# A CF Conventions Checker *for the future*

Sadie Bartholomew
CMS Weekly meeting
2025-12-05

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

University of Reading

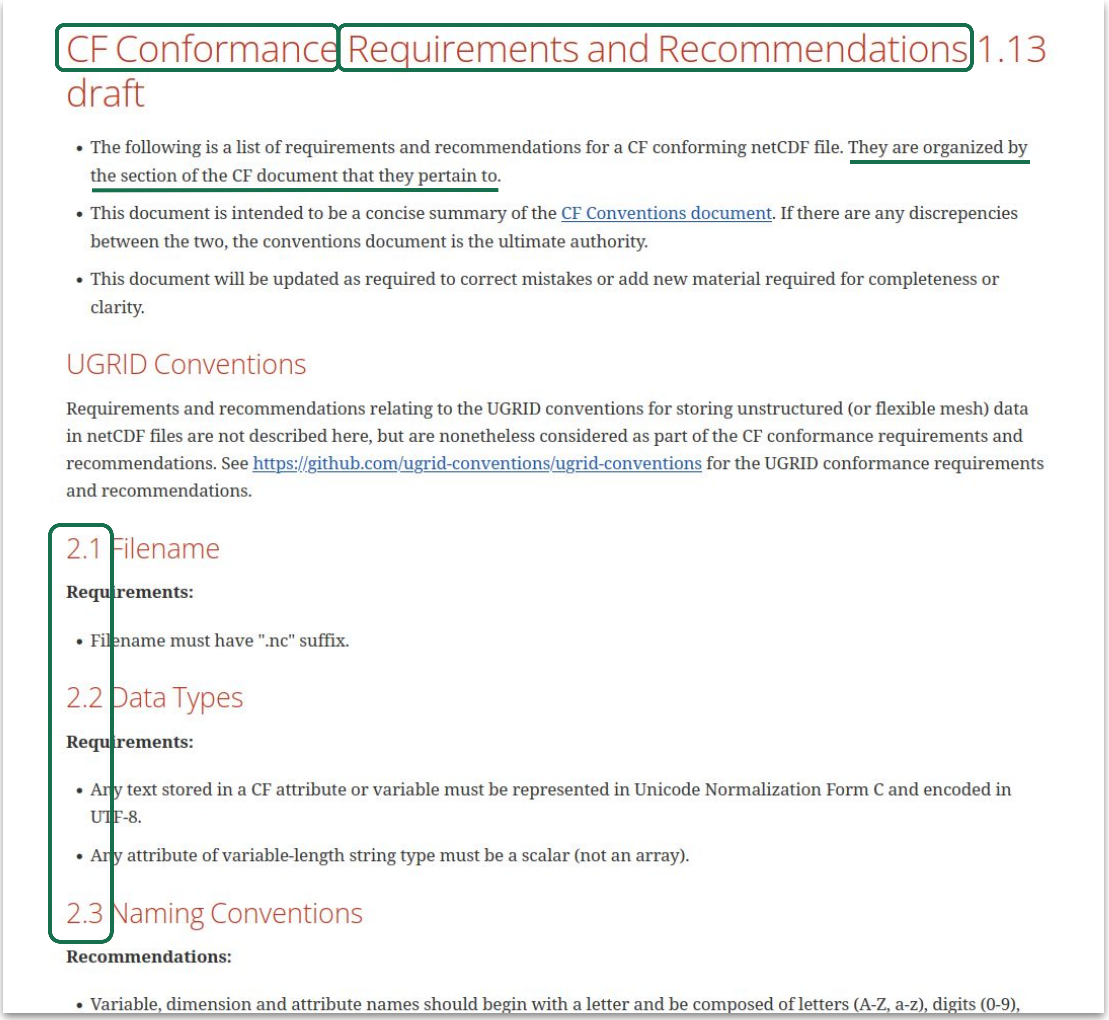
# Four-point outline:

- There is *a lot of* demand, understandably, for **a tool to validate against CF Conventions compliance**: “*Does this netCDF dataset follow the CF Conventions? If not , why not?*”
- A ‘**CF Checker**’ (Ros & CEDA) exists for the CLI & browser (cfchecker.ncas.ac.uk), but due to priorities has not been kept up to date so it can check against max v1.8 (released Feb. 2020)
- Our CF data model library, **cfdm**, already has comprehensive logic for structurally validating against the latest* CF version => the quick(est) win for a modern maintained Checker tool
- The goal is to update cfdm (mostly `netcdfread`) with validation capability, where a core consideration is the output data structure for reporting issues for optimal linkage & parsability

Part 1 of 3

# What is CF Checking?

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL



# CF Conformance Requirements and Recommendations 1.13 draft

- The following is a list of requirements and recommendations for a CF conforming netCDF file. They are organized by the section of the CF document that they pertain to.
- This document is intended to be a concise summary of the CF Conventions document. If there are any discrepencies between the two, the conventions document is the ultimate authority.
- This document will be updated as required to correct mistakes or add new material required for completeness or clarity.

## UGRID Conventions

Requirements and recommendations relating to the UGRID conventions for storing unstructured (or flexible mesh) data in netCDF files are not described here, but are nonetheless considered as part of the CF conformance requirements and recommendations. See https://github.com/ugrid-conventions/ugrid-conventions for the UGRID conformance requirements and recommendations.

## 2.1 Filename

**Requirements:**

- Filename must have ".nc" suffix.

## 2.2 Data Types

**Requirements:**

- Any text stored in a CF attribute or variable must be represented in Unicode Normalization Form C and encoded in UTF-8.
- Any attribute of variable-length string type must be a scalar (not an array).

## 2.3 Naming Conventions

**Recommendations:**

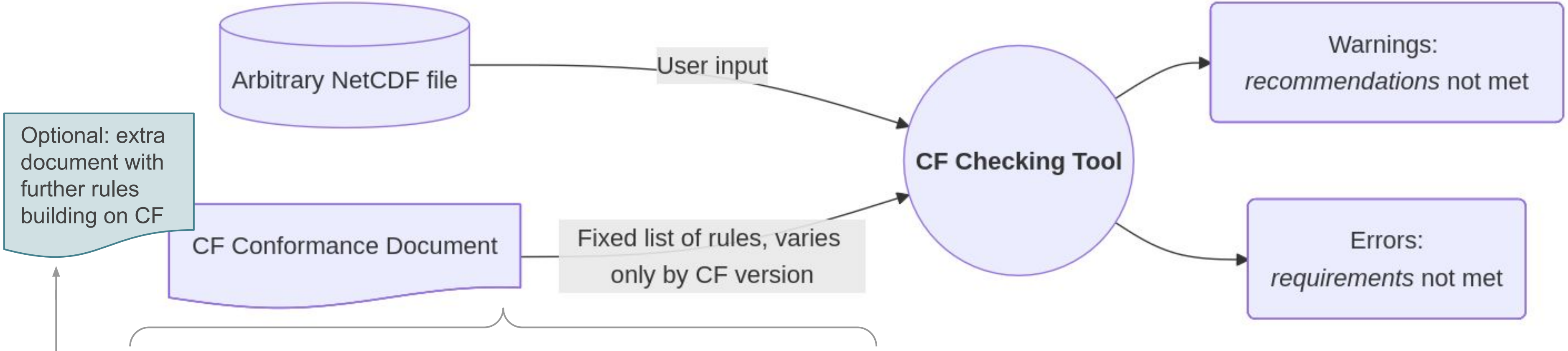
- Variable, dimension and attribute names should begin with a letter and be composed of letters (A-Z, a-z), digits (0-9),

"Each version of the CF conventions document is accompanied by a **conformance document** which lists the requirements and recommendations that could be verified by the compliance checker or any program with a similar aim.

For any *recommendation* which is not followed, the checker gives a warning; for any *requirement* which is not met, it reports an error. "

See document listing at: https://cfconventions.org/conventions.html, shown is a screenshot of the latest (v1.13) document in HTML view (green shapes added by SB to highlight some components)

# CF Checking: automating validation against these criteria


Arbitrary NetCDF file
User input
Optional: extra document with further rules building on CF
CF Conformance Document
Fixed list of rules, varies only by CF version
CF Checking Tool
Warnings: *recommendations* not met
Errors: *requirements* not met

- Nobody has the time and energy to read through the Conformance document and check against each criterion for every dataset (and that would be too prone to error)
- The *ideal* would be a general utility software tool that:
  - will produce these outputs (warnings, errors) from these inputs (any valid netCDF)
  - **could be built upon to add further checking capability, as there is much interest for extension, e.g. CMOR or MIP-specific, etc. requirements**

# Demand! Everyone* wants an up-to-date CF(+) Checker tool. Example...

CMOR

‘Climate Model Output Rewriter’ [which has no logo apparently so I used WordArt]

CMIP

The World Climate Research Programme’s Coupled Model Intercomparison Project

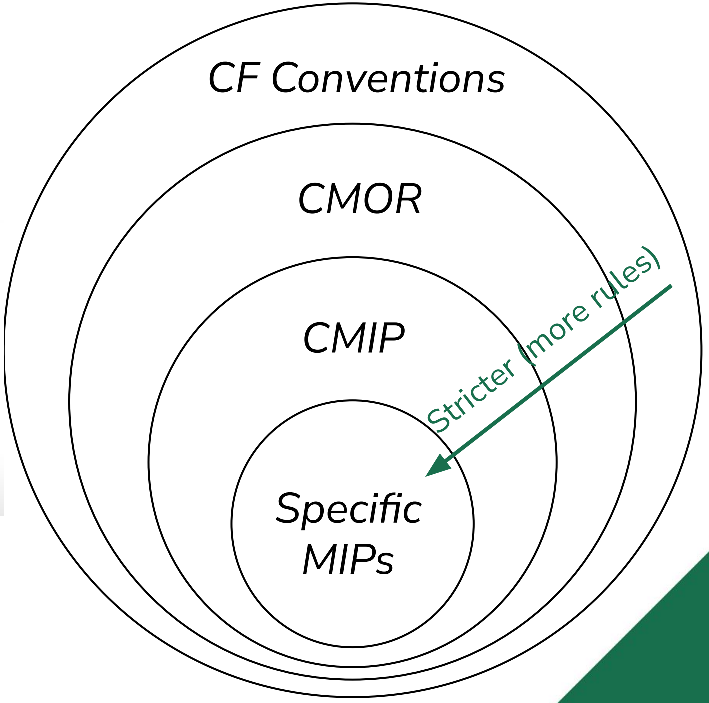
CMIP6, CMIP7, etc.

obs4MIPs

Observations for Model Intercomparisons Project

Specific MIPs, e.g obs4MIPs

(Meta)data standardisation hierarchy:



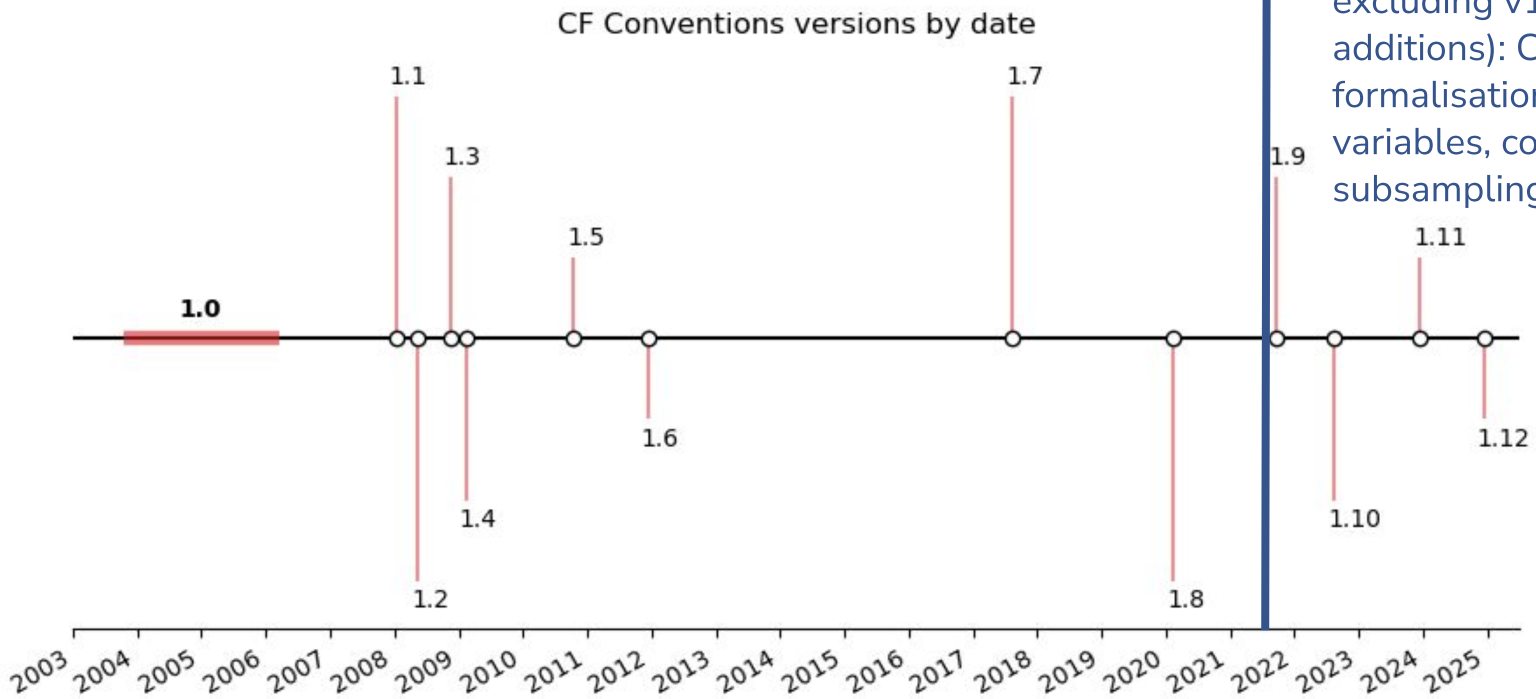
*obviously an exaggeration, but forgive me some dramatisation

Part 2 of 3

# Current status of CF Checking

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

# The main problem: no checking for v1.9+



Plot: SLB’s own, made using Python based on a matplotlib recipe for a software library versioning timeline

## The Wall

Beyond ‘The Wall’ (even excluding v1.13 upcoming additions): CF Data Model formalisation, domain variables, coordinate subsampling, UGRID, etc.

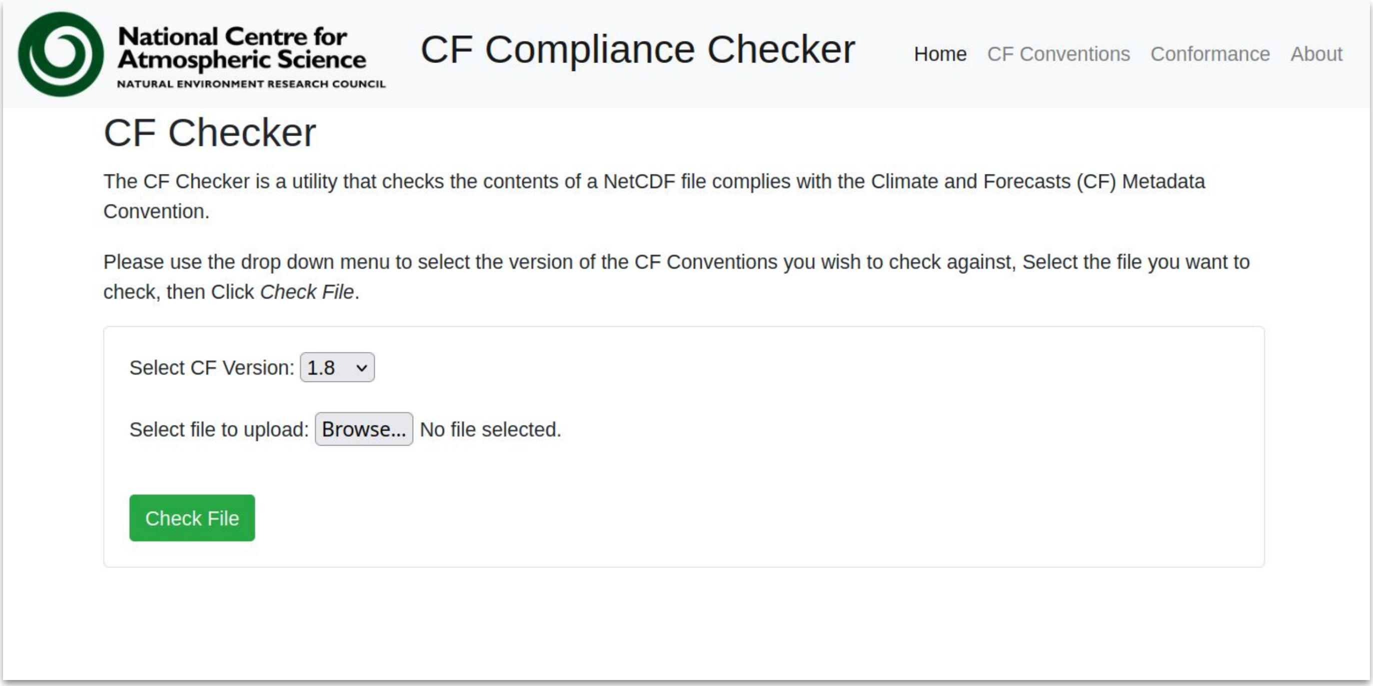
*Note the *controlled vocabularies* e.g. Standard Names have their own independent versioning, and these form separate inputs to the CF Checker, so the latest controlled vocabularies *can* be checked against, even when can’t check 1.9+

# Status of CF Conventions Checking: CF Checker interfaces

CLI basic install (can also use conda, etc.) and command:

```
$ pip install cfchecker
$ cfchecks [-a <area-types.xml>] [-r <regions.xml>] [-s <std_names.xml>] [-v <CFVersion>]
[-x] [-t <cache_time_days>] file1 [file2...]
```

CEDA server hosted browser UI version:


National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL
CF Compliance Checker
Home
CF Conventions
Conformance
About
Browse...
Check File

## CF Checker

The CF Checker is a utility that checks the contents of a NetCDF file complies with the Climate and Forecasts (CF) Metadata Convention.

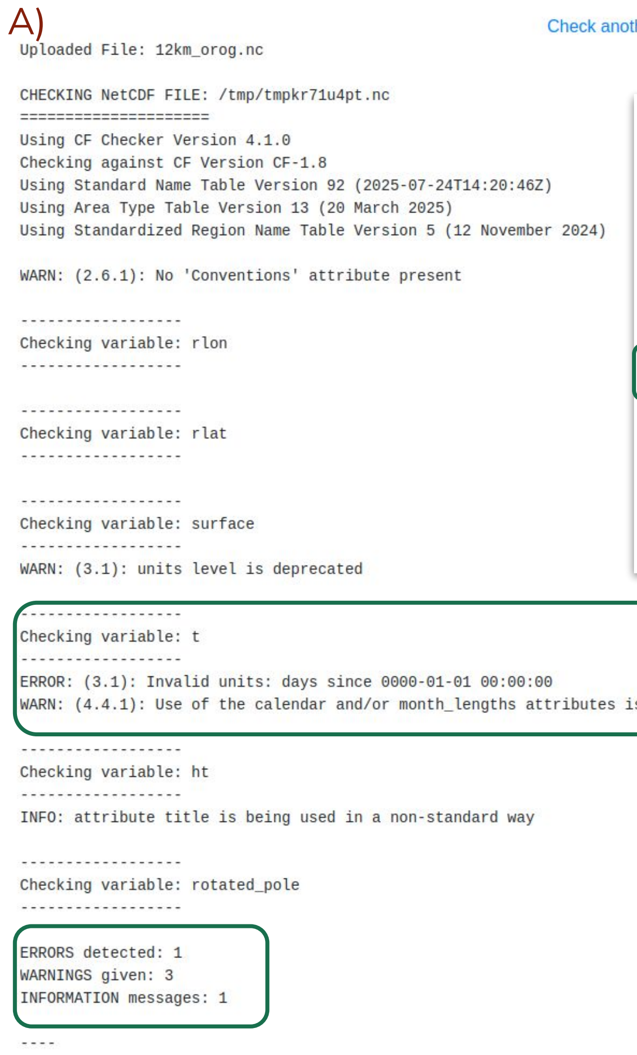
Please use the drop down menu to select the version of the CF Conventions you wish to check against, Select the file you want to check, then Click *Check File*.

Select CF Version: 1.8

Select file to upload: Browse... No file selected.

# CF Checker: example responses

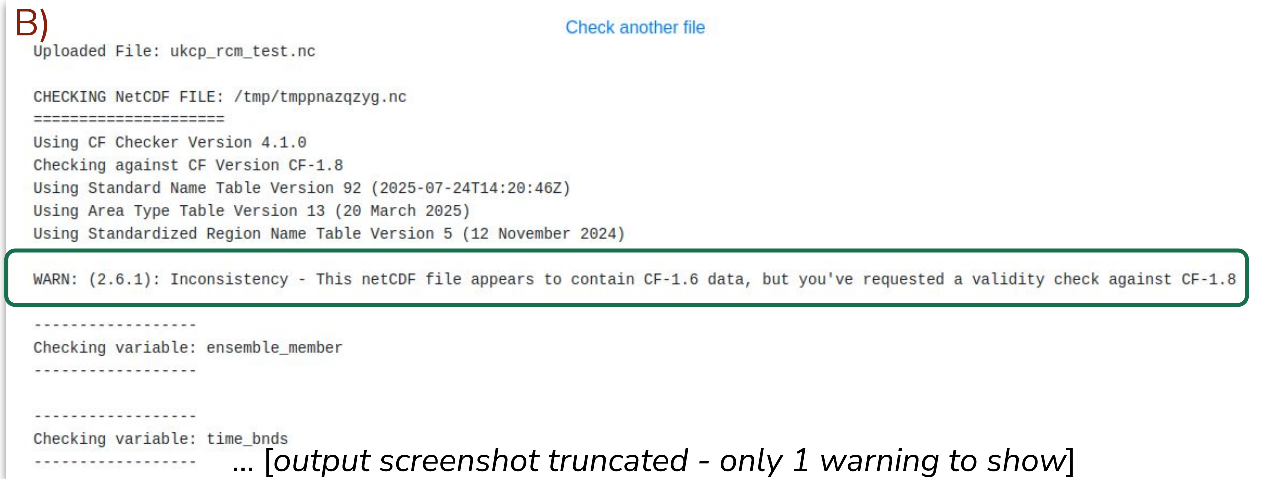
A)



Check another file

```
Uploaded File: 12km_orog.nc

CHECKING NetCDF FILE: /tmp/tmpkr71u4pt.nc
=====================
Using CF Checker Version 4.1.0
Checking against CF Version CF-1.8
Using Standard Name Table Version 92 (2025-07-24T14:20:46Z)
Using Area Type Table Version 13 (20 March 2025)
Using Standardized Region Name Table Version 5 (12 November 2024)

WARN: (2.6.1): No 'Conventions' attribute present

------------------
Checking variable: rlon
------------------

------------------
Checking variable: rlat
------------------

------------------
Checking variable: surface
------------------
WARN: (3.1): units level is deprecated

------------------
Checking variable: t
------------------
ERROR: (3.1): Invalid units: days since 0000-01-01 00:00:00
WARN: (4.4.1): Use of the calendar and/or month_lengths attributes is recommended for time coordinate variables

------------------
Checking variable: ht
------------------
INFO: attribute title is being used in a non-standard way

------------------
Checking variable: rotated_pole
------------------

ERRORS detected: 1
WARNINGS given: 3
INFORMATION messages: 1
----
```

B)

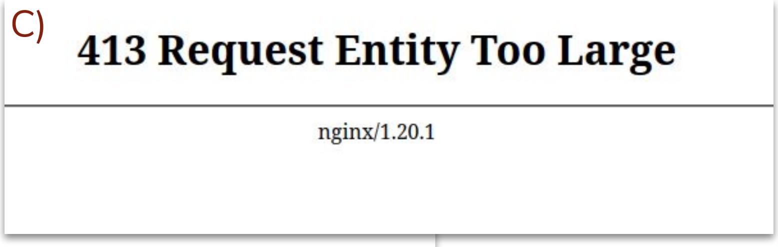


Check another file

```
Uploaded File: ukcp_rcm_test.nc

CHECKING NetCDF FILE: /tmp/tmppnazqzyg.nc
=====================
Using CF Checker Version 4.1.0
Checking against CF Version CF-1.8
Using Standard Name Table Version 92 (2025-07-24T14:20:46Z)
Using Area Type Table Version 13 (20 March 2025)
Using Standardized Region Name Table Version 5 (12 November 2024)

WARN: (2.6.1): Inconsistency - This netCDF file appears to contain CF-1.6 data, but you've requested a validity check against CF-1.8

------------------
Checking variable: ensemble_member
------------------

------------------
Checking variable: time_bnds
------------------
```

... [*output screenshot truncated - only 1 warning to show*]

C)



## 413 Request Entity Too Large

nginx/1.20.1

# Are there other CF Checking tools (already, or in development/planning)?

- Probably!
- But are there any that are *open(-source)*, *complete*, *up-to-date*, *robust* and *maintained*? Almost certainly not! (I doubt there is anything out there that would satisfy even, say, three of those)
- I've heard some whispers on the wind about the CMIP-IPO coordinating some work on CF Checking functionality of some sort.
  - Does anyone know more about this? I can't find much when searching about it...
  - Does anyone know of any other CF checking related work going on or being planned?

Part 3 of 3

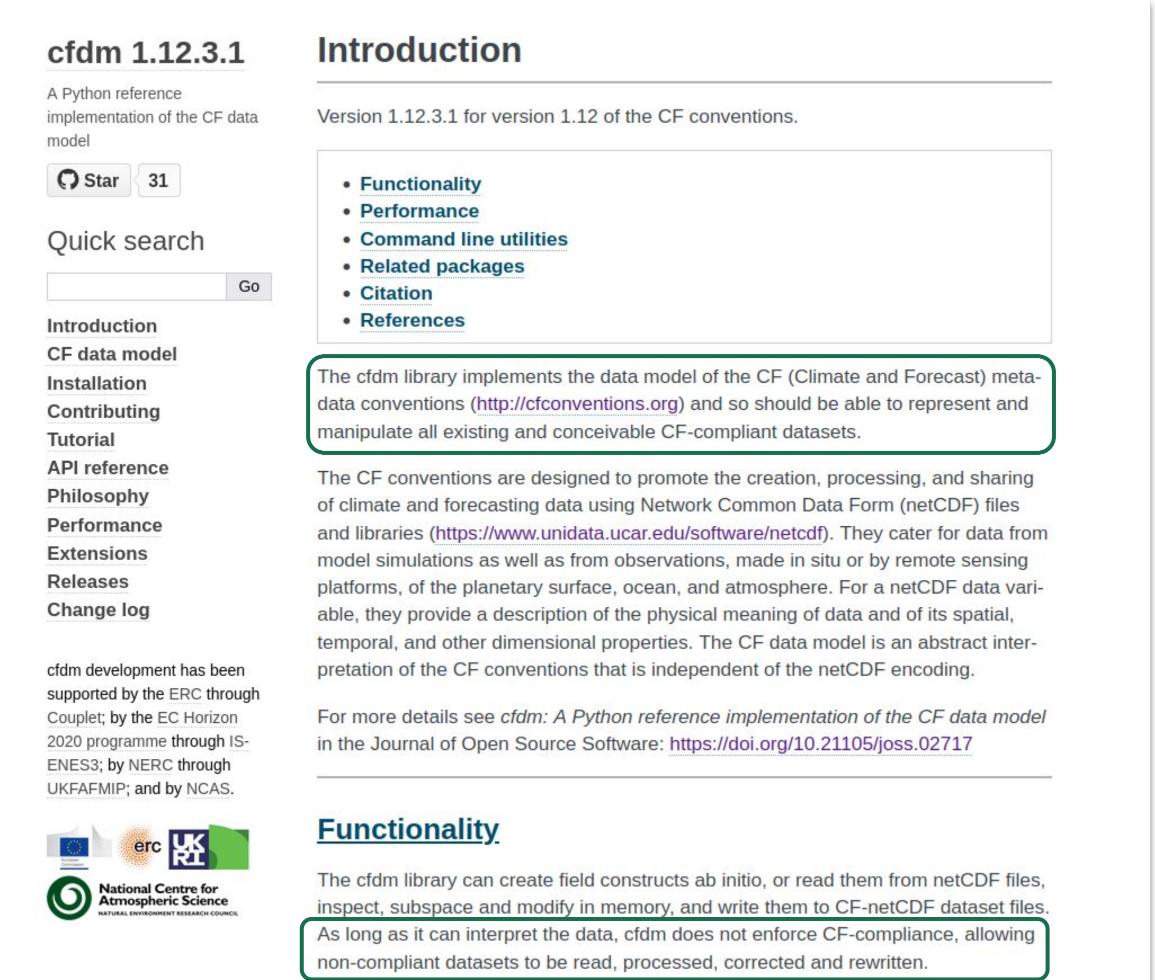
# The future(!?) of CF Checking

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

If only there was already a library which processes netCDF datasets according to the CF Conventions (with requiring any level of compliance, just valid netCDF)...
🤔

Screenshot of the cfdm docs

# Oh wait, there is! And it’s our own!



## cfdm 1.12.3.1

A Python reference implementation of the CF data model

Star 31

Quick search

Go

Introduction
CF data model
Installation
Contributing
Tutorial
API reference
Philosophy
Performance
Extensions
Releases
Change log

cfdm development has been supported by the ERC through Couplet; by the EC Horizon 2020 programme through IS-ENES3; by NERC through UKFAFMIP; and by NCAS.

## Introduction

Version 1.12.3.1 for version 1.12 of the CF conventions.

- Functionality
- Performance
- Command line utilities
- Related packages
- Citation
- References

The cfdm library implements the data model of the CF (Climate and Forecast) metadata conventions (http://cfconventions.org) and so should be able to represent and manipulate all existing and conceivable CF-compliant datasets.

The CF conventions are designed to promote the creation, processing, and sharing of climate and forecasting data using Network Common Data Form (netCDF) files and libraries (https://www.unidata.ucar.edu/software/netcdf). They cater for data from model simulations as well as from observations, made in situ or by remote sensing platforms, of the planetary surface, ocean, and atmosphere. For a netCDF data variable, they provide a description of the physical meaning of data and of its spatial, temporal, and other dimensional properties. The CF data model is an abstract interpretation of the CF conventions that is independent of the netCDF encoding.

For more details see *cfdm: A Python reference implementation of the CF data model* in the Journal of Open Source Software: https://doi.org/10.21105/joss.02717

## Functionality

The cfdm library can create field constructs ab initio, or read them from netCDF files, inspect, subspace and modify in memory, and write them to CF-netCDF dataset files. As long as it can interpret the data, cfdm does not enforce CF-compliance, allowing non-compliant datasets to be read, processed, corrected and rewritten.

- cfdm is perfect: it already has logic to process netCDF relative to CF Compliance (or lack of!)
- The logic does some checking itself with `_check_X` methods
- Includes *some existing* reporting of issues into a dedicated `dataset_compliance()` function which outputs a dictionary reporting issues, but the coverage across the Conformance Document is (as of latest release) *patchy* and *fully untested*
- Also the output dictionary is largely flat and could be better designed to cater for hierarchy

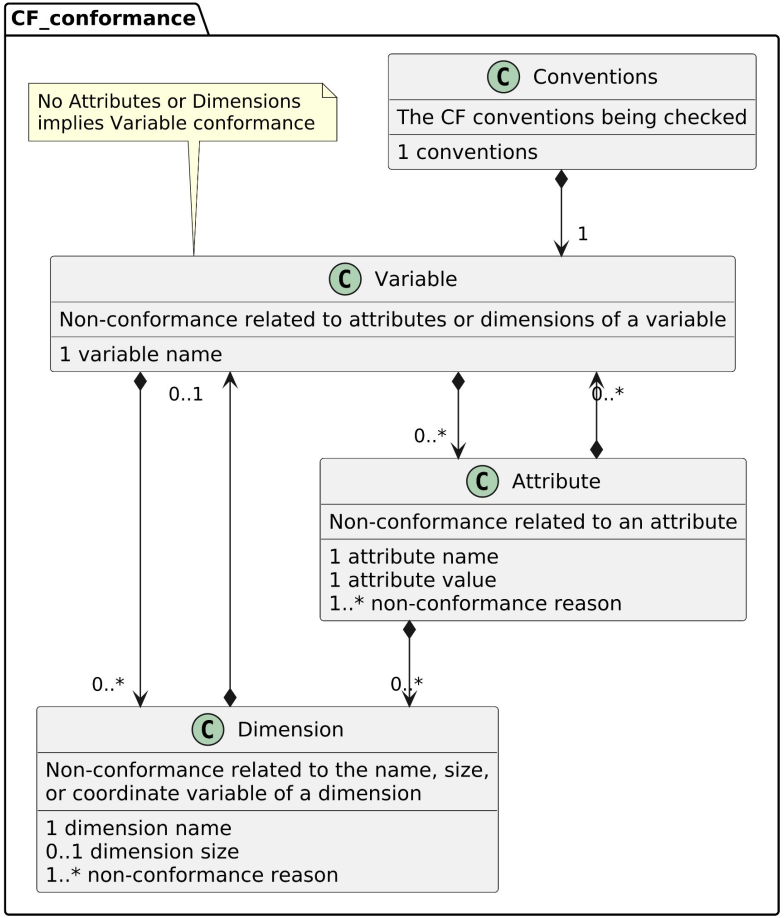
# EXPECT-aligned work on cfdm checking

- Under the EXPECT project, we had time to do some work on built-in checking in cfdm
- Ultimately, CF Checking mostly involves updating logic in the `netcdfread` module
- First considering the lowest hanging fruit, we thought a **PR to validate any standard names registered on a dataset** would be a good start!
- It is trivial to grab the list of valid standard names from the canonical XML and check each attribute value against those, but...
- ...for a more general solution we have been thinking carefully about the data structure to report the issue (see next)

**One of the simplest starting points: Is a(ny) 'standard_name' attribute valid?**

✓ `air_temperature`

✗ `temperature_of_air`


CF_conformance
No Attributes or Dimensions implies Variable conformance
Conventions
The CF conventions being checked
1 conventions
1
Variable
Non-conformance related to attributes or dimensions of a variable
1 variable name
0..1
0..*
0..*
Attribute
Non-conformance related to an attribute
1 attribute name
1 attribute value
1..* non-conformance reason
0..*
0..*
Dimension
Non-conformance related to the name, size, or coordinate variable of a dimension
1 dimension name
0..1 dimension size
1..* non-conformance reason

UML diagram for the conformance cases, courtesy of David Hassell

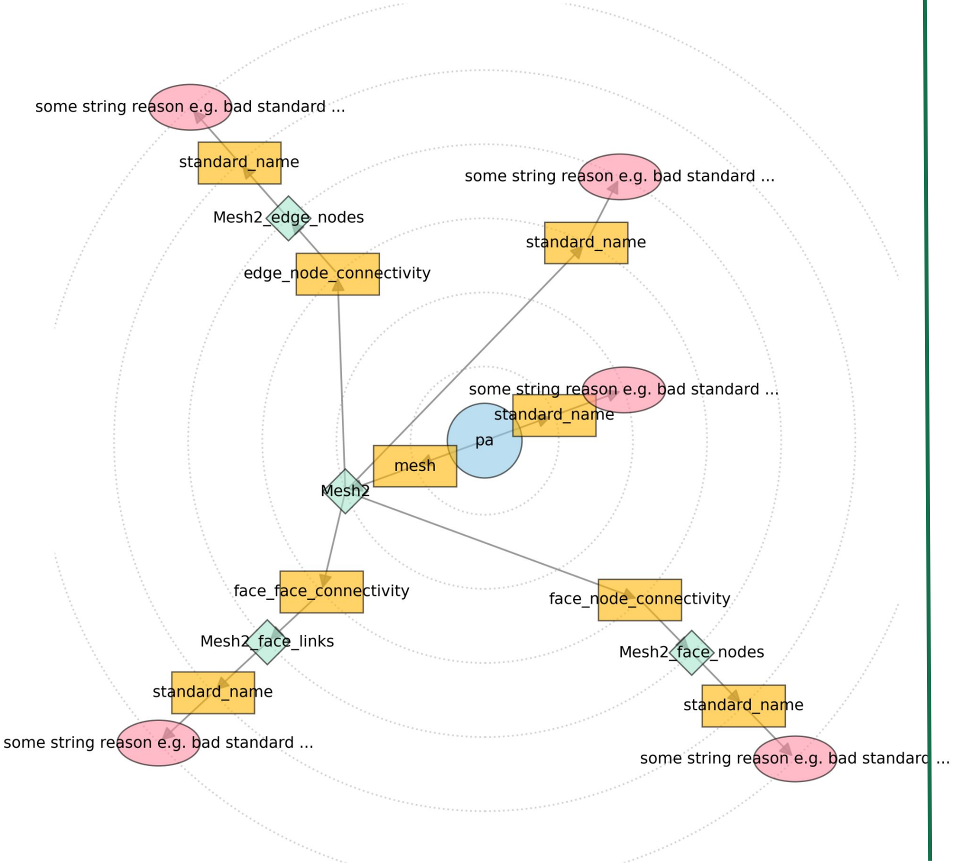
# A data model approach

- When registering issues (non-compliance by requirement or recommendation), want to note these in a **structured way to allow hierarchical storing of problems with components directly, as well as (due to) any children**
- To do this, we considered possible links between components: the 3 netCDF (Classic i.e. core) Data Model objects can map onto each other like so:
  - variable → attribute, dimension
  - attribute → variable, dimension
  - dimension → variable*
- Any ultimate issue is stored as a string 'reason', with a code corresponding to the Conformance document numbering
- For more, if interested, see Issue https://github.com/NCAS-CMS/cfdm/issues/365 where we've summarised our discussions on the topic

# Graphical illustration from ‘vibe coding’ on earlier output structure

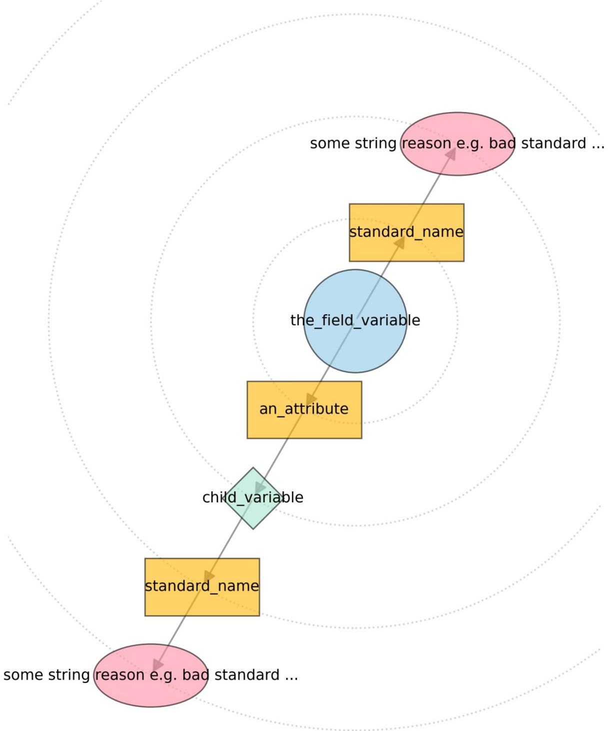
⇲ See left first (general outline), then right (UGRID)

CF Checking example, concentric visualisation: ugrid_example



some string reason e.g. bad standard ...
standard_name
Mesh2_edge_nodes
edge_node_connectivity
some string reason e.g. bad standard ...
standard_name
some string reason e.g. bad standard ...
standard_name
pa
mesh
Mesh2
face_face_connectivity
face_node_connectivity
Mesh2_face_links
Mesh2_face_nodes
standard_name
standard_name
some string reason e.g. bad standard ...
some string reason e.g. bad standard ...

CF Checking example, concentric visualisation: general_idea



some string reason e.g. bad standard ...
standard_name
the_field_variable
an_attribute
child_variable
standard_name
some string reason e.g. bad standard ...

# Example `dataset_compliance()` output structures

```
# 1. CF compliant, at v1.13:
{
    "Conventions": "CF-1.13",
    "tas": {},
}
```

```
# 2. Bad standard name, otherwise compliant
{
    "Conventions": "CF-1.13",
    "tas": {
        "attributes": {
            "standard_name": [
                {"value": "BAD_NAME", "reason": "Invalid standard name"}
            ]
        }
    },
}
```

Basic idea:
variable → [components] → reason for non-compliance

```
# 3. Bad (latitude) standard name when connection is only through a parent variable
#    dimension
{
    "tas": {
        "dimensions": {
            "lat": [
                {
                    "size": 180,
                    "variables": {
                        "lat": {
                            "attributes": {
                                "standard_name": [
                                    {"value": "BAD_NAME", "reason": "invalid name"}
                                ]
                            }
                        }
                    },
                }
            ]
        }
    }
}
```

```
# 4. Bad standard name AND cell measures with bad standard name AND bad key AND
#    spans wrong dimension
{
    "tas": {
        "attributes": {
            "standard_name": [
                {"value": "BAD_NAME", "reason": "Invalid standard name"}
            ],
            "cell_measures": [
                {"value": "BAD_KEY: areacell", "reason": "bad measure key"},
                {
                    "value": "BAD_KEY: areacell",
                    "variables": {
                        "areacell": {
                            "attributes": {
                                "standard_name": [
                                    {
                                        "value": "BAD_NAME",
                                        "reason": "Invalid standard name",
                                    }
                                ]
                            },
                            "dimensions": {
                                "BAD_DIM": [
                                    {
                                        "size": None,
                                        "reason": "dimension mismatch with field",
                                    }
                                ]
                            },
                        }
                    },
                },
            ],
        }
    }
}
```

# Four-point summary (same as the outline! as a reminder):

- There is *a lot of* demand, understandably, for **a tool to validate against CF Conventions compliance**: “*Does this netCDF dataset follow the CF Conventions? If not , why not?*”
- A ‘**CF Checker**’ (Ros & CEDA) exists for the CLI & browser (cfchecker.ncas.ac.uk), but due to priorities has not been kept up to date so it can check against max v1.8 (released Feb. 2020)
- Our CF data model library, **cfdm**, already has comprehensive logic for structurally validating against the latest* CF version => the quick(est) win for a modern maintained Checker tool
- The goal is to update cfdm (mostly `netcdfread`) with validation capability, where a core consideration is the output data structure for reporting issues for optimal linkage & parsability