Meteorologisk institutt
met.no

Fimex Introduction

Heiko Klein

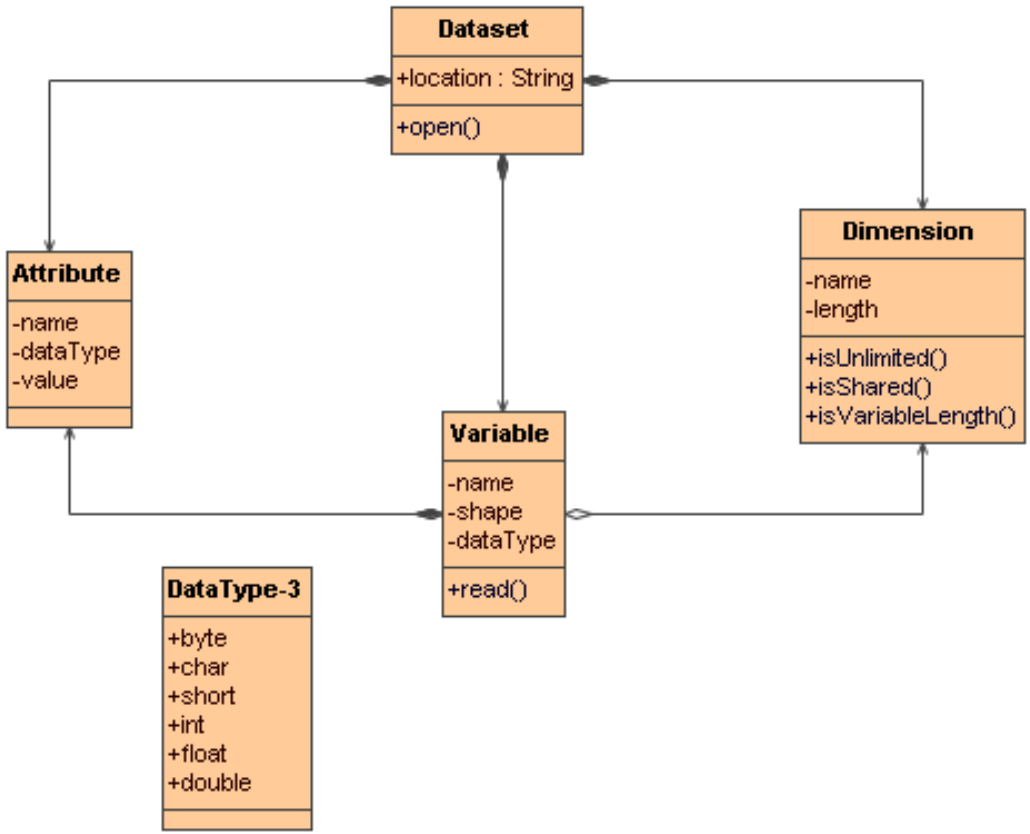
2012-03-05

UNIDATA CDM-1 (Common Data Model)

- Dataset = File or Input-stream
- Data stored in Variables (with shape (=some dimensions) and datatype)
- Additional Informatior

  1-dim array) in

  attributes
  - Global attributes
  - Variable attributes
- Dimensions are sharec



Input configurations: Mapping between file-data-model and CDM

- NetCDF and OpENDAP don't need a configuration, since they are already in CDM
- Felt, grib, metgm and wdb need mapping files written in xml, with doctype (.dtd) or schema (.xsd):

  Check your xml-file against the schema before applying to fimex, e.g.:

```
xmllint --xinclude --postvalid --noout felt2nc_variables.xml

xmllint -noout -schema cdmGribReaderConfig.xsd cdmGribReaderConfig.xml
```

Walk through the config files

- felt2nc_variables.xml
  - See also Problems with short variables in the FAQ
- gribCDMReader.xml
  - Have the grib-documentation at hand
- wdb_config.xml
  - Everything is float
  - wdb_names derived from CF-standard_names

Ncml: changing CDM on the fly

- NcML developped by UNIDATA:

  http://www.unidata.ucar.edu/software/netcdf/ncml/
- Fimex emits NcML at several stages, e.g.

  --input.printNcML, --interpolation.printNcML, --extract.printNcML
- Change, Add and Remove Dimensions, Variables, Attributes and Data
  - Example
  - Missing:
    - Groups (CDM-2 feature)
    - Aggregation (combine several similar files (see Thredds))

Fimex operations: extract

--extract reduces the data of the CDM:

Remove Variables:

--extract.removeVariable arg	remove variables
--extract.selectVariables arg	select only those variables

Cut out a rectangle:

--extract.reduceDimension.name arg	name of a dimension
--extract.reduceDimension.start arg	start position of the dimension
--extract.reduceDimension.end arg	end position of the dimension

High level operations (detect axis and reduce):

--extract.reduceTime.start arg	start-time as iso-string
--extract.reduceTime.end arg	end-time by iso-string
--extract.reduceVerticalAxis.unit arg	unit of vertical axis to reduce
--extract.reduceVerticalAxis.start arg	start value of vertical axis
--extract.reduceVerticalAxis.end arg	end value of the vertical axis

--extract.reduceToBoundingBox.south ...

Extract-example and fimex-config

```
fimex -input.file=input.nc \
--extract.reduceDimension.name=time \
--extract.reduceDimension.start=0 \
--extract.reduceDimension.end=2 --extract.selectVariables=x  \
--extract.selectVariables=y \
--extract.selectVariables=temperature --extract.printNcML
```

Make sure to select all logical variables, i.e. axes x and y in addition to temperature

And a extract.cfg file for the same:

```
[extract]
reduceDimension.name=time
reduceDimension.start=0
reduceDimension.end=2
# add a comment
selectVariables=x
selectVariables=y
selectVariables=temperature
```

and run:

```
fimex -c extract.cfg -input.file=input.nc --extact.printNcML
```

Single config-options can be overwritten by command-line options.

Coordinate Systems / Conventions

- CDM is a data-layout, it does not say what the data means
- Conventions add a meaning to the data, e.g. CF-1.x standard_name
- Fimex uses Conventions to build Coordinate-System, i.e.:
  - Horizontal and vertical axes
  - Auxiliary horizontal axes
  - Projection
  - Time and ReferenceTime
  - Additional axes: realization (ensembles), …
- Diana uses currently a 6-dim data-model (5 fixed (x,y,z,t,rt), 1 user-selectable)
- Print CoordinateSystem: `--input.printCS`

Reprojection / Horizontal Interpolation

- Requirements:
  - Input Coordinate System with horizontal axes
  - Description of new horizontal layout, either:
    - Netcdf-template with latitude/longitude axes (CS)
    - Proj-Projection + axes (in degree or m)
      - `proj -lp`
- Projection Methods
  - nearestneighbor, bilinear, bicubic
    - Require input rectangular grid with named projection
    - Allow interpolation, also from vectors (continuous data)
    - Generally fastest methods
  - coord_nearestneighbor, coord_kdtree
    - Work with any data with known latitude/longitude coordinates
    - Find closest point in input, require extend of input cell (distanceOfInterest)

- forward_max, forward_mean, forward_median, forward_sum
  - Translate input lat/lon to output-grid
  - Very fast, but likely to leave holes (unless reducing resolution)
  - Mass-conservative
  - Does not work with template-based reprojections

- Preprocessing, avoid holes due to coastlines:
  - `fill2d(critx=0.01,cor=1.6,maxLoop=100)`
  - `creepfill2d(repeat=20,weight=2)`

- **Examples**

```
[interpolate]
method = bilinear
projString = +proj=utm +zone=33 +ellps=WGS84
xAxisValues=0,500,...,x;relativeStart=0
yAxisValues=0,500,...,x;relativeStart=0
xAxisUnit = m
yAxisUnit = m
```

Extraction of Latitude/Longitude values

- Below functions work only with nearestNeighbor, bilinear and bicubic methods
- Simple example:

```
[interpolate]

method = bilinear

latitudeValues = 60,60,60.123,45.4

longitudeValues = 10,9,9,17
```

- This will interpolate to the points at (60,10), (60,9), (60.1,9), (45.4,17)

Extraction of Latitude/Longitude points / template

- For large amount (>100) of lat/lon points, templates are more feasible than text-lists. The template is a netcdf-file as given in the documentation
- Invocation:

```
[interpolate]

method = bilinear

template = template4interpolation.nc
```

Vertical Interpolation

- Interpolation of vertical levels to height, depth (in m) or pressure (in hPa).
- Requires: xyzt-coordinate system
- Examples:

```
[verticalInterpolate]
method = linear
type = depth
level1 = 3,5,10,20,50
```

```
[verticalInterpolate]
dataConversion = theta2T
dataConversion = add4Dpressure
method = linear
type = height
level1 =
0,50,100,250,500,750,1000,1250,1500,1750,2250,2750,3250,3750,4250,
4750,5500,6500,7000,7500,8000,8500,9000,9500,10000,10500,11500,12
500,13500,14500,15500,16500,17500,18500,20000
```

- Allows creation of 4D-pressure
- Allows conversion from theta → temp
- Most, but not all vertical-input types supported, no 'sigma' output-types
- Different approx. for v-interpols

Time interpolation

- Requires time-axis in coordinate system
- Example:

```
[timeInterpolate]
timeSpec=0,3,...,24;relativeUnit=hours
since 2000-01-01 00:00:00;unit=seconds
since 1970-01-01 00:00
```

- Description in Fimex-Docs
- Only linear time interpolation
- Will pick exact time if matches!

Quality changes

- Modify contents of one variable-data depending on another variable-data (including itself) (e.g. flags, maximum values)
- Example: cdmQualityConfig.xml

Writer configurations

- Netcdf: cdmWriterConfig.xml
  - Not required
  - Allows logical unit-changes
  - Change compression for variables
- Grib: cdmGribWriterConfig.xml
  - Grib1 and grib2
  - Uses standard_names or variable-names
  - Beta-stage: feedback wanted

Tips

- Debugging:
  - --...printNcML
  - --...printCS
  - -d
- Performance
  - Get faster disks
  - If CPU-load is high, use several CPUs:
    - -n 4