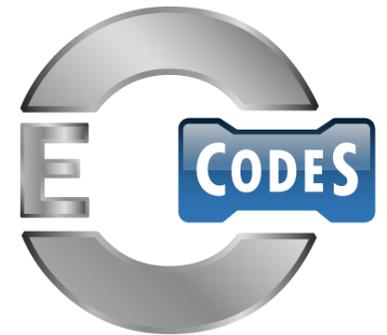


GRIB decoding with ecCodes

Introduction to ECMWF computing services

Paul Dando

Computing and Software Support Team



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

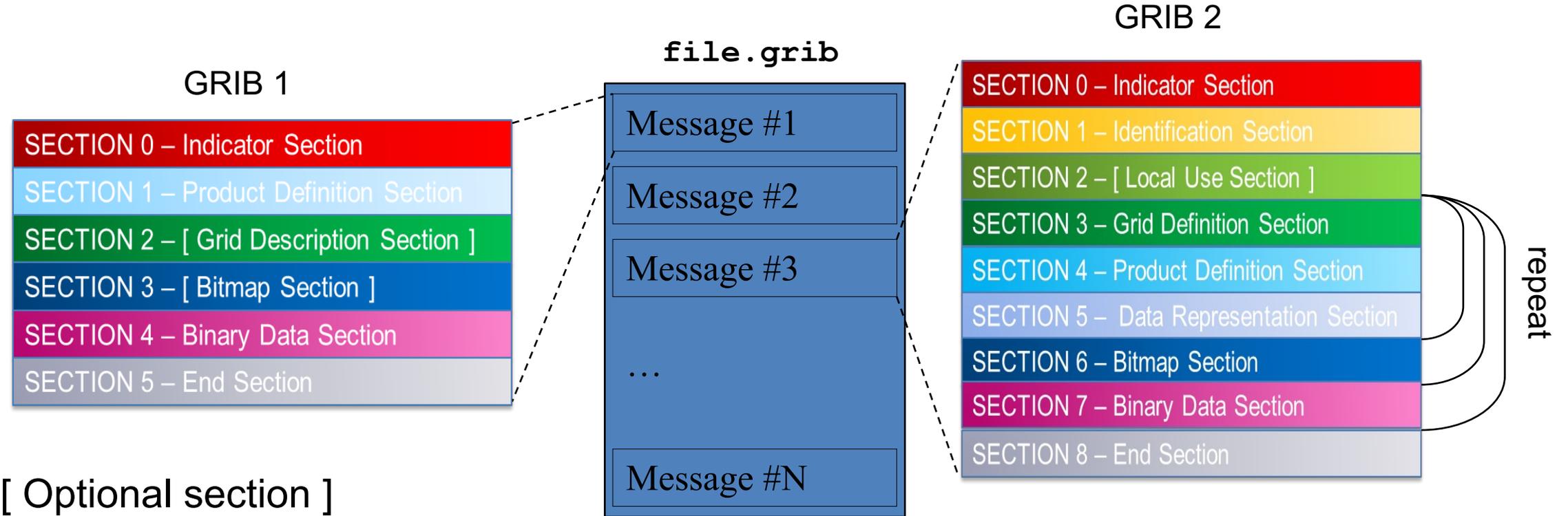
47 52 49 42	00 00 66 01	00 00 1C 01	62 01 FF 80	33 6D 00 01	06 0C	GRIB f b ~Ä3m
05 0C 00 0C	00 C8 05 00	00 00 15 00	00 00 00 00	32 02 2B 0A	00 F8	» 2 + `
01 90 80 33	C2 00 16 76	88 00 68 1A	00 76 F2 00	64 00 64 40	00 00	éÄ3- v à h vÚ d d@
00 00 80 55	F0 80 9C 40	00 00 00 00	43 3E B0 71	00 00 00 00	00 00	ÄU•Äú@ C>∞q
0C 08 80 11	3C 1F 09 7C	00 00 37 37	37 37			Ä < 7777

- 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 2026 !

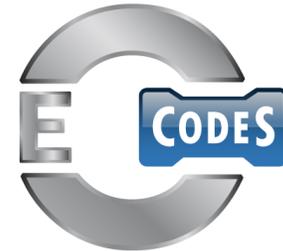
GRIB structure

- 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 "Ä3m
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	éÄ3- vâ 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	ÄUÄÚ@ C>∞q
0C 08 80 11 3C 1F 09 7C 00 00 37 37 37 37	Ä < 7777



[Optional section]



What is ecCodes ?

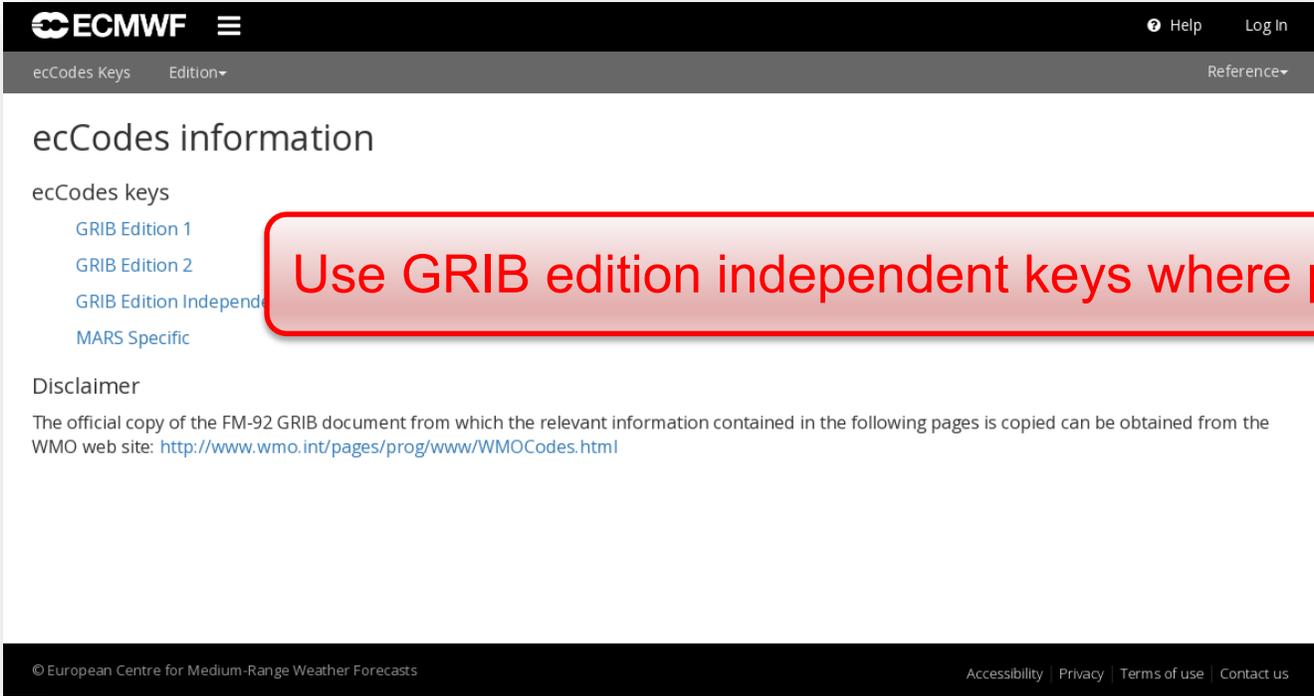
- **ecCodes** is a package developed by ECMWF for decoding and encoding messages in the following formats:
 - WMO FM-92 GRIB edition 1 and edition 2 (and edition 3)
 - WMO FM-94 BUFR edition 3 and edition 4
 - WMO GTS abbreviated header
- The package includes:
 - command line tools (the GRIB and BUFR 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/>



The screenshot shows the ECMWF website interface. At the top, there is a navigation bar with the ECMWF logo, a hamburger menu icon, and links for 'Help' and 'Log In'. Below this, a secondary navigation bar contains 'ecCodes Keys', 'Edition', and 'Reference'. The main content area is titled 'ecCodes information' and lists 'ecCodes keys' with sub-links for 'GRIB Edition 1', 'GRIB Edition 2', 'GRIB Edition Independent', and 'MARS Specific'. A 'Disclaimer' section follows, stating that the official copy of the FM-92 GRIB document can be obtained from the WMO website. The footer includes the copyright notice '© European Centre for Medium-Range Weather Forecasts' and links for 'Accessibility', 'Privacy', 'Terms of use', and 'Contact us'.

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")  
  
...
```

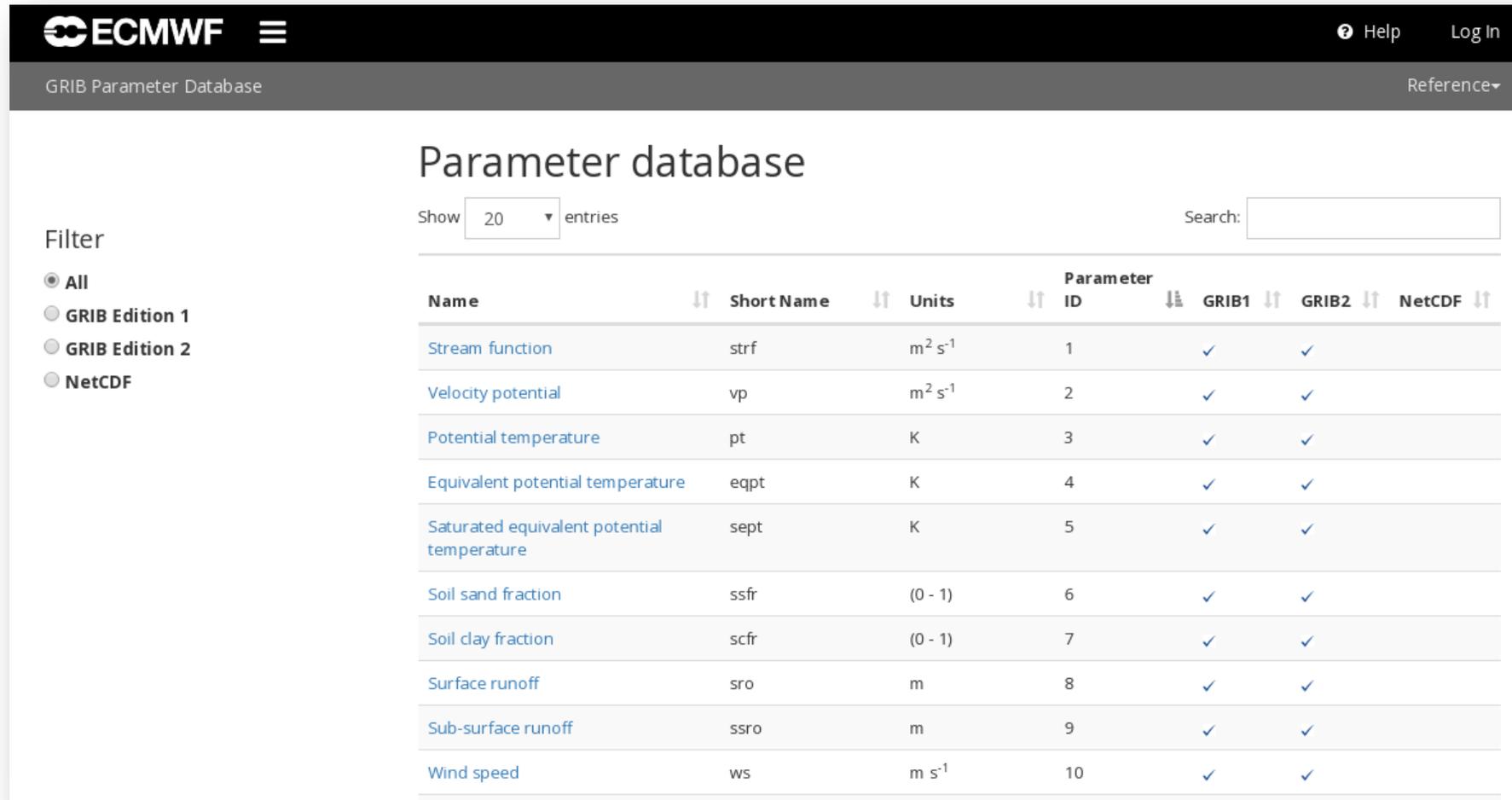


```
...  
  
shortName = codes_get(gid,"shortName")  
  
paramId = codes_get(gid,"paramId")  
  
units = codes_get(gid,"units")  
  
...
```



GRIB parameters – the Parameter Database

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



The screenshot shows the ECMWF GRIB Parameter Database interface. At the top, there is the ECMWF logo and navigation links for 'Help' and 'Log In'. Below the header, the page title 'GRIB Parameter Database' is displayed. The main content area is titled 'Parameter database' and includes a search bar and a 'Show 20 entries' dropdown. On the left, there is a 'Filter' section with radio buttons for 'All', 'GRIB Edition 1', 'GRIB Edition 2', and 'NetCDF'. The main part of the page is a table listing parameters with columns for Name, Short Name, Units, Parameter ID, GRIB1, GRIB2, and NetCDF. The table contains 10 entries, each with a checkmark in the GRIB1 and GRIB2 columns.

Name	Short Name	Units	Parameter ID	GRIB1	GRIB2	NetCDF
Stream function	strf	$\text{m}^2 \text{s}^{-1}$	1	✓	✓	
Velocity potential	vp	$\text{m}^2 \text{s}^{-1}$	2	✓	✓	
Potential temperature	pt	K	3	✓	✓	
Equivalent potential temperature	eqpt	K	4	✓	✓	
Saturated equivalent potential temperature	sept	K	5	✓	✓	
Soil sand fraction	ssfr	(0 - 1)	6	✓	✓	
Soil clay fraction	scfr	(0 - 1)	7	✓	✓	
Surface runoff	sro	m	8	✓	✓	
Sub-surface runoff	ssro	m	9	✓	✓	
Wind speed	ws	m s^{-1}	10	✓	✓	

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

The image displays three overlapping screenshots of the ECMWF Confluence documentation website. The top screenshot shows the 'ecCodes Home' page, created by Daniel Varela Santoalla, last modified by Shahram Najm on Jun 14, 2019. The middle screenshot shows 'BUFR examples' with a table of examples:

Example	Description
buf_r_attributes	How to read attributes of keys in BUFR messages.

The bottom screenshot shows 'GRIB examples' with a table of examples:

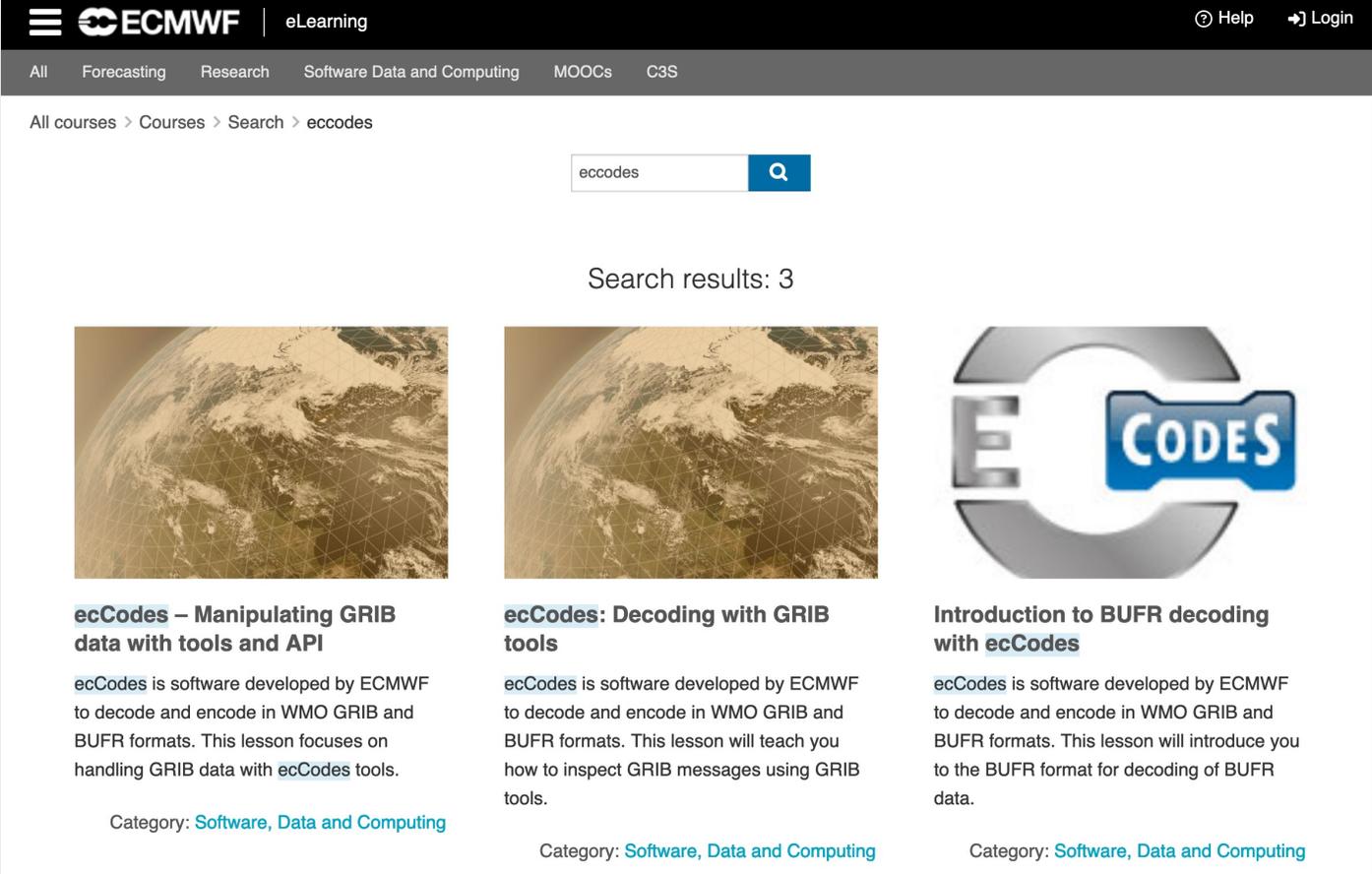
Example	Description
grib_clone	How to create a new GRIB message by cloning an existing message.

The bottom-most screenshot shows 'Frequently Asked Questions' for ecCodes GRIB, created by Faysz Abdul, last modified by Daniel Varela Santoalla on Oct 08, 2019. It lists various FAQs such as 'GRIB Keys - ecCodes GRIB FAQ', 'Keys and their types - ecCodes GRIB FAQ', 'What are namespaces - ecCodes GRIB FAQ', 'What is the GRIB key stepUnits - ecCodes GRIB FAQ', 'Failure setting key changeDecimalPrecision - ecCodes GRIB FAQ', 'What is the difference between count and countTotal - ecCodes GRIB FAQ', 'For GRIB edition 1, why does the centre change when I set stepType - ecCodes GRIB FAQ', 'Converting edition 1 to 2 - ecCodes GRIB FAQ', 'grib_compare: index based comparison - ecCodes GRIB FAQ', 'grib_compare: Is there an alternative to blocklisting keys - ecCodes GRIB FAQ', 'grib_filter/bufr_filter: How can I check if a key exists (is defined) in a message - ecCodes GRIB FAQ', 'grib_filter/bufr_filter: How do I control the format of array printing - ecCodes GRIB FAQ', 'grib_filter/bufr_filter: Do I always need to write a rules file - ecCodes GRIB FAQ', 'How can I change from second order to simple packing - ecCodes GRIB FAQ', 'How can I get the number of grib messages from my index selection - ecCodes GRIB FAQ', 'How can I multiply the field values by a constant - ecCodes GRIB FAQ', 'How can I remove the PV array (list of vertical coordinates) - ecCodes GRIB FAQ', 'How can I dump a given section - ecCodes GRIB FAQ', 'How can I see the contents of an index file - ecCodes GRIB FAQ', 'How do I append messages to an existing GRIB file - ecCodes GRIB FAQ', 'How do I copy a selected number of messages from a GRIB file - ecCodes GRIB FAQ', and 'How do I delete a message from a file - ecCodes GRIB FAQ'.

ecCodes eLearning resources

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

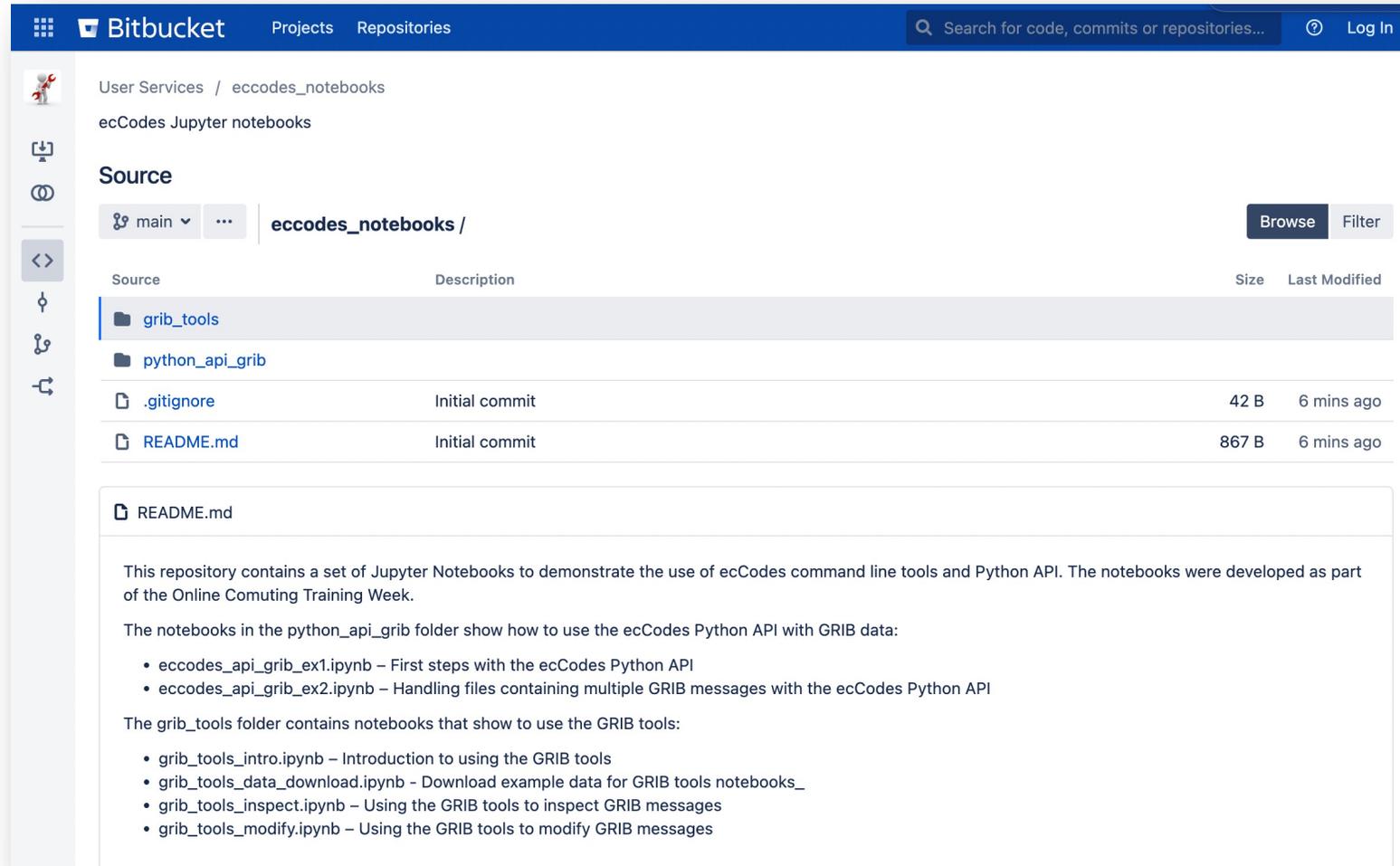
- ecCodes: Decoding with GRIB tools
- ecCodes: Manipulating GRIB data with tools and API



The screenshot shows the ECMWF eLearning website interface. At the top, there is a navigation bar with the ECMWF logo and 'eLearning' text. Below this is a secondary navigation bar with categories: All, Forecasting, Research, Software Data and Computing, MOOCs, and C3S. A search bar contains the text 'eccodes' and a search icon. Below the search bar, it says 'Search results: 3'. Three search results are displayed as cards. Each card features a satellite image of Earth with a grid overlay. The first card is titled 'ecCodes – Manipulating GRIB data with tools and API' and includes a description: 'ecCodes is software developed by ECMWF to decode and encode in WMO GRIB and BUFR formats. This lesson focuses on handling GRIB data with ecCodes tools.' and a category link: 'Category: Software, Data and Computing'. The second card is titled 'ecCodes: Decoding with GRIB tools' and includes a description: 'ecCodes is software developed by ECMWF to decode and encode in WMO GRIB and BUFR formats. This lesson will teach you how to inspect GRIB messages using GRIB tools.' and a category link: 'Category: Software, Data and Computing'. The third card is titled 'Introduction to BUFR decoding with ecCodes' and includes a description: 'ecCodes is software developed by ECMWF to decode and encode in WMO GRIB and BUFR formats. This lesson will introduce you to the BUFR format for decoding of BUFR data.' and a category link: 'Category: Software, Data and Computing'. To the right of the search results is a large logo for 'ECMWF CODES'.

ecCodes Jupyter Notebooks

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



The screenshot shows the Bitbucket web interface for the repository 'eccodes_notebooks'. The page displays a file browser view with a table of files and folders. The 'grib_tools' folder is selected. Below the table, the content of the 'README.md' file is shown, detailing the repository's purpose and the notebooks it contains.

Source	Description	Size	Last Modified
grib_tools			
python_api_grib			
.gitignore	Initial commit	42 B	6 mins ago
README.md	Initial commit	867 B	6 mins ago

README.md

This repository contains a set of Jupyter Notebooks to demonstrate the use of ecCodes command line tools and Python API. The notebooks were developed as part of the Online Computing Training Week.

The notebooks in the python_api_grib folder show how to use the ecCodes Python API with GRIB data:

- eccodes_api_grib_ex1.ipynb – First steps with the ecCodes Python API
- eccodes_api_grib_ex2.ipynb – Handling files containing multiple GRIB messages with the ecCodes Python API

The grib_tools folder contains notebooks that show to use the GRIB tools:

- grib_tools_intro.ipynb – Introduction to using the GRIB tools
- grib_tools_data_download.ipynb - Download example data for GRIB tools notebooks_
- grib_tools_inspect.ipynb – Using the GRIB tools to inspect GRIB messages
- grib_tools_modify.ipynb – Using the GRIB tools to modify GRIB messages



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_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

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

```
$ grib_ls file.grib1
```

```
file.grib1
```

edition	centre	typeOfLevel	level	dataDate	...	dataType	shortName	packingType	gridType
1	ecmf	isobaricInhPa	1000	20231022	...	an	t	spectral_complex	sh
1	ecmf	isobaricInhPa	500	20231022	...	an	t	spectral_complex	sh
1	ecmf	isobaricInhPa	200	20231022	...	an	t	spectral_complex	sh
1	ecmf	isobaricInhPa	100	20231922	...	an	t	spectral_complex	sh

```
4 of 4 grib messages in file.grib1
```

```
4 of 4 total grib messages in 1 files
```

```
$ grib_ls -p centre,dataDate,shortName,paramId,typeOfLevel,level file.grib1
```

```
file.grib1
```

centre	dataDate	shortName	paramId	typeOfLevel	level
ecmf	20231022	t	130	isobaricInhPa	1000
ecmf	20231022	t	130	isobaricInhPa	500
ecmf	20231022	t	130	isobaricInhPa	200
ecmf	20231022	t	130	isobaricInhPa	100

```
4 of 4 grib messages in file.grib1
```

```
4 of 4 total grib messages in 1 files
```

`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 file.grib1
file.grib1
my_key
not_found
```

```
$ echo $?
0
```

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/value3`
`-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.grib1 "IS"  
...  
$ grib_ls -w level!=100 file.grib1 "NOT"  
...  
$ grib_ls -w level=100,stepRange=3 file.grib1 "AND"  
...  
$ grib_ls -w level=100/200/300/500 file.grib1 "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 file.grib1
file.grib1
centre      dataDate    shortName   paramId     typeOfLevel  level
98          20231022   t           130         isobaricInhPa 1000
98          20231022   t           130         isobaricInhPa 500
98          20231022   t           130         isobaricInhPa 200
98          20231022   t           130         isobaricInhPa 100
4 of 4 grib messages in file.grib1

4 of 4 total grib 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 f1.grib1
```

```
f1.grib1
```

```
stepRange
```

```
6
```

```
12
```

```
...
```

```
$ grib_ls -s stepUnits=m -p stepRange f1.grib1
```

```
f1.grib1
```

```
stepRange
```

```
360
```

```
720
```

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`

```
grib_ls -l Latitude,Longitude,optionalMODE,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 file.grib1
```

```
***** FILE: file.grib1
#===== MESSAGE 1 ( length=4284072 ) =====
GRIB {
  editionNumber = 1;
  table2Version = 128;
  # European Center for Medium-Range Weather Forecasts (grib1/0.table)
  centre = 98;
  generatingProcessIdentifier = 141;
  # Geopotential (m**2 s**-2) (grib1/2.98.128.table)
  indicatorOfParameter = 129;
  # Surface (of the Earth, which includes sea surface) (grib1/3.table)
  indicatorOfTypeOfLevel = 1;
  level = 0;
  # Forecast product valid at reference time + P1 (P1>0) (grib1/5.table)
  timeRangeIndicator = 0;
  # Unknown code table entry (grib1/0.ecmf.table)
  subCentre = 0;
  paramId = 129;
  #-READ ONLY- units = m**2 s**-2;
  #-READ ONLY- nameECMF = Geopotential;
  #-READ ONLY- name = Geopotential;
  decimalScaleFactor = 0;
  dataDate = 20231022;
  dataTime = 0; ...
}
```

*Some keys are
read only*

*keys are case sensitive:
dataDate, dataTime*

grib_dump examples: WMO octet mode

```
$ grib_dump -O file.grib1
```

```
***** FILE: file.grib1
===== MESSAGE 1 ( length=4284072 ) =====
1-4  identifier = GRIB
5-7  totalLength = 4284072
8    editionNumber = 1
===== SECTION_1 ( length=52, padding=0 ) =====
1-3  section1Length = 52
4    table2Version = 128
5    centre = 98 [European Center for Medium-Range Weather Forecasts (grib1/0.table) ]
6    generatingProcessIdentifier = 141
7    gridDefinition = 255
8    section1Flags = 128 [10000000]
9    indicatorOfParameter = 129 [Geopotential (m**2 s**-2) (grib1/2.98.128.table) ]
10   indicatorOfTypeOfLevel = 1 [Surface (of the Earth, which includes sea surface) (grib1/3.table) ]
11-12 level = 0
13   yearOfCentury = 23
14   month = 10
15   day = 22
16   hour = 0
17   minute = 0
18   unitOfTimeRange = 1 [Hour (grib1/4.table) ]
...

```

grib_dump examples: Octet mode with types, aliases and Hex

```
$ grib_dump -OtaH file.grib1
```

```
***** FILE: file.grib1
===== MESSAGE 1 ( length=4284072 ) =====
1-4  ascii identifier = GRIB ( 0x47 0x52 0x49 0x42 )
5-7  g1_message_length totalLength = 4284072 ( 0x41 0x5E 0xA8 )
8    unsigned editionNumber = 1 ( 0x01 ) [ls.edition]
===== SECTION_1 ( length=52, padding=0 ) =====
1-3  section_length sectionLength = 52 ( 0x00 0x00 0x34 )
4    unsigned table2Version = 128 ( 0x80 ) [gribTablesVersionNo]
5    codetable centre = 98 ( 0x62 ) [European Center for Medium-Range Weather Forecasts(grib1/0.table) ]
      [identificationOfOriginatingGeneratingCentre,originatingCentre, ls.centre,
centreForTable2]
6    unsigned generatingProcessIdentifier = 141 ( 0x88 ) [generatingProcessIdentificationNumber, process]
7    unsigned gridDefinition = 255 ( 0xFF )
8    codeflag section1Flags = 128 [10000000] ( 0x80 )
9    codetable indicatorOfParameter = 129 ( 0x81 ) [Geopotential (m**2 s**-2)(grib1/2.98.128.table) ]
10   codetable indicatorOfTypeOfLevel = 1 ( 0x01 ) [Surface (of the Earth,which includes sea surface)
(grib1/3.table) ] [levelType, mars.levtype]
11-12 unsigned level = 0 ( 0x00 0x00 ) [vertical.topLevel vertical.bottomLevel, ls.level, lev]
13   unsigned yearOfCentury = 23 ( 0x17 )
14   unsigned month = 10 ( 0x0A )
15   unsigned day = 22 ( 0x16 )
16   unsigned hour = 0 ( 0x00 )
...
```

`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)

Usage

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

Missing data values are not printed by default

...

grib_get_data – example

```
$ grib_get_data -F "%.4f" f1.grib1
```

Latitude	Longitude	Value
81.000	0.000	22.5957
81.000	1.500	22.9009
81.000	3.000	22.8359
81.000	4.500	22.3379
81.000	6.000	21.5547
81.000	7.500	20.7344
81.000	9.000	19.8916
81.000	10.500	18.5747
81.000	12.000	17.2578
81.000	13.500	16.1343
81.000	15.000	14.9785
81.000	16.500	13.8296

...

*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" f1.grib1
```

```
Latitude Longitude Value
```

```
81.000000 0.000000 22.5957  
81.000000 1.500000 22.9009  
81.000000 3.000000 22.8359  
81.000000 4.500000 22.3379  
81.000000 6.000000 21.5547  
81.000000 7.500000 20.7344  
81.000000 9.000000 19.8916  
81.000000 10.500000 18.5747  
81.000000 12.000000 17.2578  
81.000000 13.500000 16.1343  
81.000000 15.000000 14.9785  
81.000000 16.500000 13.8296
```

```
...
```

grib_get_data – missing values example

- Missing values are not printed by default

```
> grib_get_data -m XXXXX -F "%.4f" f1.grib1
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
...
```



Questions ?



Practicals

- Work on ecs in your \$SCRATCH

```
cd $SCRATCH
```

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

```
tar -xvf /home/trx/grib_practicals.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_practicals
```

```
inspect modify python
```

- Remember to load the ecmwf-toolbox !

```
module load ecmwf-toolbox
```



Reception

Your turn !

Inspect:

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



Paul Dando - ECMWF



`grib_copy` – copy contents of GRIB files

- Use `grib_copy` to copy selected contents of GRIB files optionally printing some key values
- Without options `grib_copy` prints **no** key information
- Options exist to specify the set of keys to print
 - Use verbose option (`-v`) to print keys
- Output can be ordered
 - E.g. order by ascending or descending step
- Key values can be used to specify the output file names
- `grib_copy` fails if a key is not found
 - Use the `-f` option to force `grib_copy` not to fail on error

grib_copy – examples

- To copy only fields at 100 hPa from a file

```
$ grib_copy -w level=100 in.grib1 out.grib1
```

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

```
$ grib_copy -w level!=100 in.grib1 out.grib1
```

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

```
$ grib_copy -v -p shortName in.grib1 out.grib1
in.grib1
shortName
t
1 of 1 grib messages in in.grib1
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

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

`grib_set` – examples

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

```
$ grib_set -s shortName=10si in.grib1 out.grib1
```

- This changes e.g.
 - `shortName` to 10si
 - `paramId` to 207
 - `name` / `parameterName` to '10 metre wind speed'
 - `units` / `parameterUnits` to 'm s ** -1'
 - `indicatorOfParameter` to 207
 - `marsParam` to 207.128

grib_set – examples

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

```
$ grib_set -s name="10 metre wind speed" in.grib1 out.grib1
```

```
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 edition 1 dependent !

`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

- The 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.grib  
65035.92 -3626.08 2286.30
```

```
$ grib_set -s scaleValuesBy=0.102 Z.grib1 orog.grib1
```

```
$ grib_get -F "%.2f" -p max,min,average orog.grib1  
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

X *GRIB tools cannot be used to interpolate the data*

`grib_to_netcdf` – convert GRIB to netCDF

- Input GRIB fields must be on a regular grid
 - `typeOfGrid=regular_ll` or `regular_gg`

`-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.grib1 to NetCDF

```
$ grib_to_netcdf -o out.nc file.grib1
grib_to_netcdf: Version 2.2.0
grib_to_netcdf: Processing input file 'file1.grib1'.
grib_to_netcdf: Found 1 GRIB fields in 1 file.
grib_to_netcdf: Ignoring key(s): method, type, stream, refdate, hdate
grib_to_netcdf: Creating netCDF file 'out1.nc'
grib_to_netcdf: NetCDF library version: 4.3.2 of May 10 2016 11:12:41 $
grib_to_netcdf: Creating large (64 bit) file format.
grib_to_netcdf: Defining variable 't2m'.
grib_to_netcdf: Done.

> ls -s out.nc
160 out.nc
```

grib_to_netcdf – examples

- To convert the fields in file.grib1 to NetCDF with data type set to NC_FLOAT

```
$ grib_to_netcdf -D NC_FLOAT -o out.nc file.grib1
grib_to_netcdf: Version 2.2.0
grib_to_netcdf: Processing input file 'file1.grib1'.
grib_to_netcdf: Found 1 GRIB fields in 1 file.
grib_to_netcdf: Ignoring key(s): method, type, stream, refdate, hdate
grib_to_netcdf: Creating netCDF file 'out1.nc'
grib_to_netcdf: NetCDF library version: 4.3.2 of May 10 2016 11:12:41 $
grib_to_netcdf: Creating large (64 bit) file format.
grib_to_netcdf: Defining variable 't2m'.
grib_to_netcdf: Done.

> ls -s out.nc
316 out.nc
```

Output NetCDF file is about twice the size

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:

```
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 subroutines: `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 *
```

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

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

*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*

```
# 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 ("%s %s %d %.5f %.5f" % (date, levtype, level, values[0], values[-1]))
```

Values returned as an array

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

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



- 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] ...
```

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>

Paul Dando - ECMWF

