The NetCDF CF Library Users Guide and Reference

Version 1.0-alpha5
July 2009

Ed Hartnett
Unidata Program Center

Table of Contents

Summary

The CF conventions for climate and forecast metadata are designed to promote the processing and sharing of files created with the netCDF API.

This library, libcf, makes it easier to create and work with data files which conform to the CF conventions.

The functions of the CF library are intended to be interspersed with netCDF library calls. That is, the programmer will open or create a netCDF file with the netCDF API, and then add or read metadata with libcf library functions, then continue to working with the netCDF API to read and write data.

By using libcf, a data producer can produce files that conform to the CF standards, without having to write netCDF code to create and decode all the attributes that the CF convention uses to store meta-data. A data consumer can use libcf to read any file which conforms to the CF conventions; the file does not need to be created with libcf to be read by libcf.

Fortran-77 wrapper functions provide a Fortran 77 API, just as is done with netCDF itself. A Fortran 90 API is planned, but not yet begun.

For more information about the CF Conventions, see the CF Metadata web site at `http://www.cgd.ucar.edu/cms/eaton/cf-metadata/CF-1.0.html`.

Downloading and Installing libcf

Currently, and for some time to come, libcf is in alpha release. The code is tested, but not extensively. The API may be extended, and possibly changed, in each release.

You must have either netCDF-3 (or netCDF-4) installed. And reasonably recent version of netCDF should work. libcf is tested with netCDF 3.6.2-beta4.

Get the latest version of the libcf tarball at the libcf ftp site: ftp://ftp.unidata.ucar.edu/pub/libcf

Unpack the tarball, and run:

```
./configure --with-netcdf=/my/netcdf --prefix=/my/libcf && make check install > output.txt
```

Where /my/netcdf is the root install directory of the netCDF library you want to use, and /my/libcf is the root directory where you want libcf installed. (They may be the same directory.)

If you wish to use netCDF-4, you must also have HDF5 and libz, the compression library, installed. In this case, configure libcf like this:

```
./configure --with-netcdf=/s/n4_new1/install --enable-netcdf-4 --with-hdf5=/home/ed/local/s && make check install > output.txt
```

If the build does not work for you, please email libcf support: support-libcf@unidata.ucar.edu. Please send the *complete* output of the configure and build output, in ASCII (the output.txt file produced by the above build commands), and the file config.log, which is generated by the configure script.

1 The cfcheck Utility

The cfchck utility will chck a file to see if it contains valid CF metadata. Messages about the file are printed to stdout.

2 Files

2.1 Mark a File as Conforming to CF Conventions

Mark the file as following CF-1.0 conventions.

This functions is called automatically by nccf_def_file, so need not be called by the user if nccf_def_file is being called.

Usage

```
int nccf_def_convention(int ncid);
```

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

2.2 Determine if a File Claims to Conform to CF Conventions

Determine if the file claims to follow CF-1.0 conventions. This function only checks the global "Conventions" attribute. It does not look at file metadata to ensure that this is a well-formed CF file. It only tells whether the file claims to be a CF file.

Usage

```
int nccf_inq_convention(int ncid, int *cf_convention);
```

`ncid` The ncid of the file.

`cf_conventions`
: If this pointer to an int is provided, a 1 is written there if this file claims to follow CF 1.0 conventions, a 0 otherwise.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

2.3 Add description to the data with nccf_def_file

Add some or all of the CF recommended text attributes to a file. Any parameters which are set to NULL will be ignored.

Usage

```
int nccf_def_file(int ncid, char *title, char *history, char *institution,
                  char *source, char *comment, char *references);
```

`ncid` The ncid of the file.

`title` If non-NULL, this text string will be written as the CF-recommended "title" attribute.

`history` If non-NULL, this text string will be written as the CF-recommended "history" attribute.

`institution`
If non-NULL, this text string will be written as the CF-recommended "institution" attribute.

`source` If non-NULL, this text string will be written as the CF-recommended "source" attribute.

`comment` If non-NULL, this text string will be written as the CF-recommended "comment" attribute.

`references`
If non-NULL, this text string will be written as the CF-recommended "references" attribute.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

2.4 Read the description of the data with nccf_inq_file

Read any existing CF recommended text attributes from the file.

Recall that in C, strlens do not include the null terminator. To get the lengths before the strings (in order to allocated) pass NULL for any or all strings and the lengths will be returned. Then call the function again after allocating memory.

The CF version is guaranteed to be less than NC_MAX_NAME.

Any of these pointer arguments may be NULL, in which case it will be ignored.

Usage

```
int nccf_inq_file(int ncid, size_t *title_lenp, char *title,
                  size_t *history_lenp, char *history,
                  size_t *institution_lenp, char *institution,
                  size_t *source_lenp, char *source,
                  size_t *comment_lenp, char *comment,
                  size_t *references_lenp, char *references);
```

`ncid` The ncid of the file.

`title_lenp`
: Pointer to size_t which, if not NULL, will get the length of the title attribute.

`title`
: Pointer to char array which, if not NULL, will get the title string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`history_lenp`
: Pointer to size_t which, if not NULL, will get the length of the history attribute.

`history`
: Pointer to char array which, if not NULL, will get the history string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`institution_lenp`
: Pointer to size_t which, if not NULL, will get the length of the institution attribute.

`institution`
: Pointer to char array which, if not NULL, will get the institution string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`source_lenp`
: Pointer to size_t which, if not NULL, will get the length of the source attribute.

`source`
: Pointer to char array which, if not NULL, will get the source string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`comment_lenp`
: Pointer to size_t which, if not NULL, will get the length of the comment attribute.

`comment`
: Pointer to char array which, if not NULL, will get the comment string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`references_lenp`
: Pointer to size_t which, if not NULL, will get the length of the references attribute.

`references`
: Pointer to char array which, if not NULL, will get the references string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

2.5 Append to a History Attribute

This function appends a time-stamped history string to the history attribute, creating the attribute if it doesn't already exist.

Usage

```
int nccf_add_history(int ncid, const char *history);
```

`ncid` The ncid of the file.

`history` The string to append to the history attribute.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

3 Variables

3.1 Add description to a variable with nccf_def_var

Usage

```
int nccf_def_var(int ncid, int varid, const char *units,
                    const char *long_name, const char *standard_name,
                    int ncoord_vars, int *coord_varids);
```

`ncid` The ncid of the file.

`varid` The varid of the netCDF variable being described.

`units` If non-NULL, this text string will be written as the CF-recommended "units" attribute.

`long_name`
If non-NULL, this text string will be written as the CF-recommended "long_name" attribute.

`standard_name`
If non-NULL, this text string will be written as the CF-recommended "standard_name" attribute.

`ncoord_vars`
Number of coordinate variables for this variable.

`coord_varids`
The variable IDs of the coordinate variables for this variable.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

3.2 Read variable description with nccf_inq_var

Read any existing CF recommended text attributes from a variable.

Usage

```
int nccf_inq_var(int ncid, int varid, size_t *units_lenp, char *units,
                 size_t *long_name_lenp, char *long_name,
                 size_t *standard_name_lenp, char *standard_name,
                 int *ncoord_vars, int *coord_varids);
```

`ncid` The ncid of the file.

`varid` The varid of the netCDF variable.

`units_lenp`
: Pointer to size_t which, if not NULL, will get the length of the units attribute.

`units`
: Pointer to char array which, if not NULL, will get the long_name string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`long_name_lenp`
: Pointer to size_t which, if not NULL, will get the length of the long_name attribute.

`long_name`
: Pointer to char array which, if not NULL, will get the long_name string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`standard_name_lenp`
: Pointer to size_t which, if not NULL, will get the length of the standard_name attribute.

`standard_name`
: Pointer to char array which, if not NULL, will get the standard_name string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`ncoord_vars`
: Pointer to an integer, which, if non-NULL, will get the number of coordinate variables identified in the "coordinates" attribute.

`coord_varids`
: Pointer to an array of integer, which, if non-NULL, will be filled with the variable IDs of the variables listed int he "coordinates" attribute.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

3.3 Define missing data values for a variable with nccf_def_var_missing

nccf_def_notes

This function sets the "fill_value", "valid_min", and "valid_max" attributes.

Usage

```
int nccf_def_var_missing(int ncid, int varid, const void *fill_valuep,
                         const void *valid_minp, const void *valid_maxp);
```

`ncid` The ncid of the file.

`varid` The varid of the netCDF variable being described.

`fill_valuep`
: If non-NULL, this will point to a value of the same type as this varid, which will be used as the fill_value for the data.

`valid_minp`
: If non-NULL, this will point to a value of the same type as this varid, which will be written as the “valid_min” attribute. If this parameter is non-NULL, valid_max must also be provided.

`valid_maxp`
: If non-NULL, this will point to a value of the same type as this varid, which will be written as the “valid_max” attribute. If this parameter is non-NULL, valid_min must also be provided.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.4 Learn about missing data settings with nccf_inq_var_missing

Get attributes which define missing data information. If the attributes are not there, then provide the valid data anyway, based on netCDF defaults.

Usage

```
int nccf_inq_var_missing(int ncid, int varid, void *fill_valuep,
                         void *valid_minp, void *valid_maxp);
```

`ncid` The ncid of the file.

`varid` The varid of the netCDF variable.

`fill_valuep`
: If this is not NULL, the fill value of the variable will be written at this address by nccf_inq_var_missing. If the fill value was not defined for the variable, the netCDF default value will be used.

`valid_minp`
: If this is not NULL, the valid_min value of the variable will be written at this address by nccf_inq_var_missing. If the valid_min was not defined for the variable, the netCDF default value will be used.

`valid_maxp`
: If this is not NULL, the valid_max value of the variable will be written at this address by nccf_inq_var_missing. If the valid_max value was not defined for the variable, the netCDF default value will be used.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.5 Add CF-Recommended Metadata to a File or Variable with

nccf_def_notes

This functions writes up to four text attributes for either a variable, or an entire file. These text attributes, “institution,” “source,” “comment,” and “references” are recommended by the CF Metadata Convention.

Usage

```
int nccf_def_notes(int ncid, int varid, char *units, char *long_name,
                   char *standard_name, char *institution,
                   char *source, char *comment, char *references);
```

`ncid` The ncid of the file.

`varid` The varid of the netCDF variable being described. Use NC_GLOBAL if you wish these attributes to apply to the entire file.

`institution`
If non-NULL, this text string will be written as the CF-recommended “institution” attribute.

`source` If non-NULL, this text string will be written as the CF-recommended “source” attribute.

`comment` If non-NULL, this text string will be written as the CF-recommended “comment” attribute.

`references`
If non-NULL, this text string will be written as the CF-recommended “references” attribute.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

3.6 Read variable description with nccf_inq_notes

Read any existing CF recommended text attributes from a variable.

Usage

```
int nccf_inq_notes(int ncid, int varid,
                   size_t *institution_lenp, char *institution,
                   size_t *source_lenp, char *source,
                   size_t *comment_lenp, char *comment,
                   size_t *references_lenp, char *references);
```

`ncid` The ncid of the file.

`varid` The varid of the netCDF variable.

`institution_lenp`
Pointer to size_t which, if not NULL, will get the length of the institution attribute.

`institution`
Pointer to char array which, if not NULL, will get the institution string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`source_lenp`
Pointer to size_t which, if not NULL, will get the length of the source attribute.

`source` Pointer to char array which, if not NULL, will get the source string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`comment_lenp`
Pointer to size_t which, if not NULL, will get the length of the comment attribute.

`comment` Pointer to char array which, if not NULL, will get the comment string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

`references_lenp`
Pointer to size_t which, if not NULL, will get the length of the references attribute.

`references`
Pointer to char array which, if not NULL, will get the references string. Memory must be allocated before this function is called. Call this function with a NULL for this parameter to get the size first.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

3.7 Define a coordinate variable and dimension for latitude with nccf_def_latitude

Define a coordinate variable and dimension with all the CF recommended attribute accessories for latitude.

Usage

```
int nccf_def_latitude(int ncid, size_t len, nc_type xtype,
                      const char *formula_terms, int *lat_dimidp,
                      int *lat_varidp);
```

ncid The ncid of the file.

len The length of this coordinate dimension.

xtype The type of this coordinate variable.

formula_terms
: If non-NULL, a string which will be written as the "formula_terms" attribute on the coordinate variable.

lat_dimidp
: If non-NULL, nccf_def_latitude will write the dimension ID of the netCDF dimension for the latitude here.

lat_varidp
: If non-NULL, nccf_def_latitude will write the variable ID of the netCDF coordinate variable for the latitude here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.8 Learn about a latitude coordinate variable and dimension with

nccf_inq_latitude

Inquire about a latitude dimension and coordinate variable.

Usage

```
int nccf_inq_latitude(int ncid, size_t *lenp, nc_type *xtypep,
                      size_t *ft_lenp, char *formula_terms,
                      int *lat_dimidp, int *lat_varidp);
```

ncid The ncid of the file.

lenp If non-NULL, the length of the latitude dimension will be written here by nccf_inq_latitude.

xtypep If non-NULL, the type of the coordinate variable will be written here.

ft_lenp If non-NULL, the length of the value of the "formula_terms" attribute will be written here by ft_lenp. If there is no "formula_terms" attribute, zero will be written.

lat_dimidp
: If non-NULL, the dimid of the latitude dimension will be written here.

lat_varidp
: If non-NULL, the varid of the latitude coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.9 Define a coordinate variable and dimension for longitude with nccf_def_longitude

Define a coordinate variable and dimension with all the CF recommended attribute accessories for longitude.

Usage

```
int nccf_def_longitude(int ncid, size_t len, nc_type xtype,
                       const char *formula_terms, int *lon_dimidp,
                       int *lon_varidp);
```

`ncid` The ncid of the file.

`len` The length of this coordinate dimension.

`xtype` The type of this coordinate variable.

`formula_terms`
: If non-NULL, a string which will be written as the "formula_terms" attribute on the coordinate variable.

`lon_dimidp`
: If non-NULL, nccf_def_longitude will write the dimension ID of the netCDF dimension for the longitude here.

`lon_varidp`
: If non-NULL, nccf_def_longitude will write the variable ID of the netCDF coordinate variable for the longitude here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.10 Learn about a longitude coordinate variable and dimension with

nccf_inq_longitude

Inquire about a longitude dimension and coordinate variable.

Usage

```
int nccf_inq_longitude(int ncid, size_t *lenp, nc_type *xtypep,
                         size_t *ft_lenp, char *formula_terms,
                         int *lon_dimidp, int *lon_varidp);
```

`ncid` The ncid of the file.

`lenp` If non-NULL, the length of the longitude dimension will be written here by nccf_inq_longitude.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`ft_lenp` If non-NULL, the length of the value of the "formula_terms" attribute will be written here by ft_lenp. If there is no "formula_terms" attribute, zero will be written.

`lon_dimidp`
If non-NULL, the dimid of the longitude dimension will be written here.

`lon_varidp`
If non-NULL, the varid of the longitude coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.11 Define a coordinate variable and dimension for level with nccf_def_lvl

Define a coordinate variable and dimension with all the CF recommended attribute accessories for a vertical level.

Usage

```
int nccf_def_lvl(int ncid, const char *name, size_t len, nc_type xtype,
                 const char *units, int positive_up, const char *standard_name,
                 const char *long_name, const char *formula_terms,
                 int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` The name of the coordinate dimension and variable.

`len` The length of this coordinate dimension and variable.

`xtype` The type of this coordinate variable.

`units` If non-NULL, a string which will be written as the "units" attribute on the coordinate variable.

`positive_up`
Set to 0 and the attribute "positive" to "down". Set to any other value to get "up".

`standard_name`
: If non-NULL, a string which will be written as the "standard_name" attribute on the coordinate variable.

`long_name`
: If non-NULL, a string which will be written as the "long_name" attribute on the coordinate variable.

`lon_dimidp`
: If non-NULL, nccf_def_lvl will write the dimension ID of the netCDF dimension for the level here.

`lon_varidp`
: If non-NULL, nccf_def_lvl will write the variable ID of the netCDF coordinate variable for the level here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.12 Learn about a level coordinate variable and dimension with

nccf_inq_lvl

Inquire about a level dimension and coordinate variable.

Usage

```
int nccf_inq_lvl(int ncid, char *name, size_t *lenp, nc_type *xtypep,
                 size_t *ft_lenp, char *formula_terms, int *positive_upp,
                 int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, the name of this vertical level dimension (and variable) will be written here.

`lenp` If non-NULL, the length of the level dimension will be written here by nccf_inq_lvl.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`ft_lenp` If non-NULL, the length of the value of the "formula_terms" attribute will be written here by ft_lenp. If there is no "formula_terms" attribute, zero will be written.

`positive_upp`
: If non-NULL, a one will be written here if the "positive" attribute of this coordinate variable is "up", a zero will be written if it is "down".

`lvl_dimidp`
: If non-NULL, the dimid of the level dimension will be written here.

`lvl_varidp`
: If non-NULL, the varid of the level coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.13 Define a coordinate variable and dimension for level with nccf_def_lvl

Define a unitless vertical coordinate variable and dimension from Apendix D of the CF Convention, with all the CF recommended attribute accessories for a vertical level.

Usage

```
int nccf_def_lvl_vert(int ncid, int lvl_type, const char *name, nc_type xtype,
                      size_t len, int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`lvl_type` One of: CF_VERT_ATM_LN, CF_VERT_SIGMA, CF_VERT_HYBRID_SIGMA, CF_VERT_HYBRID_HEIGHT, CF_VERT_SLEVE, CF_VERT_OCEAN_SIGMA, CF_VERT_OCEAN_S, CF_VERT_OCEAN_SIGMA_Z, CF_VERT_OCEAN_DBL_SIGMA.

`name` The name of the coordinate dimension and variable.

`xtype` The type of this coordinate variable.

`len` The length of this coordinate dimension and variable.

`lvl_dimidp`
: If non-NULL, the function will write the dimension ID of the netCDF dimension for the level here.

`llvl_varidp`
: If non-NULL, the function will write the variable ID of the netCDF coordinate variable for the level here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.14 Learn about a level coordinate variable and dimension with

nccf_inq_lvl

Inquire about a level dimension and coordinate variable.

Usage

```
int nccf_inq_lvl_vert(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                      int *lvl_typep, int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, the name of this vertical level dimension (and variable) will be written here.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the level dimension will be written here.

`lvl_typep`
If non-NULL, the type of vertical dimension will be written here, one of CF_VERT_ATM_LN, CF_VERT_SIGMA, CF_VERT_HYBRID_SIGMA, CF_VERT_HYBRID_HEIGHT, CF_VERT_SLEVE, CF_VERT_OCEAN_SIGMA, CF_VERT_OCEAN_S, CF_VERT_OCEAN_SIGMA_Z, CF_VERT_OCEAN_DBL_SIGMA.

`lvl_dimidp`
If non-NULL, the dimid of the level dimension will be written here.

`lvl_varidp`
If non-NULL, the varid of the level coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.15 Define a coordinate variable and dimension for time with nccf_def_time

Define a coordinate variable and dimension with all the CF recommended attribute accessories for time.

Usage

```
int nccf_def_time(int ncid, const char *name, size_t len, nc_type xtype,
                  const char *units, const char *standard_name, const char *long_name,
                  const char *formula_terms, int *time_dimidp, int *time_varidp);
```

`ncid` The ncid of the file.

`len` The length of this coordinate dimension.

`xtype` The type of this coordinate variable.

`formula_terms`
If non-NULL, a string which will be written as the "formula_terms" attribute on the coordinate variable.

`lon_dimidp`
: If non-NULL, nccf_def_time will write the dimension ID of the netCDF dimension for the time here.

`lon_varidp`
: If non-NULL, nccf_def_time will write the variable ID of the netCDF coordinate variable for the time here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.16 Learn about a time coordinate variable and dimension with

nccf_inq_time

Inquire about a time dimension and coordinate variable.

Usage

```
int nccf_inq_time(int ncid, size_t *lenp, nc_type *xtypep, size_t *ft_lenp,
                     char *formula_terms, int *time_dimidp, int *time_varidp);
```

`ncid` The ncid of the file.

`lenp` If non-NULL, the length of the time dimension will be written here by nccf_inq_time.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`ft_lenp` If non-NULL, the length of the value of the “formula_terms” attribute will be written here by ft_lenp. If there is no “formula_terms” attribute, zero will be written.

`formula_terms`
: If non-NULL the value of the “formula_terms” attribute, if any, will be copied here.

`time_dimidp`
: If non-NULL, the dimid of the time dimension will be written here.

`time_varidp`
: If non-NULL, the varid of the time coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.17 Define atmosphere sigma coordinate.

Define formula terms attribute for atmosphere sigma coordinate variable.

Usage

```
int nccf_def_ft_sigma(int ncid, int varid, int ps_varid, int ptop_varid);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`ps_varid` The variable ID of the ps variable.

`ptop_varid`
The variable ID of the ptop variable.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.18 Inquire about atmosphere sigma coordinate.

nccf_inq_time

Inquire about atmosphere sigma coordinate.

Usage

```
int nccf_inq_lvl_sigma(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                       int *ps_varidp, int *ptop_varidp, int *lvl_dimidp,
                       int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, this pointer gets the name of the coordinate variable and dimension.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the coordinate dimension will be written here.

`ps_varidp`
If non-NULL, the variable ID of the ps variable will be written here.

`ptop_varidp`
If non-NULL, the variable ID of the ptop variable will be written here.

`lvl_dimidp`
If non-NULL, the dimid of the coordinate dimension will be written here.

`lvl_varidp`
If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.19 Define atmosphere sigma coordinate.

Define formula terms attribute for atmosphere hybrid sigma coordinate variable.

Usage

```
int nccf_def_ft_hybrid_sigma(int ncid, int varid, int a_varid, int b_varid,
                             int ps_varid, int p0_varid);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`a_varid` The variable ID of the a variable.

`b_varid` The variable ID of the b variable.

`ps_varid` The variable ID of the ps variable.

`ptop_varid`
The variable ID of the ptop variable.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.20 Inquire about atmosphere sigma coordinate.

Inquire about atmosphere hybrid sigma coordinate.

Usage

```
int nccf_inq_lvl_hybrid_sigma(int ncid, char *name, nc_type *xtypep,
                              size_t *lenp, int *a_varidp, int *b_varidp,
                              int *ps_varidp, int *p0_varidp, int *lvl_dimidp,
                              int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, this pointer gets the name of the coordinate variable and dimension.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the coordinate dimension will be written here.

`a_varidp` If non-NULL, the variable ID of the a variable will be written here.

`b_varidp` If non-NULL, the variable ID of the b variable will be written here.

`ps_varidp`
If non-NULL, the variable ID of the ps variable will be written here.

`ptop_varidp`
If non-NULL, the variable ID of the ptop variable will be written here.

`lvl_dimidp`
If non-NULL, the dimid of the coordinate dimension will be written here.

`lvl_varidp`
If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.21 The formula_terms attribute for atmosphere hybrid height

Define formula terms attribute for atmosphere hybrid height coordinate variable

Usage

```
int nccf_def_ft_hybrid_height(int ncid, int varid, int a_varid, int b_varid,
                              int orog_varid);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`a_varid` The variable ID of the a variable.

`b_varid` The variable ID of the b variable.

`orog_varid`
The variable ID of the orog variable.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.22 Inquire about hybrid height coordinate.

Inquire about the hybrid height coordinate.

Usage

```
int nccf_inq_lvl_hybrid_height(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                               int *a_varidp, int *b_varidp, int *orog_varidp,
                               int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, this pointer gets the name of the coordinate variable and dimension.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the coordinate dimension will be written here.

`a_varidp` If non-NULL, the variable ID of the a variable will be written here.

`b_varidp` If non-NULL, the variable ID of the b variable will be written here.

`orog_varidp`
If non-NULL, the variable ID of the orog variable will be written here.

`lvl_dimidp`
If non-NULL, the dimid of the coordinate dimension will be written here.

`lvl_varidp`
If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.23 Define atmosphere sleve coordinate.

Define formula terms attribute for the atmosphere sleve coordinate variable.

Usage

```
int nccf_def_ft_sleve(int ncid, int varid, int a_varid, int b1_varid,
                      int b2_varid, int ztop_varid, int zsurf1_varid,
                      int zsurf2_varid);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`a_varid` The variable ID of the a variable.

`b1_varid` The variable ID of the b1 variable.

`b2_varid` The variable ID of the b2 variable.

`ztop_varid`
The variable ID of the ztop variable.

zsurf1_varid
: The variable ID of the zsurf1 variable.

zsurf2_varid
: The variable ID of the zsurf2 variable.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.24 Inquire About Sleve Coordinate.

Inquire about the sleve coordinate.

Usage

```
int nccf_inq_lvl_sleve(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                       int *a_varidp, int *b1_varidp, int *b2_varidp, int *ztop_varidp,
                       int *zsurf1_varidp, int *zsurf2_varidp, int *lvl_dimidp,
                       int *lvl_varidp);
```

ncid
: The ncid of the file.

name
: If non-NULL, this pointer gets the name of the coordinate variable and dimension.

xtypep
: If non-NULL, the type of the coordinate variable will be written here.

lenp
: If non-NULL, the length of the coordinate dimension will be written here.

a_varidp
: If non-NULL, the variable ID of the a variable will be written here.

b1_varidp
: If non-NULL, the variable ID of the b1 variable will be written here.

b2_varidp
: If non-NULL, the variable ID of the b2 variable will be written here.

ztop_varidp
: If non-NULL, the variable ID of the ztop variable will be written here.

zsurf1_varidp
: If non-NULL, the variable ID of the zsurf1 variable will be written here.

zsurf2_varidp
: If non-NULL, the variable ID of the zsurf2 variable will be written here.

lvl_dimidp
: If non-NULL, the dimid of the coordinate dimension will be written here.

lvl_varidp
: If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.25 Define Ocean Sigma Coordinate.

Define the formula terms attribute for the ocean sigma coordinate variable.

Usage

```
int nccf_def_ft_ocean_sigma(int ncid, int varid, int eta_varid, int depth_varid);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`eta_varid`
The variable ID of the eta variable.

`depth_varid`
The variable ID of the depth variable.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.26 Inquire About Ocean Sigma Coordinate.

Inquire about the ocean sigma coordinate.

Usage

```
int nccf_inq_lvl_ocean_sigma(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                             int *eta_varidp, int *depth_varidp, int *lvl_dimidp,
                             int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, this pointer gets the name of the coordinate variable and dimension.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the coordinate dimension will be written here.

`eta_varidp`
If non-NULL, the variable ID of the eta variable will be written here.

`lvl_dimidp`
If non-NULL, the dimid of the coordinate dimension will be written here.

`lvl_varidp`
If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.27 Define Ocean S Coordinate.

Define ocean s coordinate.

Usage

```
int nccf_def_ft_ocean_s(int ncid, const char *name, nc_type xtype, size_t len,
                         int eta_varid, int depth_varid, int a_varid, int b_varid,
                         int depth_c_varid, int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`xtype` The type of this coordinate variable.

`len` The length of this coordinate dimension.

`eta_varid`
The variable ID of the eta variable.

`depth_varid`
The variable ID of the depth variable.

`a_varid` The variable ID of the a variable.

`b_varid` The variable ID of the b variable.

`depth_c_varid`
The variable ID of the depth_c variable.

`lvl_dimidp`
If non-NULL, the dimension ID of the netCDF coordinate dimension will be written here.

`lvl_varidp`
If non-NULL, the variable ID of the netCDF coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.28 Inquire About Ocean S Coordinate.

nccf_inq_time

Inquire about the ocean s coordinate.

Usage

```
int nccf_inq_lvl_ocean_s(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                         int *eta_varidp, int *depth_varidp, int *a_varidp, int *b_varidp,
                         int *depth_c_varidp, int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, this pointer gets the name of the coordinate variable and dimension.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the coordinate dimension will be written here.

`eta_varidp`
If non-NULL, the variable ID of the eta variable will be written here.

`depth_varidp`
If non-NULL, the variable ID of the depth variable will be written here.

`a_varidp` If non-NULL, the variable ID of the a variable will be written here.

`b_varidp` If non-NULL, the variable ID of the b variable will be written here.

`depth_c_varidp`
If non-NULL, the variable ID of the depth_c variable will be written here.

`lvl_dimidp`
If non-NULL, the dimid of the coordinate dimension will be written here.

`lvl_varidp`
If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.29 Define Ocean Sigma Z Coordinate.

Define ocean sigma z coordinate.

Usage

```
int nccf_def_ft_ocean_sigma_z(int ncid, const char *name, nc_type xtype,
                              size_t len, int eta_varid, int depth_varid,
                              int depth_c_varid, int nsigma_varid,
                              int zlev_varid, int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`xtype` The type of this coordinate variable.

`len` The length of this coordinate dimension.

`eta_varid`
The variable ID of the eta variable.

`depth_varid`
The variable ID of the depth variable.

`depth_c_varid`
The variable ID of the depth_c variable.

`nsigma_varid`
The variable ID of the nsigma variable.

`zlev_varid`
The variable ID of the zlev variable.

`lvl_dimidp`
If non-NULL, the dimension ID of the netCDF coordinate dimension will be written here.

`lvl_varidp`
If non-NULL, the variable ID of the netCDF coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.30 Inquire About Ocean Sigma Z Coordinate.

Inquire about the ocean sigma z coordinate.

Usage

```
int nccf_inq_lvl_ocean_sigma_z(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                               int *eta_varidp, int *depth_varidp, int *depth_c_varidp,
                               int *nsigma_varidp, int *zlev_varidp, int *lvl_dimidp,
                               int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, this pointer gets the name of the coordinate variable and dimension.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the coordinate dimension will be written here.

`eta_varidp`
If non-NULL, the variable ID of the eta variable will be written here.

`depth_varidp`
If non-NULL, the variable ID of the depth variable will be written here.

`depth_c_varidp`
: If non-NULL, the variable ID of the depth_c variable will be written here.

`nsigma_varidp`
: If non-NULL, the variable ID of the nsigma variable will be written here.

`zlev_varidp`
: If non-NULL, the variable ID of the zlev variable will be written here.

`lvl_dimidp`
: If non-NULL, the dimid of the coordinate dimension will be written here.

`lvl_varidp`
: If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.31 Define Ocean Double Sigma Coordinate.

Define ocean double sigma coordinate.

Usage

```
int nccf_def_ft_ocean_dbl_sigma(int ncid, const char *name, nc_type xtype, size_t len,
                                int depth_varid, int z1_varid, int z2_varid, int a_varid,
                                int href_varid, int k_c_varid, int *lvl_dimidp,
                                int *lvl_varidp);
```

`ncid` The ncid of the file.

`varid` The varid of the vertical coordinate variable.

`xtype` The type of this coordinate variable.

`len` The length of this coordinate dimension.

`depth_varid`
: The variable ID of the depth variable.

`z1_varid` The variable ID of the z1 variable.

`z2_varid` The variable ID of the z2 variable.

`a_varid` The variable ID of the a variable.

`href_varid`
: The variable ID of the href variable.

`k_c_varid`
: The variable ID of the k_c variable.

`lvl_dimidp`
: If non-NULL, the dimension ID of the netCDF coordinate dimension will be written here.

`lvl_varidp`
: If non-NULL, the variable ID of the netCDF coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.32 Inquire About Ocean Double Sigma Coordinate.

Inquire about the ocean double sigma coordinate.

Usage

```
int nccf_inq_lvl_ocean_dbl_sigma(int ncid, char *name, nc_type *xtypep, size_t *lenp,
                                 int *depth_varidp, int *z1_varidp, int *z2_varidp,
                                 int *a_varidp, int *href_varidp, int *k_c_varidp,
                                 int *lvl_dimidp, int *lvl_varidp);
```

`ncid` The ncid of the file.

`name` If non-NULL, this pointer gets the name of the coordinate variable and dimension.

`xtypep` If non-NULL, the type of the coordinate variable will be written here.

`lenp` If non-NULL, the length of the coordinate dimension will be written here.

`depth_varidp`
: If non-NULL, the variable ID of the depth variable will be written here.

`z1_varidp`
: If non-NULL, the variable ID of the z1 variable will be written here.

`z2_varidp`
: If non-NULL, the variable ID of the z2 variable will be written here.

`a_varidp` If non-NULL, the variable ID of the a variable will be written here.

`href_varidp`
: If non-NULL, the variable ID of the href variable will be written here.

`k_c_varidp`
: If non-NULL, the variable ID of the k_c variable will be written here.

`lvl_dimidp`
: If non-NULL, the dimid of the coordinate dimension will be written here.

`lvl_varidp`
: If non-NULL, the varid of the coordinate variable will be written here.

Return Codes

This function returns zero for success, or an error code for failure.

Example

3.33 Get a geographic subset of the data.

Get a geographic subset of the data.

Usage

```
int nccf_get_vara(int ncid, int varid, float *lat_bounds, int *nlat, float *lon_bounds,
  int *nlon, int *lvl_index, int *nlvl, int rec, void *data);
```

`ncid` The ncid of the file.

`varid` The varid of the data variable from which the subset will be taken.

`lat_bounds`
A length two array, this holds the latitude start and stop values for the range of interest.

`nlat` A pointer to an integer which will get the number of latitude values which fall within the range.

`lon_bounds`
A length two array, this holds the longitude start and stop values for the range of interest. (Wrapping around the dateline is not allowed!)

`nlon` A pointer to an integer which will get the number of longitude values which fall within the range.

`lvl_index`
A zero-based index number for the verticle level of interest to the subsetter. (Ignored if data has no vertical axis).

`timestep` A zero-based index number for the timestep of interest to the subsetter. (Ignored if data has no time axis).

`data` A pointer to which the data subset will be written. Memory must be allocated (and deallocated) by the user.

Return Codes

This function returns zero for success, or an error code for failure.

Example

4 Coordinate Systems

4.1 Label the axis type of a coordinate var with nccf_def_axis_type

Usage

```
int
nccf_def_axis_type(int ncid, int varid, int axis_type);
```

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

4.2 Find out the axis type of a coordinate var with nccf_inq_axis_type

Usage

```
int
nccf_inq_axis_type(int ncid, int varid, int *axis_type);
```

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

4.3 Define a coordinate system with nccf_def_coord_system

Usage

```
int
nccf_def_coord_system(int ncid, const char *name, int naxes, int *axis_varids,
                      int *system_varid);
```

Define a coordinate system consisting of naxes axes, each axis represented by a coordinate varid in the axis_varids array. This create a new (scalar, NC_CHAR) var, whose varid is returned in system_varid.

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

4.4 Find out about a coordinate system with nccf_inq_coord_system

Usage

```
int
nccf_inq_coord_system(int ncid, int system_varid, char *name,
                      int *naxes, int *axis_varids);
```

Find out about a coordinate system, it's name, number of axes, and the varid of each axis coordinate var.

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

4.5 Assign a coordinate system to a var with nccf_assign_coord_system

Usage

```
int
nccf_assign_coord_system(int ncid, int varid, int system_varid);
```

Assign a coordinate system to a var. This adds an attribute to the var.

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

4.6 Define a coordinate transform with nccf_def_transform

Usage

```
int
nccf_def_transform(int ncid, const char *name, const char *transform_type,
```

```
                      const char *transform_name, int *transform_varid);
```

Define a coordinate transform. This adds a (scalar, NC_CHAR) var, which contains some attributes. The varid of this new variable is returned in transform_varid.

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

4.7 Find out about a coordinate transform with nccf_inq_transform

Usage

```
int
nccf_inq_transform(int ncid, int transform_varid, char *name, size_t *type_len,
                 char *transform_type, size_t *name_len, char *transform_name);
```

Find out about a coordinate transform, it's name, and the contents of the transform_type and transform_name attributes. Pass NULL for any that you're not interested in. Pass NULL for transform_type and transform_name to get their lengths with type_len and name_len.

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

4.8 Assign a coordinate transform to a coordinate system with nccf_assign_transform

Usage

```
int
nccf_assign_transform(int ncid, int system_varid, int transform_varid);
```

Assign a coordinate transform to a coordinate system. This adds an attribute to the variable that holds the coordinate system attributes.

`ncid` The ncid of the file.

Return Codes

This function returns zero for success, or a netCDF error code for failure.

Example

Index

A

S