NetCDF

NetCDF is a binary format for storing arrays of data. For a full list and details of functions and operators on vectors, see NetCDF Functions.

NetCDF

NetCDF stands for Network Common Data Form. NetCDF is an interface for array-oriented data access and a freely-distributed collection of software libraries for C, Fortran, C++, Java, and perl that provide implementations of the interface. The netCDF libraries define a machine-independent format for representing scientific data. Together, the interface, libraries, and format support the creation, access, and sharing of scientific data. For full details on NetCDF, see http://www.unidata.ucar.edu/software/netcdf/ .

NetCDF can be used to handle a great number of scientific data types. Metview uses NetCDF as an internal format for representing data units which can't be conveniently represented by GRIB, BUFR or geopoints, for example, to handle data arrays representing cross-sections and cross-sectional averages, vertical profiles, Hovmøller matrices, etc. Such NetCDF data files can then be imported by other NetCDF conversant software if users so wish.

Metview provides a set of NetCDF handling macro functions, allowing users to apply a number of functions and operators to NetCDF data - see NetCDF Functions.

How operators and functions work on NetCDF

NetCDF files can contain a wide variety of data. They can contain a number of data units, e.g. a set of cross section plots, a set of 2D geographical grids, a set of time series or vertical profiles, etc. Each component of a set is stored in the netCDF file as a separate *netcdf variable* . The handling of and computations with netCDF files are based on the concept of *current variable*.

Out of the N variables contained in a netCDF file, one is always set to be the current variable. *Functions and operators acting upon the netcdf file will act only upon the current variable* .

In general, users set the current variable (which by default is the first variable in the file) to one of those contained in the file and can then apply a number of functions and operators to them :

```
setcurrent(netcdf1, 6) # use the 6th variable in this netCDF file
setcurrent(netcdf2, 2) # use the 2nd variable in this netCDF file
netcdf3 = netcdf1 op netcdf2
out = function(netcdf1, ...)
```

However, because functions and operators work only on the current variable, when the netCDF contains more than one variable (a multi-variable netCDF), special care must be taken when you need the operator/function to apply to all variables - this is detailed later in this page.

When two netCDF variables are involved, *both files have to have the same number of data points* in each current variable, as an operation between two netCDFs is carried out between each pair of corresponding data values. Thus :

nc3 = nc1+nc2	corresponds to	for each data value i nc3i = nc2i + nc1i
nc2 = nc1+a	corresponds to	for each data value i nc1i = nc1i + a
nc2 = f(nc1)	corresponds to	for each data value i nc2i = f(nc1i)

NOTE : Like fieldsets, a netCDF resulting from an operation on two other netCDFs, will take the metadata (e.g. date, time, parameter, levels, ...) from the first netCDF.

Scaled values (Metview 5)

By default, Metview will apply any **scale_factor** and **add_offset** attributes for the current variable. This behaviour can be toggled using the function `netcdf_auto_scale_values()`.

Missing values (Metview 5)

By default, Metview will check the **_FillValue** attribute of the current variable and will not include any such values in its calculations. This behaviour can be toggled using the function `netcdf_preserve_missing_values()`.

Working with multi-variable NetCDF files

Functions and operators working on netCDF files apply only to the current variable. When the netCDF file contains several variables, you need to address each variable separately and explicitly, and apply the function /operator to each in turn. For this purpose, Metview Macro provides functions to query the contents of a netCDF file and to set one of its variables to be the current variable.

Users can list the variables held in a netcdf variable by means of the function `variables()` :

```
var_list = variables(netcdf)
```

This returns a list of strings, each holding a variable name. Counting the number of elements in the output list gives the number of variables.

To set one of the existing variables to be the current variable, use function `setcurrent()`, which takes the index (starting at 1) of the desired variable:

```
setcurrent(netcdf, n)
```

The two functions above provide the basic framework to operate on multi-variable netcdf files.

Example 1 : To operate on a netcdf file which you want to overwrite with the new results

```
# Derive a cross section of temperature data in a netcdf variable
(...)

# derive the list of netcdf variables
var_list = variables(temp_xs)

# loop over variables and subtract scalar
for i = 1 to count(var_list) do
    setcurrent(temp_xs, i)
    temp_xs = temp_xs - 273.15 # acts on current variable only
end for
```

Example 2 : To operate on two netcdf files, assigning the result to a third netcdf, you should create the output netcdf first by a simple copying of one of the input netcdfs :

```
# Derive cross sections of temperature forecast and analysis
# in two netcdf variables, tfc_xs and tan_xs....

(...)

# derive the list of netcdf variables
var_list = variables(tfc_xs)

# create output netcdf
diff_xs = tfc_xs

# loop over variables and create fc-an difference cross-section
for i = 1 to count(var_list) do
    setcurrent(tan_xs, i)
    setcurrent(tfc_xs, i)
    setcurrent(diff_xs, i)
    diff_xs = tfc_xs - tan_xs
end for
```

Extracting NetCDF values

If you need to have access to the data values of a netcdf current variable, you can use function `values()` :

```
val_list = values(netcdf)
```

which returns a vector with all the values for the current variable.You can then use the Vector functions to manipulate the data. This technique could even be used to create a new Geopoints variable with the netCDF data, or to insert the values into a GRIB field.

An alternative method for accessing individual values is to the use the function `value()` :

```
val = value(netcdf, n)
```

which returns the n*th* value from the current netcdf variable.

Time variables (Metview 5)

Variables which are detected to be of 'time' type (e.g. attribute standard_name='time', plus other checks) are, by default, retrieved by the `value()` and `values()` functions as a date or a list of dates. This behaviour can be toggled calling the function `netcdf_auto_translate_times()` with an argument of 1 or 0 to activate/deactivate the translation to dates. If deactivated, the `value()` and `values()` functions will instead return a number or a vector of raw numbers as they are encoded in the variable.

Automatic rescaling of values (Metview 5)

If Metview performs a computation on a variable which results in its values overflowing the data type used to pack the values in the netCDF variable (e.g. adding 3000 to the values of a Byte variable), **and** the variable has **scale_factor** and **add_offset** attributes, Metview will compute new **scale_factor** and **add_offset** attributes so that the values can be packed within the data type, making best use of its data range. In this case, the **_FillValue** may be modified too.
This rescaling will not be performed if these attributes are not defined for the current variable, or if Metview has been instructed to ignore them (via the `netcdf_auto_scale_values()` function).
The automatic rescaling behaviour can be toggled on or off via the `netcdf_auto_rescale_values_to_fit_packed_type()` function. If disabled, and the computed values overflow the data type, the macro will fail.