GRIB decoding with ecCodes

Introduction to ECMWF computing services

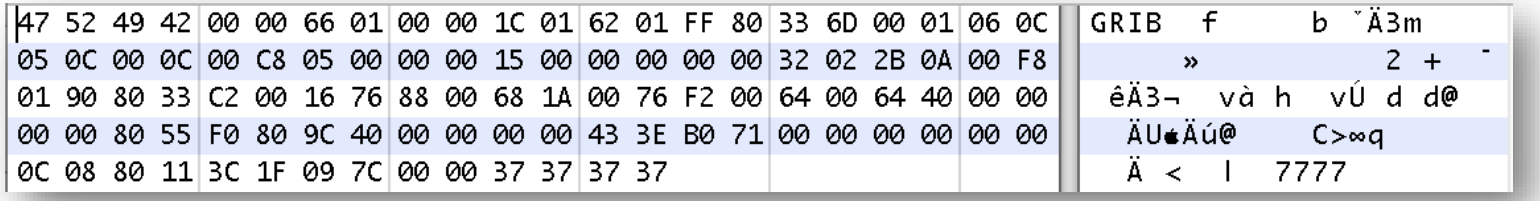
Paul Dando and Yiğit Altıntas

Computing and Software Support Team

ECMWF

What is GRIB ?

- GRIB – “**G**eneral **R**egularly-distributed **I**nformation in **B**inary form”
  - Code defined by the WMO / CBS in 1985 for the exchange of large volumes of gridded data
  - Machine independent
  - Requires software for encoding and decoding

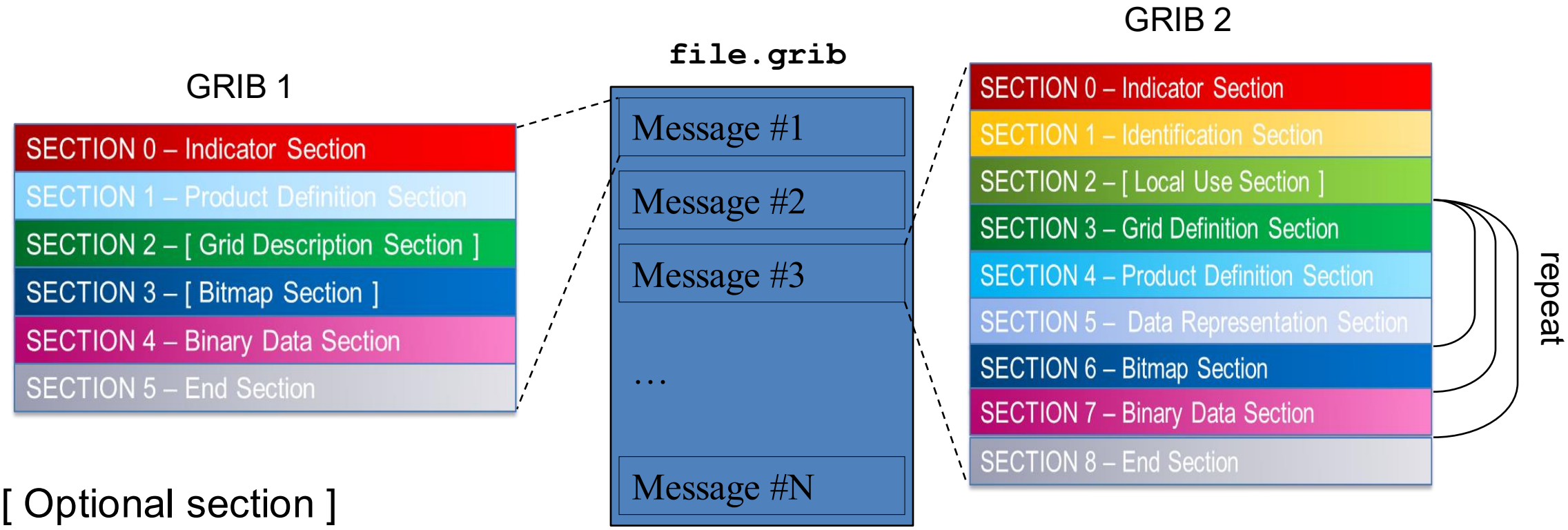


- Currently there are two different coding standards
  - GRIB edition 1
    - Used currently for ECMWF operational pressure level and (most) surface level data
  - GRIB edition 2
    - Used for ECMWF operational model level data since 11 May 2011 and some new surface data
- ECMWF plans to migrate fully to GRIB edition 2 by end of 2026 !

GRIB structure

- A file may contain one or more GRIB messages
- Each message contains several sections
- A file can contain a mix of editions 1 and 2

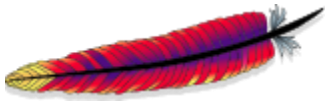
```
47 52 49 42 00 00 66 01 00 00 1C 01 62 01 FF 80 33 6D 00 01 06 0C   GRIB  f     b ˇÄ3m
05 0C 00 0C 00 C8 05 00 00 00 15 00 00 00 00 00 32 02 2B 0A 00 F8      »           2 + ˚
01 90 80 33 C2 00 16 76 88 00 68 1A 00 76 F2 00 64 00 64 40 00 00   êÄ3¬  và h  vÚ d d@
00 00 80 55 F0 80 9C 40 00 00 00 00 43 3E B0 71 00 00 00 00 00 00    ÄU◆Äú@    C>∞q
0C 08 80 11 3C 1F 09 7C 00 00 37 37 37 37                           Ä <  |  7777
```



[Optional section]

What is ecCodes ?

- ecCodes is a package developed by ECMWF for decoding and encoding messages in the following formats:
  - WMO FM-92 GRIB edition 1 and edition 2 (and edition 3)
  - WMO FM-94 BUFR edition 3 and edition 4
  - WMO GTS abbreviated header
- The package includes:
  - command line tools (the GRIB and BUFR Tools) to provide a quick and easy way to manipulate data
  - C, Python and Fortran 90 APIs
- Provides an easy and reliable way of encoding and decoding
  - GRIB 1 and GRIB 2 (and GRIB 3) messages using the **SAME** function calls
- Released under the Apache 2.0 license

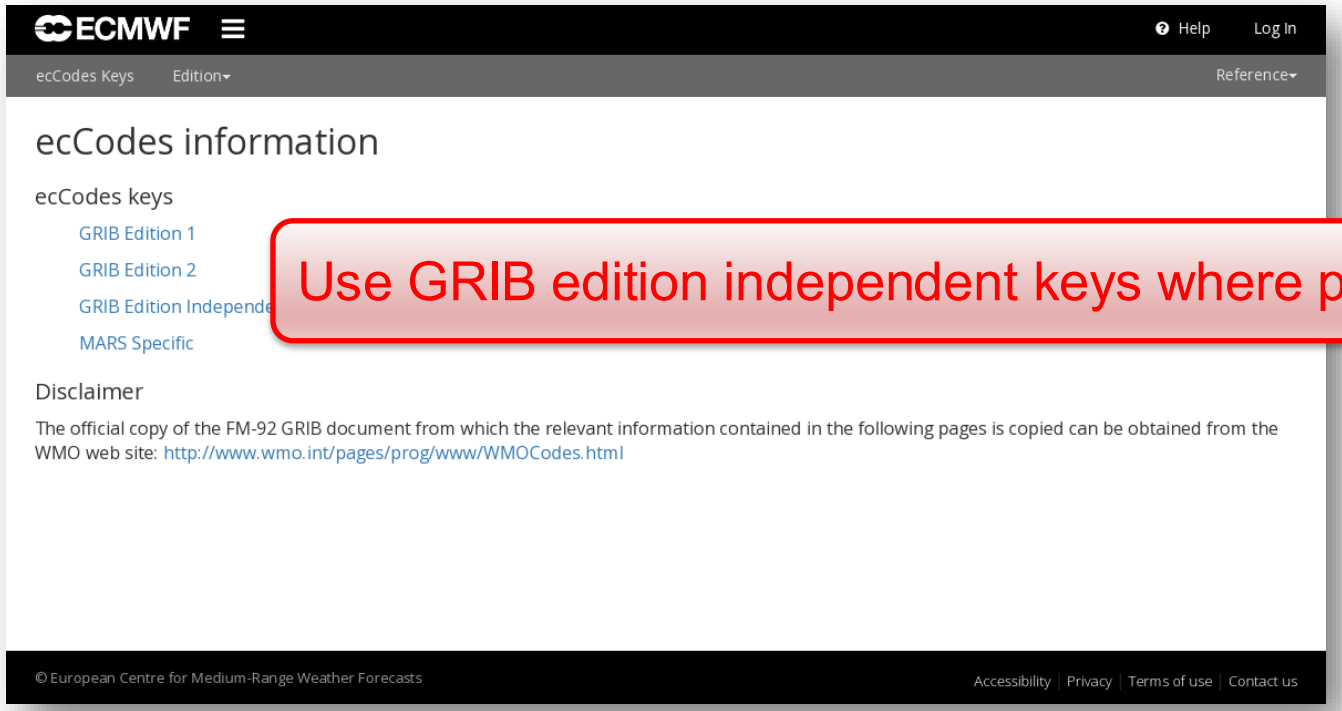


- Available on GitHub

ecCodes key / value based approach

- ecCodes uses a key/value based approach to access information in a GRIB message
  - numberOfPointsAlongAParallel => Number of points along a parallel
  - numberOfPointsAlongAMeridian => Number of points along a meridian
  - …

https://codes.ecmwf.int/grib/



Use GRIB edition independent keys where possible !

ecCodes keys – parameter

- The definition of the parameter is very different in the two GRIB editions

GRIB 1 keys	GRIB 2 keys
centre	discipline
table2Version	parameterCategory
indicatorOfParameter	parameterNumber
levelType	typeOfFirstFixedSurface
level	scaleFactorOfFirstFixedSurface
…	scaledValueOfFirstFixedSurface
	typeOfSecondFixedSurface
	scaleFactorOfSecondFixedSurface
	scaledValueOfSecondFixedSurface
	…

ecCodes keys – parameter namespace

- ecCodes provides some edition-independent keys to identify a parameter

Key name	Example value
paramId	151
shortName	msl
centre	ecmf (or 98)
name	Mean sea level pressure
unit	Pa

- This set of keys is the parameter ***namespace***
- ecCodes provides a number of additional namespaces:
  - ‘parameter’, ‘time’, ‘geography’, ‘vertical’, ‘statistics’, ‘mars’

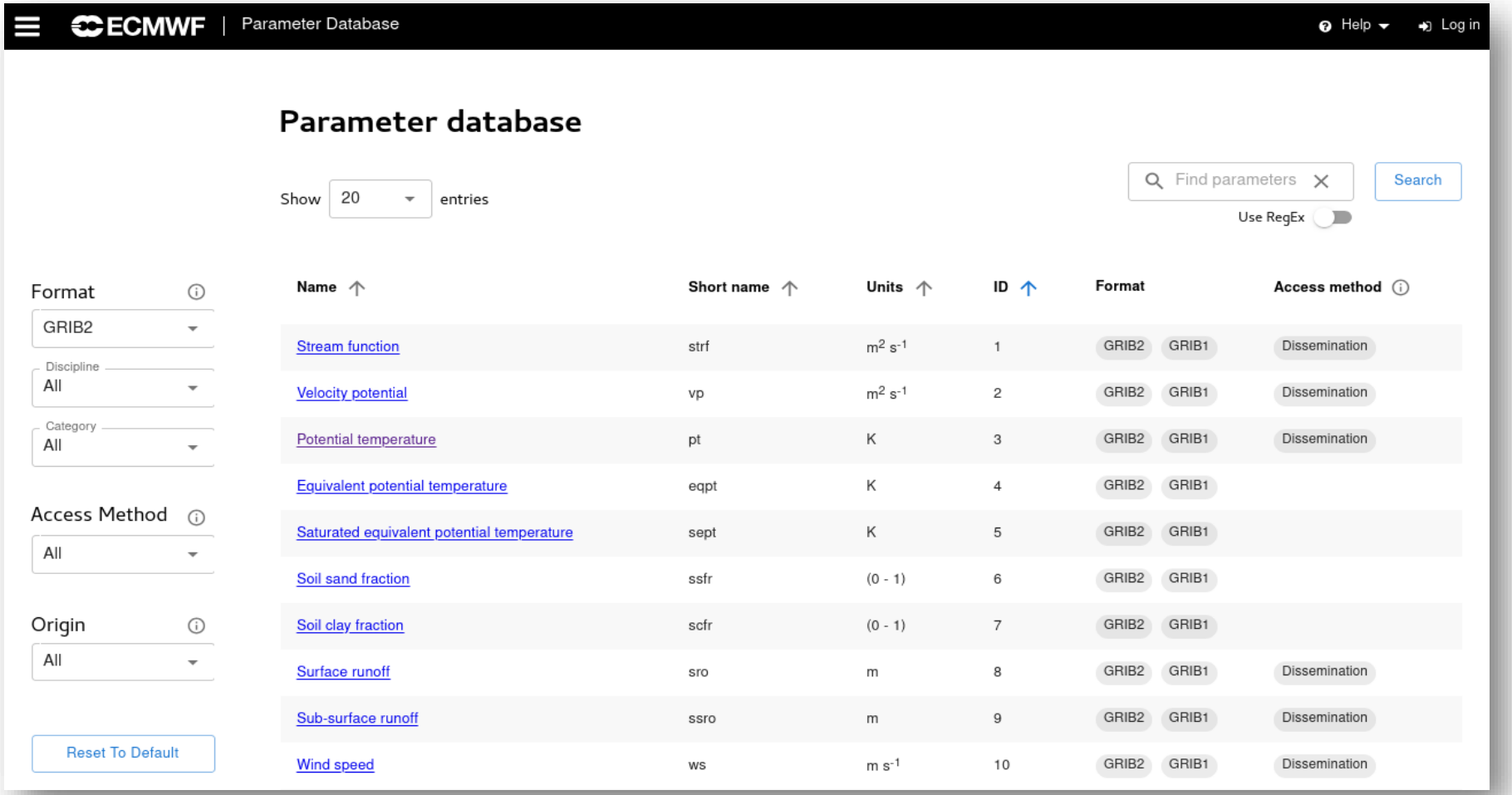
ecCodes – using GRIB edition-independent keys

```
...

edition = codes_get(gid,"edition")

if edition == 1:

  paramNum = codes_get(gid,"indicatorOfparameter")

  table = codes_get(gid,"table2Version")

elif edition == 2:

  paramNum = codes_get(gid,"parameterNumber")

  category = codes_get(gid,"parameterCategory")

...
```

```
...

shortName = codes_get(gid,"shortName")

paramId = codes_get(gid,"paramId")

units = codes_get(gid,"units")

...
```

GRIB parameters – the Parameter Database

https://codes.ecmwf.int/grib/param-db/

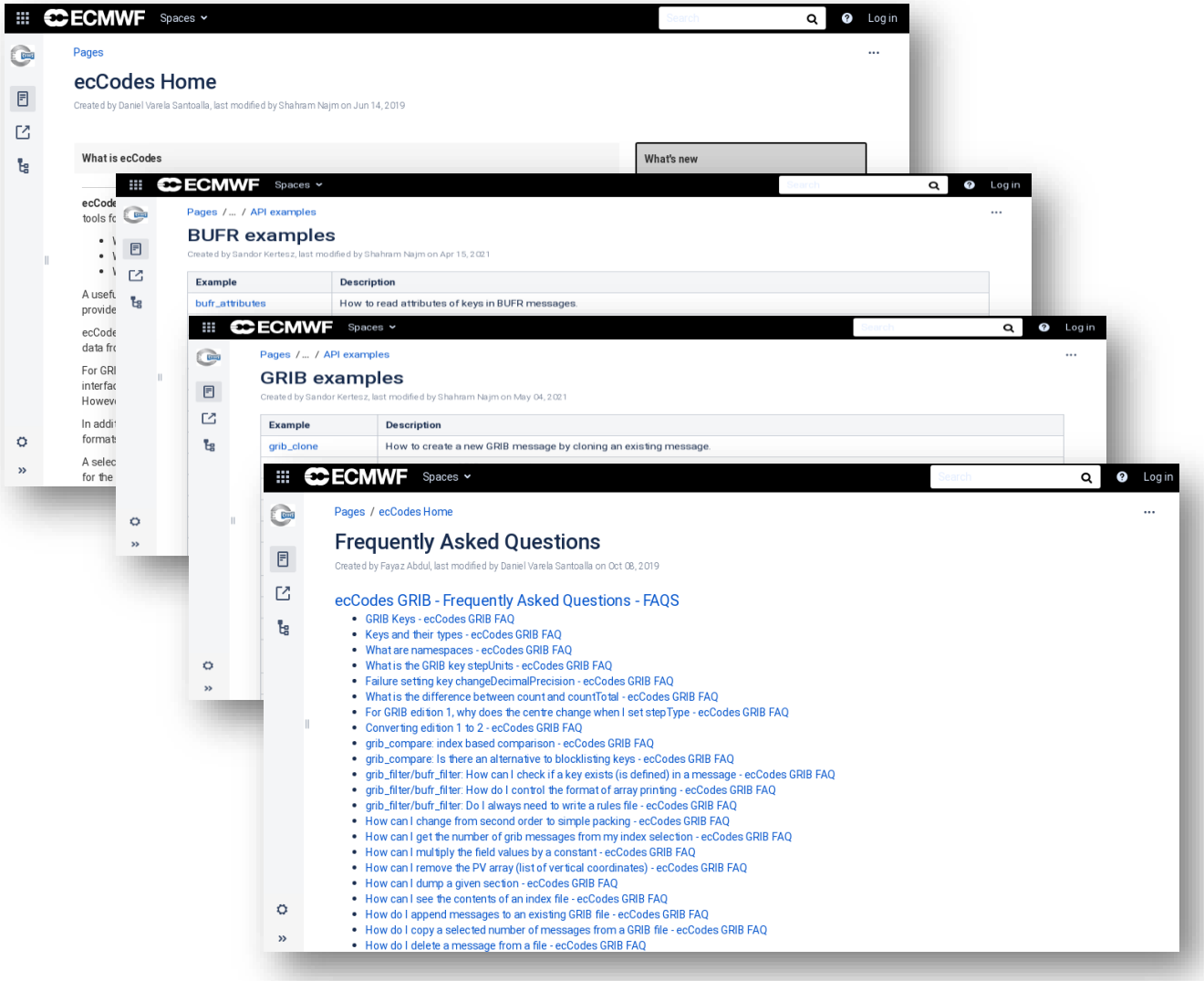


ecCodes documentation

https://confluence.ecmwf.int/display/ECC/ecCodes+Home

- GRIB and BUFR API examples
  - C
  - Python
  - Fortran 90
  - Command-line tools
- FAQs

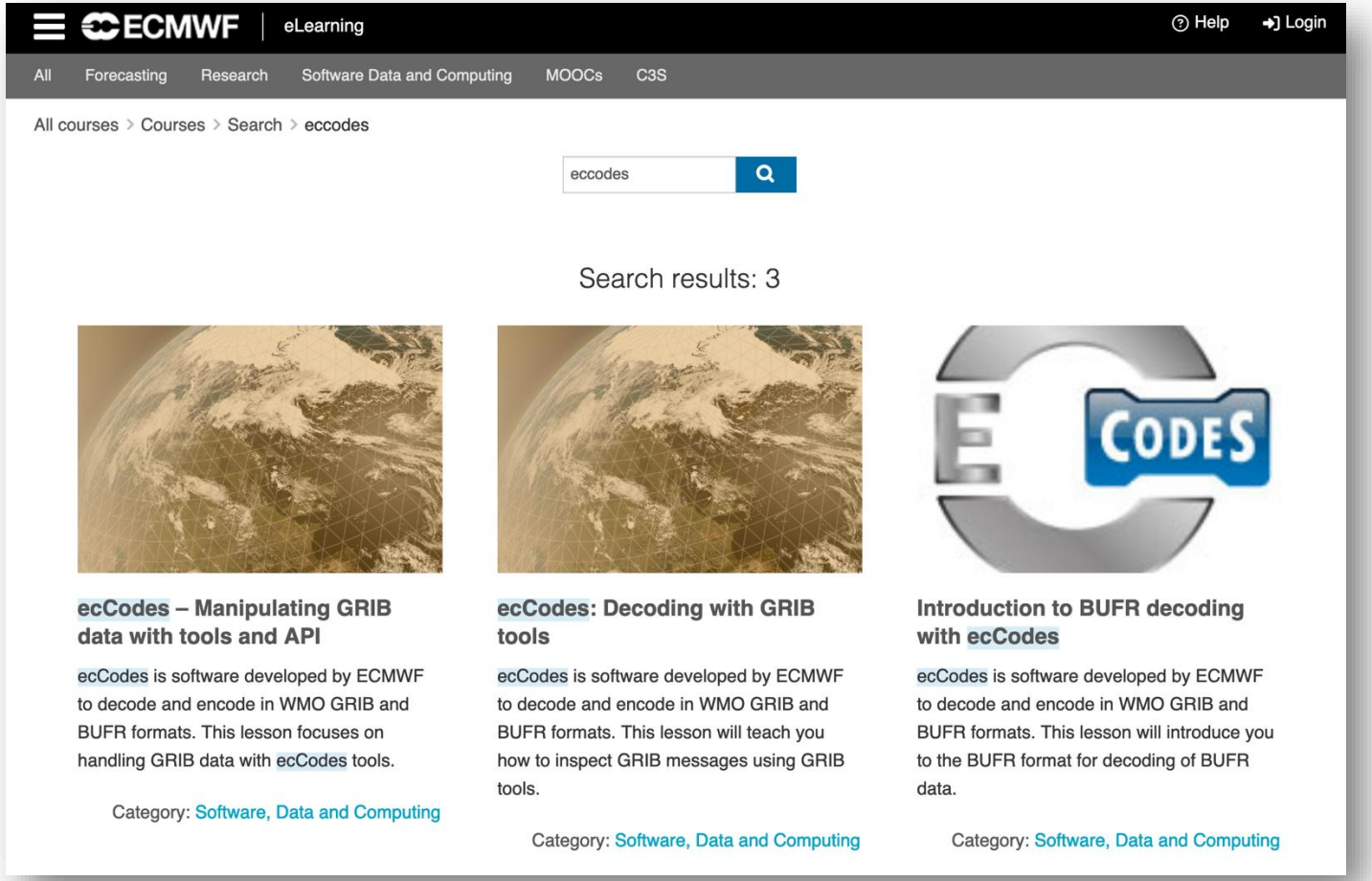
Python3 only



ecCodes eLearning resources

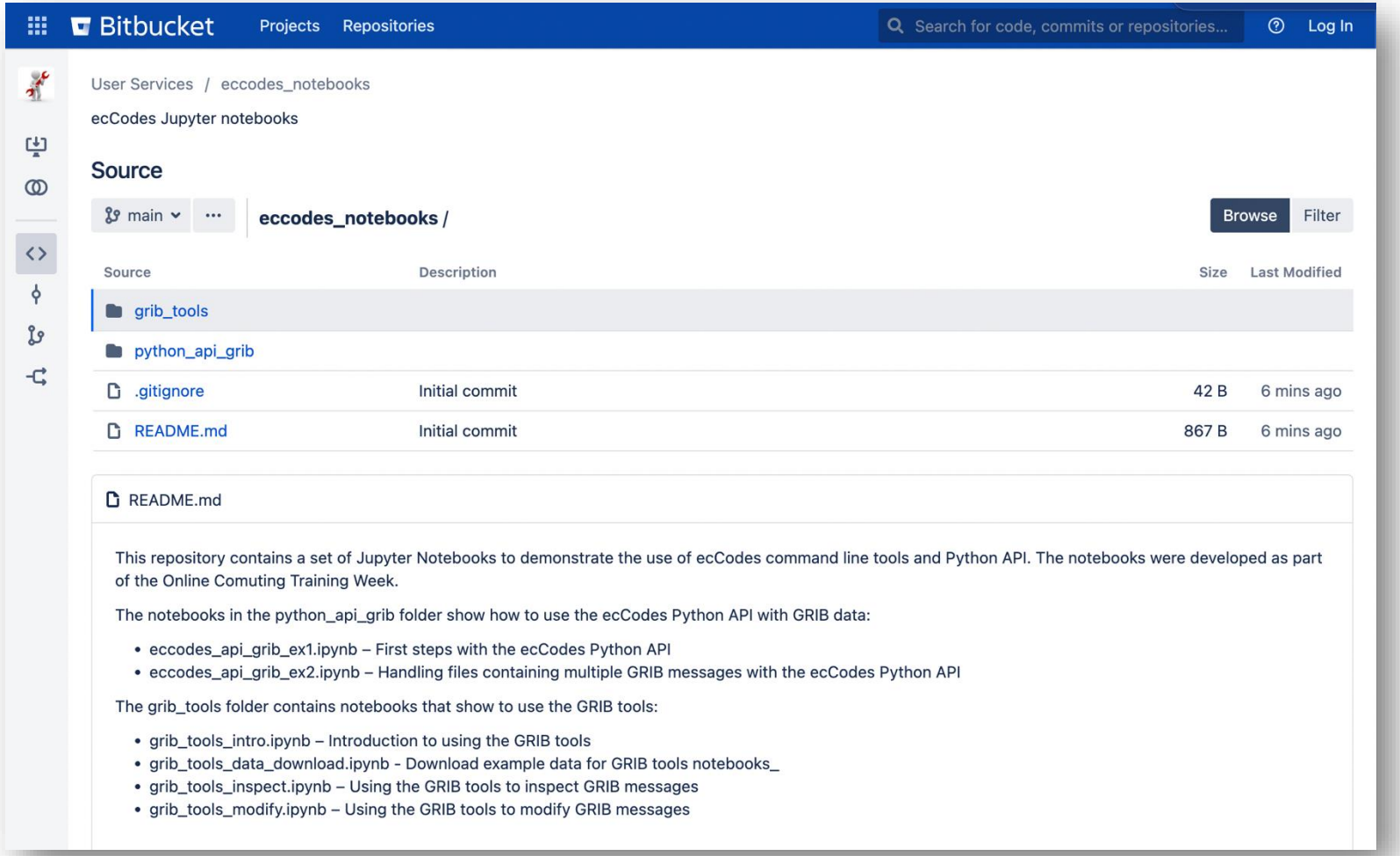
https://www.ecmwf.int/en/learning/education-material/elearning-online-resources

- ecCodes: Decoding with GRIB tools
- ecCodes: Manipulating GRIB data with tools and API



ecCodes Jupyter Notebooks

https://git.ecmwf.int/projects/USS/repos/eccodes_notebooks/browse



Questions ?

GRIB Tools

- All of the tools use a common syntax

```
grib_<tool> [options] grib_file grib_file … [output_grib]
```

- There is tools for getting information about the ecCodes installation
  - codes_info
- There are tools to inspect the content of GRIB messages
  - grib_ls grib_dump grib_get grib_get_data
- There are tools for counting and copying some messages
  - grib_count, grib_copy codes_split_file
- There is a tool for comparing GRIB messages
  - grib_compare
- There are tools for making changes to the content of a GRIB message and converting from GRIB to NetCDF
  - grib_set, grib_filter, grib_to_netcdf

GRIB Tools – getting help

- UNIX ‘man’-style pages are available for each tool by running the tool without any options or input file

```
$ grib_dump
NAME    grib_dump

DESCRIPTION
        Dump the content of a grib file in different formats.

USAGE
        grib_dump [options] grib_file grib_file ...

OPTIONS
        -O      Octet mode. WMO documentation style dump.
        -D      Debug mode.
        -d      Print all data values.
...
```

GRIB tools for inspecting GRIB messages

- grib_ls
  - Provides a customisable summary of the content of GRIB files
- grib_dump
  - Provides a detailed view of the contents of the messages in GRIB files
- grib_get
  - Print the values of specified keys in a GRIB message
  - Similar to grib_ls but more for use in shell scripts
- grib_get_data
  - Print the latitude, longitude and data values of GRIB messages in one or more GRIB file

Demo Preliminaries

Access to files on ATOS HPC:

To copy exercise files to your $SCRATCH, do the following on ATOS HPC:

```
$ cd $SCRATCH
$ tar -xvf /home/trx/grib_decoding.tar
```

or access online: https://sites.ecmwf.int/repository/opencharts-sample-data/2025_computing/

```
$ ls grib_decoding/demo
20251012000000-114h-enfo-cf.grib2
A1S10071800100815001
nvo_x1_aifs-single_ai_oper_fc_20251011T060000Z_20251011T120000Z_6h
output.grib1
python_eccodes.py
```

Access env.yaml and eccodes_python.py files: https://github.com/ecmwf-training/2025-computing-training-week/tree/main/eccodes_grib

grib_ls – list the content of GRIB files

- Use grib_ls to get a summary of the content of GRIB files

Main options

`-p key1,key2,…`	Keys to print
`-P key1,key2,…`	Additional keys to print
`-w key1=val1,key2!=val2…`	Where option
`-B "key asc, key desc"`	Order by: “step asc, centre desc”
`-n namespace`	Print keys for `namespace`
`-m`	Print MARS keys
`-j`	JSON output
`-l lat,lon[,MODE,FILE]`	Value(s) nearest to lat-lon point
`-F format`	Format for floating point values
`-s key1=val1,key2=val2`	Key values to set for output

grib_ls – examples

```
$ grib_ls 20251012000000-114h-enfo-cf.grib2
20251012000000-114h-enfo-cf.grib2
edition    centre    date    datatype    gridType    stepRange    typeOfLevel       level    shortName    model    packingType
2          ecmf    20251012    cf       regular_ll     114       isobaricInhPa      400     t          aifs-ens    grid_ccsds
2          ecmf    20251012    cf       regular_ll     114       highCloudLayer     450     hcc        aifs-ens    grid_ccsds
2          ecmf    20251012    cf       regular_ll     114       surface              0     skt        aifs-ens    grid_ccsds
2          ecmf    20251012    cf       regular_ll     114       soilLayer            1     sot        aifs-ens    grid_ccsds
2          ecmf    20251012    cf       regular_ll     114       meanSea              0     msl        aifs-ens    grid_ccsds
2          ecmf    20251012    cf       regular_ll     114       entireAtmosphere     0     tcw        aifs-ens    grid_ccsds
2          ecmf    20251012    cf       regular_ll     114       surface              0     sp         aifs-ens    grid_ccsds
2          ecmf    20251012    cf       regular_ll     114       heightAboveGround  100     100u       aifs-ens    grid_ccsds
8 of 8 messages in 20251012000000-114h-enfo-cf.grib2
8 of 8 total messages in 1 files
```

```
$ grib_ls -p centre,dataDate,shortName,paramId,typeOfLevel,level 20251012000000-114h-enfo-cf.grib2
20251012000000-114h-enfo-cf.grib2
centre       dataDate     shortName    paramId      typeOfLevel       level
ecmf         20251012     t            130          isobaricInhPa       400
ecmf         20251012     hcc          3075         highCloudLayer      450
ecmf         20251012     skt          235          surface               0
ecmf         20251012     sot          260360       soilLayer             1
ecmf         20251012     msl          151          meanSea               0
ecmf         20251012     tcw          136          entireAtmosphere      0
ecmf         20251012     sp           134          surface               0
ecmf         20251012     100u         228246       heightAboveGround   100
8 of 8 messages in 20251012000000-114h-enfo-cf.grib2
8 of 8 total messages in 1 files
```

Use the -p option to specify a list of keys to be printed

grib_ls – examples

- When a key is not present in the GRIB file, it returns “not found” for this key

```
$ grib_ls -p my_key output.grib1
output.grib1
my_key
not_found
not_found
2 of 2 messages in output.grib1
2 of 2 total messages in 1 files
$ echo $?
0
$ grib_get -p my_key output.grib1
ECCODES ERROR   :  my_key (Key/value not found)
$ echo $?
246
```

grib_ls: exit code returned = 0

grib_get: exit code returned != 0

- If you need the command to fail, use the grib_get tool
  - grib_ls is more for interactive use
  - use grib_get within scripts

Using the ‘where’ option

- The where option **-w** can be used with all the GRIB Tools
- Constraints are of the form key=value or key!=value or key=value1/value2/value2

```
-w key1=value1,key2:i!=value2,key3:s=value3
```

- Messages are processed only if they match ALL the key / value constraints

```
$ grib_ls -w level=100 file.grib2                    “IS”
...
$ grib_ls -w level!=100 file.grib2                   “NOT”
...
$ grib_ls -w level=100,stepRange=3 file.grib2        “AND”
...
$ grib_ls -w level=100/200/300/500 file.grib2        “OR”
...
```

Specifying the type of the key

- All ecCodes keys have a default type
  - e.g. string, integer, floating point
- The type of the key can be specified as follows:
  - **key** → native type
  - **key:i** → integer
  - **key:s** → string
  - **key:d** → double

```
$ grib_ls -p centre:i,dataDate,shortName,paramId,typeOfLevel,level 20251012000000-114h-enfo-cf.grib2
20251012000000-114h-enfo-cf.grib2
centre       dataDate     shortName    paramId      typeOfLevel    level
98           20251012     t            130          isobaricInhPa      400
98           20251012     hcc          3075         highCloudLayer     450
98           20251012     skt          235          surface              0
98           20251012     sot          260360       soilLayer            1
98           20251012     msl          151          meanSea              0
98           20251012     tcw          136          entireAtmosphere     0
98           20251012     sp           134          surface              0
98           20251012     100u         228246       heightAboveGround  100
8 of 8 messages in 20251012000000-114h-enfo-cf.grib2

8 of 8 total messages in 1 files
```

grib_ls – stepRange and stepUnits

- The step is always printed as an integer value
- By default the units of the step are printed in hours
- To obtain the step in other units set the stepUnits key appropriately with the **-s** option

```
$ grib_ls -p stepRange output.grib1
output.grib1
stepRange
0
360
...
$ grib_ls -s stepUnits=m -p stepRange output.grib1
output.grib1
stepRange
0
21600
...
```

stepUnits can be s, m, h, 3h, 6h, 12h, D, M, Y, 10Y, 30Y, C

Finding nearest grid points with grib_ls

- The value of a GRIB field close to a specified Latitude/Longitude point can be found with grib_ls

optional

```
grib_ls -l Latitude,Longitude,MODE,file grib_file
```

MODE Can take the values

4 Print values at the 4 nearest grid points (default)
1 Print value at the closest grid point

file Specifies a GRIB file to use as a mask
The closest *land* point (with mask ≥ 0.5) is printed

- GRIB files specified must contain grid point data

grib_dump – dump contents of GRIB files

- Use grib_dump to get a detailed view of the content of a file containing one or more GRIB messages

Main options:

`-O`	Octet mode (WMO Documentation style)
`-D`	Debug mode
`-a`	Print key alias information
`-t`	Print key type information
`-H`	Print octet content in hexadecimal
`-w key{=/!=}value,…`	Where option
`-j`	JSON mode
`...`	

grib_dump – examples

```
$ grib_dump 20251012000000-114h-enfo-cf.grib2
***** FILE: 20251012000000-114h-enfo-cf.grib2
#=============   MESSAGE 1 ( length=601632 )             =============
GRIB {
  # Meteorological products (grib2/tables/35/0.0.table)
  discipline = 0;
  editionNumber = 2;
  # European Centre for Medium-Range Weather Forecasts (common/c-11.table)
  centre = 98;
  subCentre = 0;
  # Start of forecast (grib2/tables/35/1.2.table)
  significanceOfReferenceTime = 1;
  dataDate = 20251012;
  dataTime = 0;
  # Operational products (grib2/tables/35/1.3.table)
  productionStatusOfProcessedData = 0;
  # ML based forecast (grib2/tables/35/1.4.table)
  typeOfProcessedData = 10;
  # MARS labelling (grib2/grib2LocalSectionNumber.98.table)
  grib2LocalSectionNumber = 1;
  # Operational AIFS (mars/class.table)
....
  parameterNumber = 0;
  #-READ ONLY- parameterUnits = K;
  #-READ ONLY- parameterName = Temperature;
  # Ensemble forecast (grib2/tables/35/4.3.table)
  typeOfGeneratingProcess = 4;
  generatingProcessIdentifier = 1;
  # Hour (grib2/tables/35/4.4.table)
....
```

keys are case sensitive:

dataDate, dataTime

Some keys are read only

grib_dump examples: WMO octet mode

```
$ grib_dump -O 20251012000000-114h-enfo-cf.grib2
***** FILE: 20251012000000-114h-enfo-cf.grib2
#=============   MESSAGE 1 ( length=601632 )             =============
1-4       identifier = GRIB
5-6       reserved = MISSING
7         discipline = 0 [Meteorological products (grib2/tables/35/0.0.table) ]
8         editionNumber = 2
9-16      totalLength = 601632
======================   SECTION_1 ( length=21, padding=0 )    ====================
1-4       section1Length = 21
5         numberOfSection = 1
6-7       centre = 98 [European Centre for Medium-Range Weather Forecasts (common/c-11.table) ]
8-9       subCentre = 0
10        tablesVersion = 35 [Version implemented on 19 May 2025 (grib2/tables/1.0.table) ]
11        localTablesVersion = 0 [Local tables not used (grib2/tables/local/ecmf/1.1.table) ]
12        significanceOfReferenceTime = 1 [Start of forecast (grib2/tables/35/1.2.table) ]
13-14     year = 2025
15        month = 10
16        day = 12
17        hour = 0
18        minute = 0
19        second = 0
20        productionStatusOfProcessedData = 0 [Operational products (grib2/tables/35/1.3.table) ]
21        typeOfProcessedData = 10 [ML based forecast (grib2/tables/35/1.4.table) ]
======================   SECTION_2 ( length=17, padding=0 )    ====================
1-4       section2Length = 17
5         numberOfSection = 2
6-7       grib2LocalSectionNumber = 1 [MARS labelling (grib2/grib2LocalSectionNumber.98.table) ]
....
```

grib_dump examples: Octet mode with types, aliases and Hex

```
$ grib_dump -OtaH 20251012000000-114h-enfo-cf.grib2
***** FILE: 20251012000000-114h-enfo-cf.grib2
#=============   MESSAGE 1 ( length=601632 )              =============
1-4       tascii (str) identifier = GRIB ( 0x47 0x52 0x49 0x42 )
5-6       unsigned (int) reserved = MISSING ( 0xFF 0xFF )
7         codetable (int) discipline = 0 ( 0x00 ) [Meteorological products (grib2/tables/35/0.0.table) ]
8         unsigned (int) editionNumber = 2 ( 0x02 ) [ls.edition]
9-16      section_length (int) totalLength = 601632 ( 0x00 0x00 0x00 0x00 0x00 0x09 0x2E 0x20 )
======================   SECTION_1 ( length=21, padding=0 )    ======================
1-4       section_length (int) section1Length = 21 ( 0x00 0x00 0x00 0x15 )
5         unsigned (int) numberOfSection = 1 ( 0x01 )
6-7       codetable (int) centre = 98 ( 0x00 0x62 ) [European Centre for Medium-Range Weather Forecasts (common/c-
    11.table) ] [identificationOfOriginatingGeneratingCentre, ls.centre, originatingCentre, centreForLocal]
8-9       unsigned (int) subCentre = 0 ( 0x00 0x00 )
10        codetable (int) tablesVersion = 35 ( 0x23 ) [Version implemented on 19 May 2025 (grib2/tables/1.0.table) ]
    [gribMasterTablesVersionNumber]
11        codetable (int) localTablesVersion = 0 ( 0x00 ) [Local tables not used (grib2/tables/local/ecmf/1.1.table) ]
    [versionNumberOfGribLocalTables]
12        codetable (int) significanceOfReferenceTime = 1 ( 0x01 ) [Start of forecast (grib2/tables/35/1.2.table) ]
13-14     unsigned (int) year = 2025 ( 0x07 0xE9 )
15        unsigned (int) month = 10 ( 0x0A )
16        unsigned (int) day = 12 ( 0x0C )
...
```

grib_get_data – print data values

- Use grib_get_data to print a list of latitude, longitude (for grid point data) and data values from one or more GRIB files
- C-style format of output using the **`-F`** option

  `-F "%.4f"` – Decimal format with 4 decimal places (1.2345)

  `-F "%.4e"` – Exponent format with 4 decimal places (1.2345E-03)

Main options

`-p key1,key2,…`	Keys to print
`-w key1=val1,key2!=val2,…`	Where option
`-m missingValue`	Specify missing value string
`-F / -L format`	C-style format for output values / lat-lon
`-f`	Do *not* fail on error (fails on error by default)
`-V`	Print ecCodes Version

Missing data values are not printed by default

`...`

grib_get_data – example

```
$ grib_get_data -F "%.4f" output.grib1
Latitude Longitude Value
 89.946    0.000 100952.6875
 89.946   18.000 100956.4375
 89.946   36.000 100960.6875
 89.946   54.000 100965.1875
 89.946   72.000 100968.9375
 89.946   90.000 100971.9375
 89.946  108.000 100973.9375
 89.946  126.000 100974.9375
 89.946  144.000 100974.1875
 89.946  162.000 100972.6875
...
```

Format -F option applies to values only - not to the Latitudes and Longitudes

grib_get_data – format latitude and longitudes

```
$ grib_get_data -L"%9.6f %9.6f" -F "%.4f" output.grib1
Latitude Longitude Value
89.946188  0.000000 100952.6875
89.946188 18.000000 100956.4375
89.946188 36.000000 100960.6875
89.946188 54.000000 100965.1875
89.946188 72.000000 100968.9375
89.946188 90.000000 100971.9375
89.946188 108.000000 100973.9375
89.946188 126.000000 100974.9375
89.946188 144.000000 100974.1875
89.946188 162.000000 100972.6875
89.946188 180.000000 100969.9375
89.946188 198.000000 100966.4375
...
```

grib_get_data – missing values example

- Missing values are not printed by default

```
$ grib_get_data -m XXXXX -F "%.4f" f1.grib2
Latitude Longitude Value
...
   81.000    90.000 9.4189
   81.000    91.500 8.6782
   81.000    93.000 XXXXX
   81.000    94.500 XXXXX
   81.000    96.000 XXXXX
   81.000    97.500 XXXXX
   81.000    99.000 6.7627
   81.000   100.500 7.4097
   81.000   102.000 7.9307
...
```

Missing values are printed with XXXXX

Questions ?

Practicals

- Work on ecs-login or hpc-login in your $SCRATCH

```
cd $SCRATCH
```

- Make a copy of the practicals directory in your $SCRATCH

```
tar -xvf /home/trx/grib_decoding.tar
```

- This will create a directory in your $SCRATCH containing the GRIB data files for all today's practicals
- There are sub-directories for each practical:

```
ls $SCRATCH/grib_decoding
```

```
demo inspect modify python
```

- Remember to load the ecmwf-toolbox !

```
module load ecmwf-toolbox/new
```

Your turn !

Inspect:

- https://confluence.ecmwf.int/x/sAWaFQ

GRIB tools for modifying GRIB messages

- grib_copy
  - Copy selected messages from GRIB files to new files based on key values
  - Re-order GRIB messages in a file

- grib_set
  - Set key / value pairs in the input GRIB file
  - Make simple changes to key / value pairs in the input GRIB file
  - Make simple modifications to all data values

- grib_to_netcdf
  - convert GRIB messages to netCDF
  - Input GRIB fields must be on a regular grid: typeOfGrid=regular_ll or regular_gg
  - Used by MARS WebAPI to provide data in netCDF

grib_copy – copy contents of GRIB files

- Use grib_copy to copy selected contents of GRIB files optionally printing some key values
- Output can be ordered
  - E.g. order by ascending or descending step
- Key values can be used to specify the output file names

Main options

`-p key1,key2,…`	Keys to print (only with `-v`)
`-w key1=val1,key2!=val2…`	Where option
`-B "key asc, key desc"`	Order by: “step asc, centre desc”
`-f`	Do *not* fail on error
`-v`	Verbose
`-r`	Repack data when changing packing algorithm
`...`	

grib_copy – examples

- To copy only fields at 100 hPa from a file

```
$ grib_copy -w level=100 in.grib2 out.grib2
```

- To copy only those fields that are not at 100 hPa

```
$ grib_copy -w level!=100 in.grib2 out.grib2
```

- Information can be output using the –v and –p options

```
$ grib_copy -v -p shortName in.grib2 out.grib2
in.grib2
shortName
t
1 of 1 grib messages in in.grib2
1 of 1 total grib messages in 1 files
```

grib_copy – using key values in output file

- Key values can be used to specify the output file name

```
$ grib_copy in.grib "out_[shortName].grib"

$ ls out_*

out_2t.grib  out_msl.grib ...
```

Use quotes to protect the [ ]s

This provides a convenient way to filter GRIB messages into separate files

grib_set – set key / value pairs

- Use grib_set to
  - Set key / value pairs in the input GRIB file
  - Make simple changes to key / value pairs in the input GRIB file
- By default all GRIB messages in input file are written to the output file

Main options

`-s key1=val1,key2=val2,…`	List of key / values to set
`-p key1,key2,…`	Keys to print (only with **`-v`**)
`-w key1=val1,key2!=val2…`	Where option
`-d value`	Set all data values to **`value`**
`-f`	Do *not* fail on error
`-v`	Verbose
`-S`	Strict – copy only messages matching all constraints
`-r`	Repack data when changing packing algorithm

`...`

grib_set – examples

- To set the parameter value of a field to 10m wind speed (10si)

```
$ grib_set -s shortName=10si 10u.grib2 10si.grib2
```

- This changes e.g.
  - shortName to 10si
  - paramId to 207
  - name / parameterName to ‘10 metre wind speed’
  - units / parameterUnits to ‘m s ** -1’
  - parameterNumber to 1

grib_set – examples

- Some keys are read-only and cannot be changed directly

```
$ grib_set -s name="10 metre wind speed" in.grib2 out.grib2

ECCODES ERROR   :  grib_set_values[0] name (3) failed: Value is read only
```

- The read-only keys can only be set by setting one of the other keys, e.g.
  - shortName=10si
  - paramId=207
  - indicatorOfParameter=207

grib_set – modify data values

- An offset can be added to all data values in a GRIB message by setting the key offsetValuesBy

```
$ grib_get -F "%.5f" -p max,min,average TK.grib
315.44727 216.96680 286.34257

$ grib_set -s offsetValuesBy=-273.15 TK.grib TC.grib

$ grib_get -F "%.5f" -p max,min,average TC.grib
42.29726 -56.18321 13.19257
```

grib_set – modify data values

- Data values in a GRIB message can be multiplied by a factor by setting the key scaleValuesBy

```
$ grib_get -F "%.2f" -p max,min,average Z.grib2
65035.92 -3626.08 2286.30

$ grib_set -s scaleValuesBy=0.102 Z.grib2 orog.grib2

$ grib_get -F "%.2f" -p max,min,average orog.grib2
6633.64 -369.86 233.20
```

grib_set – using key values in output file

- Key values can be used to specify the output file

```
$ grib_set -s time=0000 in.grib "out_[shortName].grib"


$ ls out_*
out_2t.grib  out_msl.grib ...
```

Remember: use quotes to protect the []s !

What cannot be done with grib_set

- grib_set cannot be used for making transformations to the data representation
  - It cannot be used to transform data from spectral to grid-point representation (and vice-versa)
- grib_set cannot be used transform data from one grid representation to another
  - It cannot be used to transform data from regular or reduced Gaussian grids to regular latitude-longitude grids
- grib_set cannot be used to select sub-areas of data
  - It will change the value of, e.g. latitudeOfFirstGridPointInDegrees etc, but the data will still be defined on the original grid

✗ *GRIB tools cannot be used to transform, crop or regrid data*

grib_to_netcdf – convert GRIB to netCDF

- Use grib_to_netcdf to convert GRIB messages to netCDF
- Input GRIB fields must be on a regular grid: typeOfGrid=regular_ll or regular_gg

MainOptions

-o output_file	Output netCDF file
-R YYYYMMDD	Use **YYYYMMDD** as reference date (default: 19000101)
-D NC_DATATYPE	netCDF data type: NC_BYTE, NC_SHORT (default), NC_INT, NC_FLOAT or NC_DOUBLE
-k kind	Kind of file to be created: 1 → netCDF classic file format 2 → netCDF 64 bit classic file format (Default) 3 → netCDF-4 file format 4 → netCDF-4 classic model file format
-T	Do not use time of validity.
-u dimension	Set **dimension** to be an unlimited dimension
...	

Used by the MARS WebAPI to provide data in netCDF

grib_to_netcdf – examples

- To convert the fields in file.grib2 to netCDF

```
$ grib_to_netcdf -o out.nc file.grib2
grib_to_netcdf: Version 2.44.0
grib_to_netcdf: Processing input file 'file.grib2'.
grib_to_netcdf: Found 5 GRIB fields in 1 file.
grib_to_netcdf: Ignoring key(s): method, type, stream, refdate, hdate
grib_to_netcdf: Creating netCDF file 'out.nc'
grib_to_netcdf: NetCDF library version: 4.9.3 of Mar 14 2025 07:27:22 $
grib_to_netcdf: Creating large (64 bit) file format.
grib_to_netcdf: Defining variable 't'.
grib_to_netcdf: Done.

> ls -sh out.nc
50M out.nc
```

grib_to_netcdf – examples

- To convert the fields in file.grib2 to netCDF with data type set to NC_FLOAT

```
$ grib_to_netcdf -D NC_FLOAT -o out.nc file.grib2
grib_to_netcdf: Version 2.44.0
grib_to_netcdf: Processing input file 'file.grib2'.
grib_to_netcdf: Found 5 GRIB fields in 1 file.
grib_to_netcdf: Ignoring key(s): method, type, stream, refdate, hdate
grib_to_netcdf: Creating netCDF file 'out.nc'
grib_to_netcdf: NetCDF library version: 4.9.3 of Mar 14 2025 07:27:22 $
grib_to_netcdf: Creating large (64 bit) file format.
grib_to_netcdf: Defining variable 't'.
grib_to_netcdf: Done.

> ls -sh out.nc
79M out.nc
```

ecCodes user interfaces

- For some processing it is more convenient – or even necessary – to write a program
- The ecCodes library supports three user interfaces:
  - C: `#include <eccodes.h>`
  - Fortran 90 interface: `use eccodes`
  - Python interface: `import eccodes`
- At ECMWF two environment variables ECCODES_INCLUDE and ECCODES_LIB are defined to aid compilation and linking of Fortran 90 and C programs
- On ATOS HPC at ECMWF:

```
module load ecmwf-toolbox

gcc  myprog.c $ECCODES_INCLUDE $ECCODES_LIB -lm

gfortran myprog.f90 $ECCODES_INCLUDE $ECCODES_LIB
```

General framework for decoding

- Open one or more GRIB files (for read or write)
  - Standard Fortran calls cannot be used to open a GRIB file – you must use codes_open_file
- Calls to load one or more GRIB messages into memory
  - Two main routines: codes_grib_new_from_file / codes_new_from_index
  - These return a unique identifier used to manipulate the loaded GRIB messages
- Calls to decode the loaded GRIB messages – only loaded GRIB messages can be decoded
  - codes_get
  - Decode only what you need (not the full message !)
- Release the loaded GRIB messages:
  - codes_release
- Close the opened GRIB files
  - Standard Fortran calls cannot be used to close a GRIB file – you must use codes_close_file

ecCodes – Python API decoding example

```
import sys
from eccodes import *

# Load all the GRIB messages contained in file.grib1
ifile = open('file.grib2‘,’rb’)
while 1:
    igrib = codes_grib_new_from_file(ifile)
    if igrib is None: break

    # Decode data from the loaded message
    date = codes_get( igrib , “dataDate”)
    levtype = codes_get(igrib, “typeOfLevel”)
    level = codes_get(igrib, “level”)
    values = codes_get_array(igrib, “values”)
    print (f“{date} {levtype:20s} {level:5d} {values[0]:.5f} {values[-1]:.5f}”)

    # Release
    codes_release(igrib)
ifile.close()
```

Decode as little as possible !
You will never decode all the loaded GRIB message

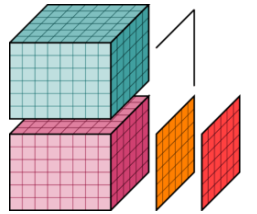
Loop on all the messages in a file. A new grib message is loaded from file. igrib is the grib id to be used in subsequent calls

Values returned as an array

Release the memory !

ecCodes and cfgrib

- cfgrib provides a Python interface to map GRIB files to the netCDF Common Data Model following the CF Convention using ecCodes.
- The high level enables the engine='cfgrib' option to read GRIB files with Xarray
- Inspired by netCDF4-python and h5netcdf
- ecCodes used for low-level decoding



xarray

- Install from conda

```
$ conda install -c conda-forge cfgrib
```

- Or PyPI

```
$ pip install cfgrib
```

https://github.com/ecmwf/cfgrib

```
>>> import xarray as xr
>>> ds = xr.open_dataset('era5-levels-members.grib',
engine='cfgrib')
>>> ds
<xarray.Dataset>
Dimensions:         (isobaricInhPa: 2, latitude: 61,
longitude: 120, number: 10, time: 4)
Coordinates:
  * number          (number) int64 0 1 2 3 4 5 6 7 8 9
  * time            (time) datetime64[ns] 2017-01-01
... 2017-01-02T12:00:00
    step            timedelta64[ns] ...
  * isobaricInhPa   (isobaricInhPa) float64 850.0 500.0
  * latitude        (latitude) float64 90.0 87.0 84.0
81.0 ... -84.0 -87.0 -90.0
  * longitude       (longitude) float64 0.0 3.0 6.0 9.0
... 351.0 354.0 357.0
    valid_time      (time) datetime64[ns] ...
```

GRIB and earthkit-data

- Format agnostic handling of geospatial data
- Fields in GRIB files abstracted to an array of values plus a dict-like set of metadata
- Accessed as a list of fields or converted to NumPy, Xarray or pandas

- Install from PyPI

```
$ pip install earthkit-data
```

- All code available on GitHub for open development

  https://github.com/ecmwf/earthkit-data

- Documentation

  https://earthkit.readthedocs.io/en/latest

- More details this afternoon !

```
>>> from earthkit.data import from_source
>>> filename="20251012000000-114h-enfo-cf.grib2"
>>> ds = from_source("file",filename)
>>> ds.ls()
  centre shortName        typeOfLevel  level  dataDate
dataTime stepRange dataType  number    gridType
0   ecmf         t      isobaricInhPa    400  20251012
0       114       cf       0  regular_ll
1   ecmf       hcc     highCloudLayer    450  20251012
0       114       cf       0  regular_ll
2   ecmf       skt            surface      0  20251012
0       114       cf       0  regular_ll
3   ecmf       sot          soilLayer      1  20251012
0       114       cf       0  regular_ll
4   ecmf       msl            meanSea      0  20251012
0       114       cf       0  regular_ll
5   ecmf       tcw   entireAtmosphere      0  20251012

...
```

ecCodes can do more…

- The idea is to provide a set of high-level keys or subroutines to derive / compute extra information from a loaded GRIB message
- For example:
  - keys (READ-ONLY) to return average, min, max of values, distinct latitudes or longitudes, etc …
  - Routines to compute the latitude, longitude and values
    - codes_grib_get_data
  - Routines to extract values
    - codes_grib_find_nearest: extract values closest to given geographical points
    - codes_get_element: extract values from a list of indexes
  - Routines for indexed access
    - Usually much faster than sequential access for “random” access

For lat/lon, Gaussian, reduced Gaussian grids. It is similar to the grib_get_data GRIB tool

Similar to “grib_ls -l “or “grib_get -l”

GRIB decoding – summary

Use GRIB Tools where possible	• It is not always necessary to write a program !
Use edition-independent keys	• Provides transparent access to GRIB 1 and GRIB 2 messages
If you do need to write a program think carefully about how the fields are accessed	• Indexed access can be much faster than sequential access

Questions ?

Your turn !

Modify:

- https://confluence.ecmwf.int/x/ogeaFQ

Python API:

- https://confluence.ecmwf.int/x/LwGwFQ