p> <p>Return a list of the global attribute names.</p>

Table C.2 cuDataset Methods

Type	Definition
List	<p><b>listvariable ()</b></p> <p>Return a list of the variables in the file.</p>
None	<p><b>showall (vname=None, all=None, device=sys.stdout)</b></p> <p>Print a description of the variable.</p> <p><i>vname</i> is the string name of the variable.</p> <p>If <i>all</i> is non-zero, dimension values, weights, and bounds are returned as well.</p> <p>Output is sent to <i>device</i>.</p>
None	<p><b>showattribute (vname=None, device=sys.stdout)</b></p> <p>Print the attributes of a variable.</p> <p><i>vname</i> is the string name of the variable.</p> <p>Output is sent to <i>device</i>.</p>
None	<p><b>showdimension (vname=None, device=sys.stdout)</b></p> <p>Print the dimension names associated with a variable.</p> <p><i>vname</i> is the string name of the variable.</p> <p>Output is sent to <i>device</i>.</p>
None	<p><b>showglobal (device=sys.stdout )</b></p> <p>Print the global file attributes.</p> <p>Output is sent to <i>device</i>.</p>
None	<p><b>showvariable (device=sys.stdout )</b></p> <p>Print the list of variables in the file.</p>

---

# Index

## A

- arange 68
- argsort 69
- arrayrange 68
- asarray 69
- asComponentTime 36
- asRelativeTime 36
- assignValue
  - axis 36
  - variable 85
- astype 85
- asVariable 25
- average 70

## C

- CDML
  - Climate Data Markup Language 135
  - element 136
  - identifier 138
  - tags 136
- cdms module 25
- CdmsFile
  - as a dictionary 45
  - calling as a function 45
- cdscan 150
- choose 70
- clone 85
- close 45
  - database 53
  - dataset 65
- concatenate 70
- connect 53
- copyAxis 46
- copyGrid 46
- count 70
- createAxis
  - cdmsFile 35, 46
  - dataset 35
  - transient 25, 35
- createDataset 44, 62

---

---

- createEqualAreaAxis 26, 35
- createGaussianAxis 26, 35
- createGaussianGrid 26, 73
- createGenericGrid 27, 73
- createGlobalMeanGrid 27, 74
- createRectGrid 74
  - cdmsFile 47, 73
  - dataset 65, 73
  - transient 28, 73
- createUniformGrid 28, 74
- createUniformLatitudeAxis 29, 35
- createUniformLongitudeAxis 29, 35
- createVariable 29, 47, 82, 83, 84
- createVariableCopy 48
- createZonalGrid 29, 74
- crossSectionRegrid 86
- crossSectionRegridder 115

## D

- database 50
- Dataset
  - as a dictionary 65
  - calling as a function 64
- designateCircular 37
- designateLatitude 37
- designateLevel 37
- designateLongitude 37
- designateTime 37
- DRS 154

## G

- getAutoBounds 29
- getAxis 65, 74, 86
- getAxisIds 86
- getAxisIndex 86
- getAxisList 87
- getAxisListIndex 87
- getBounds
  - axis 38
  - grid 75
- getCalendar 38
- getDomain 88
- getGrid 66, 88
- getLatitude

---

---

- grid 75
  - variable 88
- getLevel 88
- getLongitude
  - grid 75
  - variable 88
- getMask 75
- getMissing 88
- getObject 59
- getOrder
  - grid 75
  - variable 89
- getPaths
  - dataset 66
  - variable 89
- getTemplate 90
- getTime 90
- getType 76
- getValue
  - axis 38
- getVariable 66
- getWeights 76
- GRIB 154

## **H**

HDF 153

## **I**

- id 50
- isCircular 38
- isLatitude 39
- isLevel 39
- isLinear 39
- isLongitude 39
- isMaskedVariable 70
- isTime 39
- isVariable 30

## **L**

- len 39, 58, 90
- listDatasets 53

## **M**

- mapInterval 39

---

---

mapIntervalExt 40  
masked\_array 68  
masked\_equal 70  
masked\_greater 70  
masked\_greater\_equal 70  
masked\_less 71  
masked\_less\_equal 71  
masked\_not\_equal 71  
masked\_object 68  
masked\_outside 71  
masked\_values 68  
masked\_where 71  
maximum 71  
minimum 71

## **N**

name alias 154  
netCDF 153

## **O**

ones 68  
open 54, 62  
openDataset 30, 44, 54, 62  
order string 89  
order2index 30  
orderparse 31  
outerproduct 71

## **P**

plot method 130  
power 71  
pressureRegrid 90, 115  
product 71

## **R**

rank 90  
regrid 91  
regrid function 120  
Regridder 116  
relative name 50  
repeat 72  
reshape 69  
resize 69

---

## S

- search result 58
- search result entry 59
- searchFilter 55
- searchPredicate 58
- set\_default\_fill\_value 72
- setAutoBounds 31
- setAxis 91
- setAxisList 91
- setBounds
  - axis 42
  - grid 77
- setCalendar 42
- setClassifyGrids 31
- setMask 77
- setMissing 91
- setType 77
- size 92
- sort 72
- subaxis 43
- subGrid 78
- subGridRegion 79
- subRegion 92
- subSlice 93
- sum 72
- sync
  - cdmsFile 48
  - dataset 66

## T

- tag 50
- take 72
- template Specifiers 63
- transpose 72, 80
- typecode
  - axis 43
  - variable 93

## W

- where 72
- write 49

## X

- XML 135

---

---

**Z**  
zeros 69