

Using GRIB Tools

Computer User Training Course 2015

Paul Dando & Carsten Maass

User Support

advisory@ecmwf.int



Contents

- **GRIB Tools basics and getting help**
- **Information tools**
- **Inspection tools**
- **Modification tools**
- **Getting key / value pairs**
- **Getting data values**
- **Comparing messages**
- **Copying messages**
- **Setting key / value pairs**

GRIB Tools – basic concepts

- **The GRIB tools are a set of command line programs for interactive and batch processing of GRIB data**
- **They provide ready and tested solutions to the most common processing of GRIB data**
- **Their use will avoid the need to write new code and thus speed up your work**
 - **Consider using GRIB Tools instead of writing your own program**
- **The tools are provided with a common set of options so that it is quick to apply the same options to different tools**
- **Use of the tools is recommended whenever possible!**

GRIB Tools – more basics

- **All of the tools use a common syntax**
 - `grib_<tool> [options] grib_file [grib_file] ... [output_grib]`
- **There is a tool for getting information about the GRIB API installation**
 - `grib_info`
- **There is a tool to count the messages in a GRIB file**
 - `grib_count`

GRIB Tools – more basics

- **There are tools to inspect the content of and compare GRIB files**
 - `grib_ls`, `grib_dump`, `grib_get`, `grib_get_data`, `grib_compare`
- **There is a tool to copy some messages**
 - `grib_copy`
- **There are tools to change the content of a GRIB message**
 - `grib_set`, `grib_filter`
- **There is a tool to convert a GRIB file to netCDF format**
 - `grib_to_netcdf`

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.
```

```
...
```

Documentation

- The GRIB API manual is available on the ECMWF website at

<https://software.ecmwf.int/wiki/display/GRIB/GRIB+API>

- The GRIB Tools are documented at

<https://software.ecmwf.int/wiki/display/GRIB/GRIB+tools>

Includes some examples of how to use the tools

- The WMO FM 92 GRIB Edition 1 and GRIB Edition 2 Manuals can be obtained from

<http://www.wmo.int/pages/prog/www/WMOCodes.html>

- The GRIB API software can be downloaded from

<https://software.ecmwf.int/wiki/display/GRIB/Releases>

grib_info – information about GRIB API

- The **grib_info** tool gives basic information about the GRIB API package being used
 - GRIB API Version
 - Path to definition files: **GRIB_DEFINITION_PATH**
 - Path to sample files: **GRIB_SAMPLES_PATH**

```
> grib_info
```

```
grib_api Version 1.13.0
```

```
Default definition files path is used:
```

```
  /usr/local/apps/grib_api/1.13.0/share/grib_api/definitions
```

```
Definition files path can be changed setting GRIB_DEFINITION_PATH  
environment variable
```

```
Default SAMPLES path is used:
```

```
  /usr/local/apps/grib_api/1.13.0/share/grib_api/samples
```

```
SAMPLES path can be changed setting GRIB_SAMPLES_PATH environment  
variable
```


GRIB keys

- For definitions of edition independent keys, GRIB1 or GRIB2 keys see

<https://software.ecmwf.int/wiki/display/GRIB/GRIB+API>

- Usage of edition independent keys should be preferred

`grib_count` – counts messages

- **Counts (very quickly) the number of messages in a list of GRIB files**
- **Syntax**

```
grib_count grib_file1 [grib_file2 ...]
```

(takes wildcards)

`grib_dump` – dump content of GRIB files

- Use `grib_dump` to dump the content of a file containing one or more GRIB messages
- Various output formats are supported
 - `Octet mode` provides a WMO documentation style dump
 - `Debug mode` prints all keys available in the GRIB file
 - `Octet` and `Debug modes` cannot be used together
 - Octet content can also be printed in hexadecimal format
- Options also exist to print key `aliases` and key `type` information
- Output to JSON (JavaScript Object Notation)
 - Easy to process

grib_dump – usage

```
grib_dump [options] grib_file grib_file ...
```

Basic options

- `-O` Octet mode
(WMO documentation style)
- `-a` Print key alias information
- `-t` Print key type information
- `-H` Octet content in Hexadecimal
- `-D` Debug mode
- `-w key[:{s|l|d}]{=|!=}value,...` Where clause
- `-j` JSON output
- `-V` Print GRIB API Version
- ...

grib_dump – examples

```
> grib_dump file.grib1
```

```
***** FILE: file.grib1
#===== MESSAGE 1 ( length=3280398 ) =====
GRIB {
  editionNumber = 1;
  table2Version = 128;
  # European Center for Medium-Range Weather Forecasts (grib1/0.table)
  centre = 98;
  generatingProcessIdentifier = 139;
  # Geopotential (m**2 s**-2) (grib1/2.98.128.table)
  indicatorOfParameter = 129;
  # Isobaric level pressure in hectoPascals (hPa) (grib1/3.table)
  indicatorOfTypeOfLevel = 100;
  level = 1000;
  # 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 = 20110223;
  dataTime = 1200; ...
```

`grib_dump` – examples

```
> grib_dump -O file.grib1
```

```
***** FILE: file.grib1
```

```
#===== MESSAGE 1 ( length=3280398 ) =====  
1-4      identifier = GRIB  
5-7      totalLength = 3280398  
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 = 139  
7        gridDefinition = 255  
8        section1Flags = 128 [10000000]  
9        indicatorOfParameter = 129 [Geopotential (m**2 s**-2)  
                                                (grib1/2.98.128.table) ]  
10       indicatorOfTypeOfLevel = 100 [Isobaric level pressure in hectoPascals  
                                                (hPa) (grib1/3.table) ]  
11-12    level = 1000  
13       yearOfCentury = 11  
14       month = 2  
15       day = 23  
16       hour = 12 ...
```

grib_dump – examples

```
> grib_dump -OtaH file.grib1
```

```
***** FILE: file.grib1
#===== MESSAGE 1 ( length=3280398 ) =====
1-4      ascii identifier = GRIB ( 0x47 0x52 0x49 0x42 )
5-7      g1_message_length totalLength = 3280398 ( 0x32 0x0E 0x0E )
8        unsigned editionNumber = 1 ( 0x01 ) [ls.edition]
===== SECTION_1 ( length=52, padding=0 ) =====
1-3      section_length section1Length = 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 = 139 ( 0x8B )
      [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 = 100 ( 0x64 ) [Isobaric level pressure in
      hectoPascals (hPa) (grib1/3.table) ] [levelType, mars.levtype]
11-12    unsigned level = 1000 ( 0x03 0xE8 ) [vertical.topLevel,
      vertical.bottomLevel, ls.level, lev, mars.levelist]
13       unsigned yearOfCentury = 11 ( 0x0B )
14       unsigned month = 2 ( 0x02 )
15       unsigned day = 23 ( 0x17 )
16       unsigned hour = 12 ( 0x0C )
17       unsigned minute = 0 ( 0x00 ) . . .
```

grib_dump – examples

```
> grib_dump -D file.grib1
```

```
***** FILE: file.grib1
#===== MESSAGE 1 ( length=9358 )
:
=====> section GRIB (9358,9358,0)
0-0 constant ieeeFloats = 0
=====> section section_0 (0,0,0)
----> label empty
<==== section section_0
0-4 ascii identifier = GRIB
4-7 gl_message_length totalLength = 9358
7-8 unsigned editionNumber = 1 [ls.edition]
=====> section section_1 (52,52,0)
:
36-36 gldate dataDate = 20110223 [mars.date, time.dataDate]
36-36 evaluate year = 2011
36-36 time dataTime = 1200 [mars.time]
36-36 julian_day julianDay = 2.45562e+06
36-36 codetable stepUnits = 1 [Hour (stepUnits.table) ]
36-36 concept stepType = instant
36-36 glstep_range stepRange = 0 [time.stepRange]
36-36 long_vector startStep = 0
36-36 long_vector endStep = 0 [stepInHours, mars.step]
36-36 mars_param marsParam = 129.128 [mars.param]
36-36 validity_date validityDate = 20110223
36-36 validity_time validityTime = 1200
...

```

In debug mode computed keys are shown

*ls.<key> ,
mars.<key> and
time.<key> denote
keys in
namespaces*

Practical

- **Work in your \$SCRATCH**

```
cd $SCRATCH
```

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

```
tar -xvf /home/ectrain/trx/grib_api/grib_tools.tar
```

- **This will create a directory in your \$SCRATCH containing the GRIB data files for all the practicals**

- **There is a sub-directory for each practical:**

```
ls $SCRATCH/grib_tools
```

```
grib_compare  grib_copy    grib_dump    grib_get
```

```
grib_ls
```

```
grib_set    . . .
```

Practical: using grib_dump

- **Use the web documentation to look at the different keys available for type GRIB1 and type GRIB2 messages**
 - Identify some keys common to both GRIB1 and GRIB2
- **Experiment with using the different `grib_dump` options (`-O`, `-a` and `-t`). Inspect the GRIB message in the files `file1.grib1` and `file1.grib2` and identify:**
 - the GRIB edition used to encode the messages
 - the (MARS)parameter ID, date, time, forecast step and the grid geometry
- **What are the maximum, minimum and average values of the fields?**

`grib_ls` – list the content of GRIB files

- Use `grib_ls` to list the content of GRIB files
- Without options `grib_ls` prints a default list of keys
 - The default list printed is different for GRIB 1 and GRIB 2
- Options exist to specify the set of keys to print or to print other keys in addition to the default set
- Output can be ordered
 - e.g. order by ascending or descending step
- `grib_ls` does not fail if a key is not found
- `grib_ls` can also be used to find the grid point(s) nearest to a specified latitude-longitude and print the value of the field at that point(s)
 - Modes available to obtain one or four nearest grid points

grib_ls – usage

```
grib_ls [options] grib_file grib file ...
```

Options

- | | |
|--|---|
| <code>-p key[:{s l d}],...</code> | Keys to print |
| <code>-P key[:{s l d}],...</code> | Additional keys to print |
| <code>-w key[:{s l d}]{= !=}value,...</code> | Where clause |
| <code>-B "key asc, key desc..."</code> | Order by: "step asc, centre desc" |
| <code>-n namespace</code> | Print all the keys belonging to namespace (ls, parameter, statistics, geography, time, mars, vertical) |
| <code>-m</code> | Print MARS keys |
| <code>-W width</code> | Minimum column width (default 10) |
| <code>...</code> | |

grib_ls – examples

```
> grib_ls file.grib2
```

```
file.grib2
```

edition	centre	date	...	gridType	...	typeOfLevel	level	shortName	packingType
2	ecmf	20110226	...	reduced_gg	...	isobaricInhPa	1000	q	grid_simple
2	ecmf	20110226	...	reduced_gg	...	isobaricInhPa	850	q	grid_simple
2	ecmf	20110226	...	reduced_gg	...	isobaricInhPa	700	q	grid_simple
2	ecmf	20110226	...	reduced_gg	...	isobaricInhPa	500	q	grid_simple

4 of 4 grib messages in file1.grib2

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

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

```
> grib_ls -p centre:l,dataDate,shortName,paramId,typeOfLevel,level file.grib2
```

```
file.grib2
```

Centre	dataDate	shortName	paramId	typeOfLevel	level
98	20110226	q	133	isobaricInhPa	1000
98	20110226	q	133	isobaricInhPa	850
98	20110226	q	133	isobaricInhPa	700
98	20110226	q	133	isobaricInhPa	500

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



- Similar behaviour to grib_get (see later)
 - grib_ls is better for interactive use
 - use grib_get within scripts

Using the 'where' option

- The 'where option' `-w` can be used with all GRIB Tools
- Constraints are of the form `key=value` or `key!=value`

`-w key[:{s|l|d}]=value, key[:{s|l|d}]!=value`

- Messages are processed only if they match ALL key/value constraints
- Values separated by / represent "OR" condition

```
> grib_ls -w levelType=pl file.grib1
```

```
...
```

```
> grib_ls -w step!=6,level=700/850 file.grib1
```

```
...
```

```
> grib_ls -w count=3 file.grib1
```

Practical: using grib_ls

- Use **grib_ls** to inspect the files **msl.grib1** and **msl.grib2**
 - Which keys does **grib_ls** show by default for the two files ?
 - What fields do they contain ?
- Use **grib_ls** to print the MARS keys
- Use **grib_ls** with other namespaces
- Use **grib_ls** to order the output in descending step order
- Use **grib_ls** to print the centre, **dataDate**, **stepRange**, **levelType**, **shortName** and **paramId** for both files
 - Experiment with both **-P** and **-p** options and **'key:l'**, **'key:s'**

Finding nearest grid points with grib_ls

- The value(s) of a GRIB field close to the point of a Latitude/Longitude can be found with **grib_ls**

```
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

Practical: using grib_ls –I

- The file `msl.grib1` contains the mean sea-level pressure from the EPS control forecast at 6-hourly time steps for the first 24 hours on a N100 regular Gaussian grid
- Find the value of the MSLP at the grid point nearest to ECMWF (Lat 51.42°N, Lon 0.95° W) at each forecast step
 - What is the lat-lon value of the grid point nearest to ECMWF ?
 - How far is the chosen grid point from ECMWF ?
- Change the command used to output only the forecast step and the MSLP value at the nearest grid point
- Change the command to output the MSLP values at the four grid points nearest to ECMWF
- Use the file `lsm.grib1` to provide a land-sea mask
 - Are all four nearest grid points land points (mask ≥ 0.5) ?

GRIB Examiner (Metview 4)



- **Interactive examiner using GRIB API**
- **Actively developed and maintained by the Metview team**
- **Can be started up from the command line. E.g. on ecgate use**

```
metview -e grib your_grib_file
```

GRIB Examiner: The user interface

File: /home/graphics/cgr/metview/Local/UPlot/tz_60.grb
Permissions: -rw-r----- Owner: cgr Group: graphics Size: 3.4MB Modified: 2009-11-05 09:04
Total number of messages: 60

Index	Name	Date	Time	Step	Level	LevTyp
01	t	20090504	1200	0	1000	pl
02	z	20090504	1200	0	1000	pl
03	t	20090504	1200	0	850	pl
04	z	20090504	1200	0	850	pl
05	t	20090504	1200	0	700	pl
06	z	20090504	1200	0	700	pl
07	t	20090504	1200	0	500	pl
08	z	20090504	1200	0	500	pl
09	t	20090504	1200	0	400	pl
10	z	20090504	1200	0	400	pl
11	t	20090504	1200	0	300	pl
12	z	20090504	1200	0	300	pl
13	t	20090504	1200	12	1000	pl
14	z	20090504	1200	12	1000	pl
15	t	20090504	1200	12	850	pl
16	z	20090504	1200	12	850	pl
17	t	20090504	1200	12	700	pl
18	z	20090504	1200	12	700	pl
19	t	20090504	1200	12	500	pl
20	z	20090504	1200	12	500	pl
21	t	20090504	1200	12	400	pl
22	z	20090504	1200	12	400	pl
23	t	20090504	1200	12	300	pl
24	z	20090504	1200	12	300	pl
25	t	20090504	1200	24	1000	pl
26	z	20090504	1200	24	1000	pl
27	t	20090504	1200	24	850	pl
28	z	20090504	1200	24	850	pl
29	t	20090504	1200	24	700	pl
30	z	20090504	1200	24	700	pl

Meta data of the selected message
Dump mode: WMO-style dump

Position	Key name (GRIB API)	Value
Section 1		
Section 2		
1-3	section2Length	32
4	numberOfVerticalCoordinateVa...	0
5	pvLocation	255
6	dataRepresentationType	0 [Latitude/Longitude Grid (grib1/6.table)]
7-8	Ni	240
9-10	Nj	121
11-13	latitudeOfFirstGridPoint	90000
14-16	longitudeOfFirstGridPoint	0
17	resolutionAndComponentFlags	128 [10000000]
18-20	latitudeOfLastGridPoint	-90000
21-23	longitudeOfLastGridPoint	358500
24-25	iDirectionIncrement	1500
26-27	jDirectionIncrement	1500
28	scanningMode	0 [00000000]
29-32	padding_grid_1	= 4 {
Section 4		
1-3	section4Length	58092
4	dataFlag	8 [00001000]
5-6	binaryScaleFactor	-9
7-10	referenceValue	236.152
11	bitsPerValue	16
12-5...	values	= (29040,58081) {
Section 5		

Log
Clear log

Task: Generating grib key list for message: 1
Method: GRIB API C interface
Status: OK

Task: Generating grib key list for all the messages
Method: GRIB API C interface
Status: OK

Status: OK

File information

Meta data (grib_dump)

Message list (with user defined GRIB API key selection)

Log

GRIB Examiner: Managing GRIB API keys

Insert/edit keys from header menu

iv Edit key ? - □ ×

Name:

Header:

Description:

✓ OK ✗ Cancel

Messages

Index	Name	Date	Time	Step	Level	LevTyp
01	t	20090504	1200	0	1000	pl
02	z	20090504	1200	0	1000	pl
03	t	20090504	1200	0	850	pl
04	z	20090504	1200	0	850	pl
05	t	20090504	1200	0	700	pl
06	z	20090504	1200	0	700	pl
07	t	20090504	1200	0	500	pl
08	z	20090504	1200	0	500	pl
09	t	20090504	1200	0	400	pl
10	z	20090504	1200	0	400	pl
11	t	20090504	1200	0	300	pl
12	z	20090504	1200	0	300	pl
13	t	20090504	1200	12	1000	pl

Meta data of the selected message

Dump mode: WMO-style dump

Tree view Plain text

Position	Key name (GRIB API)	Value
+ Section 1		
- Section 2		
1-3	section2Length	32
4	numberOfVerticalCoordinateVa...	0
5	pvlLocation	255
6	dataRepresentationType	0 [Latitu
7-8	Ni	240
9-10	Nj	121
11-13	latitudeOfFirstGridPoint	90000
14-16	longitudeOfFirstGridPoint	0

Drag and drop a new key

`grib_get` – get key / value pairs

- Use `grib_get` to get the values of one or more keys from one or more GRIB files – very similar to `grib_ls`
- By default `grib_get` **fails** if an error occurs (e.g. key not found) returning a non-zero exit code
 - Suitable for use in scripts to obtain key values from GRIB messages
 - Can force `grib_get` not to fail on error
- Options available to get all MARS keys or all keys for a particular namespace
 - Can get other keys in addition to the default set
- Format of floating point values can be controlled with a C-style format statement

grib_get – usage

```
grib_get [options] grib_file grib_file ...
```

- Options

<code>-p key[:{s l d}],...</code>	Keys to get
<code>-P key[:{s l d}],...</code>	Additional keys to get with <code>-m</code> , <code>-n</code>
<code>-w key[:{s l d}]{=/{!}=}value,...</code>	Where option
<code>-s key[:{s/l/d}]=value,...</code>	Keys to set
<code>-m</code>	Get all MARS keys
<code>-n namespace</code>	Get all keys for <code>namespace</code>
<code>-l lat,lon[,MODE,FILE]</code>	Value(s) nearest to lat-lon point
<code>-F format</code>	Format for floating point values
<code>-f</code>	Do <i>not</i> fail on error

...

grib_get – examples

- To get the centre of the first (**count=1**) GRIB message in a file (both as a 'string' and a 'long')

```
> grib_get -w count=1 -p centre f1.grib1
ecmf

> grib_get -w count=1 -p centre:l f1.grib1
98
```

- `grib_get` fails if there is an error

```
> grib_get -p mykey f1.grib1
GRIB_API ERROR      : Key/value not found

> echo $?
246
```

← returns the exit code from the previous command

grib_get – examples

- To get all the MARS keys, optionally printing the `shortName`

```
> grib_get -m fl.grib1
```

```
g sfc 20150223 1200 0 167.128 od an oper 0001
```

```
> grib_get -m -P shortName fl.grib1
```

```
2t g sfc 20150223 1200 0 167.128 od an oper 0001
```

- To get all keys belonging to the statistics namespace

```
> grib_get -n statistics fl.grib1
```

```
314.24 214.613 277.111 21.0494 41379.8 2.48314e-05 0
```

- `grib_get -m` is the same as `grib_get -n mars`

`grib_get` – controlling output format

- The format of floating point values can be controlled by using a C-style format statement with the `-F` option
 - `-F "%.4f"` - Decimal format with 4 decimal places (1.2345)
 - `-F "%.4e"` - Exponent format with 4 decimal places (1.2345E-03)

```
> grib_get -F "%.6f" -p maximum f1.grib1
```

```
314.240280
```

```
> grib_get -F "%.4e" -p maximum f1.grib1
```

```
3.1424e+02
```

- Default format is `-F "%.10e"`

`grib_get` – `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` appropriately with the `-s` option

```
> grib_get -p stepRange f1.grib1
```

```
6
```

```
12
```

```
> grib_get -s stepUnits=m -p stepRange f1.grib1
```

```
360
```

```
720
```

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

Finding nearest grid points with grib_get

- The value of a GRIB field close to a specified point of Latitude/Longitude can be found with `grib_get`
 - Works in the same way as `grib_ls`

```
> grib_get -l 52.0,-1.43 f1.grib1
```

```
273.58 272.375 273.17 273.531
```

```
> grib_get -F "%.5f" -P stepRange -l 52.0,-1.43,1 f1.grib1
```

```
0 272.37505
```

- GRIB files specified must contain grid point data

Getting data values at a grid point

- The value of a GRIB field at a particular grid point can be printed using `grib_get` with the `-i` option
- For example, find the index of a nearest grid point with `grib_ls` and then use this with `grib_get` to build a list of values at that point:

```
> grib_get -F "%.2f" -i 2159 -p step,dummy:s f1.grib1
```

```
6 99429.31  
12 99360.25  
18 99232.31  
24 99325.56
```

*Forces a space
between step and
value*

- Also returns a value for non-grid point data !

`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
- The format of the output can be controlled by using a C-style format statement with the `-F` option
 - `-F "%.4f"` - Decimal format with 4 decimal places (1.2345)
 - `-F "%.4e"` - Exponent format with 4 decimal places (1.2345E-03)The default format is `-F "%.10e"`
- By default missing values are not printed
 - A user-provided string can be printed in place of any missing values
- By default `grib_get_data` fails if there is an error
 - Use the `-f` option to force `grib_get_data` not to fail on error

grib_get_data – usage

```
grib_get_data [options] grib_file grib_file ...
```

- Options

<code>-p key[:{s l d}],...</code>	Keys to print
<code>-w key[:{s l d}]{=/{!}=}value,...</code>	Where option
<code>-m missingValue</code>	Specify missing value string
<code>-F format</code>	C-style format for output values
<code>-f</code>	Do <i>not</i> fail on error
<code>...</code>	

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 option
applies to values
only - not to the
Latitudes and
Longitudes*

grib_get_data – missing values example

```
> 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
```

```
...
```

*Missing values are
printed with
XXXXXX*

Practicals

- Work in your \$SCRATCH

```
cd $SCRATCH
```

- There is a sub-directory for each practical:

```
ls $SCRATCH/grib_tools
```

```
grib_compare  grib_copy  grib_dump
```

```
grib_filter  grib_get  grib_ls  grib_set
```

```
grib_to_netcdf
```

Practical: using `grib_get` & `grib_get_data`

1. Use `grib_get` to obtain a list of all the pressure levels available for parameter T in the file `tz_an_pl.grib1`
2. Use `grib_get` to print the `shortName`, `dataTime`, `dataDate` and `level` for the 500 & 1000 hPa levels only
3. Use `grib_get` to print the `stepRange` for the fields in the file `surface.grib1` in (a) hours (b) minutes and (c) seconds
4. Use `grib_get_data` to print the latitude, longitude and values for the first field in `surface.grib1`
 - Output results in decimal format with 5 decimal places
 - Output results in exponential format with 10 decimal places
5. Use `grib_get_data` to print the data values for the temperature at 500 hPa **only** from the file `tz_an_pl.grib1`
 - Make sure you print only the data for T500 ! What is printed ?

`grib_compare` – compare GRIB messages

- Use `grib_compare` to compare the GRIB messages contained in two files
- By default, messages are compared in the same order, bit-by-bit and with floating point values compared exactly
 - Tolerances for data values can be specified based on the absolute, relative or packing error
 - Default tolerance is absolute error = 0
- If differences are found `grib_compare`
 - switches to a key-based mode to find out which coded keys are different
 - **fails** returning a non-zero exit code
- Options are available to compare only specific keys or sets of keys

grib_compare – basic usage

```
grib_compare [options] grib_file grib_file
```

- Options

`-b key,key,...`

All keys in this list are skipped when comparing files

`-c key[:{s|l|d|n}],...`

Compare these keys only

`-H`

Compare message headers only

`-e`

Edition-independent compare

`-w key[:{s|l|d}] {=/{!=}value,...`

Where option

`-f`

Do *not* fail on error

`-r`

Messages not in the same order

`-v`

Verbose

...

`grib_compare` – a simple example

- Two GRIB messages in `f1.grib1` and `f2.grib1` contain the land-sea mask at different forecast time steps

```
> grib_compare f1.grib1 f2.grib1
-- GRIB #1 -- shortName=lsn paramId=172 stepRange=3
  levelType=sfc level=0 packingType=grid_simple
  gridType=reduced_gg --
long [P1]: [3] != [6]

> echo $?
1
```

- The exit code is set to 1 because the comparison failed

grib_compare – a simple example

- If we blacklist the key **P1** and compare the files again

```
> grib_compare -b P1 f1.grib1 f2.grib1
```

```
> echo $?
```

```
0
```

- The exit code is set to 0 because the comparison is successful according to the blacklist

`grib_compare` – verbose output

- The verbose option shows details of all keys being compared

```
> grib_compare -v f1.grib1 f2.grib1
f1.grib1
    comparing totalLength as long
    comparing editionNumber as long
    comparing section1Length as long
    comparing table2Version as long
    comparing centre as string
    comparing generatingProcessIdentifier as long
    comparing gridDefinition as long
    ...
    comparing P1 as long
-- GRIB #1 -- shortName=lsn paramId=172 stepRange=3 levelType=sfc
evel=0 packingType=grid_simple gridType=reduced_gg --
long [P1]: [3] != [6]
    comparing P2 as long
    ...
```


`grib_compare` – limit the keys compared

- The `-c` option can be used to compare only specific keys

```
> grib_compare -c dataDate f1.grib1 f2.grib1
-- GRIB #1 -- shortName=lsn paramId=172 stepRange=3
  levelType=sfc level=0 packingType=grid_simple
  gridType=reduced_gg --
long [dataDate]: [20150222] != [20150223]
```

- Or a set of keys in a particular namespace

```
> grib_compare -c time:n f1.grib1 f2.grib1
-- GRIB #1 -- shortName=lsn paramId=172 stepRange=3
  levelType=sfc level=0 packingType=grid_simple
  gridType=reduced_gg --
long [dataDate]: [20130222] != [20150223]
long [validityDate]: [20150222] != [20150223]
```

`grib_compare` – compare headers only

- To compare only the headers of two GRIB messages use the `-H` option

```
> grib_compare -H f1.grib1 f2.grib1
-- GRIB #1 -- shortName=lsn paramId=172 stepRange=3
  levelType=sfc level=0 packingType= gridType= --
long [day]: [22] != [23]
```

- The `-H` option cannot be used with the `-c` option

`grib_compare` – edition-independent

- Two GRIB messages are very different if they are encoded with different editions

```
> grib_compare sp.grib1 sp.grib2
```

```
-- GRIB #1 -- shortName=sp paramId=134 stepRange=0 levelType=sfc  
  level=0 packingType=grid_simple gridType=reduced_gg --  
long [totalLength]: [4284072] != [4284160]  
long [editionNumber]: [1] != [2]  
long [section1Length]: [52] != [21]  
[table2Version] not found in 2nd field  
[gridDefinition] not found in 2nd field  
[indicatorOfParameter] not found in 2nd field  
[indicatorOfTypeOfLevel] not found in 2nd field  
[yearOfCentury] not found in 2nd field  
[unitOfTimeRange] not found in 2nd field  
[P1] not found in 2nd field  
[P2] not found in 2nd field  
...
```

`grib_compare` – edition-independent

- Using the `-e` option `grib_compare` compares only the higher level information common to the two messages

```
> grib_compare -e sp.grib1 sp.grib2
-- GRIB #1 -- shortName=sp paramId=134 stepRange=0
  levelType=sfc level=0 packingType=grid_simple
  gridType=reduced_gg --
string [param]: [134.128] != [134]
```

- The two messages contain the same information encoded in two different ways
- Only the MARS param is different

grib_compare – summary of differences

- When files contain several fields and some keys are different, it is useful to have a summary report

```
> grib_compare -f f1.grib1 f2.grib1
-- GRIB #1 -- shortName=z paramId=129 stepRange=0 levelType=pl
   level=1000 packingType=spectral_complex gridType=sh --
long [marsType]: [an] != [fc]

-- GRIB #3 -- shortName=z paramId=129 stepRange=0 levelType=pl
   level=850 packingType=spectral_complex gridType=sh --
long [marsType]: [an] != [fc]
...
## ERRORS SUMMARY #####
##
## Summary of different key values
## marsType ( 6 different )
##
## 6 different messages out of 12
```

`grib_compare` – order-independent compare

- There are many errors if two files are compared which contain the same messages but in a different order

```
> grib_compare -f -H f1.grib1 f2.grib1
```

```
...
```

```
## ERRORS SUMMARY #####
```

```
##
```

```
## Summary of different key values
```

```
## indicatorOfParameter ( 6 different )
```

```
## level ( 7 different )
```

```
##
```

```
## 10 different messages out of 12
```

*By default
grib_compare
assumes messages
are in the same
order*

- To compare messages when they are not in the same order, use the `-r` option – this is VERY time expensive

```
> grib_compare -r -f -H f1.grib1 f2.grib1
```

`grib_compare` – comparing data values

- By default floating point values are compared exactly
- Different tolerances can be provided using one of the following options

`-A absolute_error`

Use absolute error as tolerance

`-R key=rel_error,...`

Use relative error as tolerance for `key`

`-P`

Use packing error as tolerance

`-T factor`

Compare data values using tolerance specified in options `-A`, `-R`, `-P` multiplied by an integer `factor`

`grib_compare` – setting the tolerance

- Comparison of the data values in two files shows that one of the seven values is different with the default absolute error tolerance of zero

```
> grib_compare -c data:n f1.grib1 f2.grib1
-- GRIB #1 -- shortName=2t paramId=167 stepRange=0 levelType=sfc level=0
   packingType=grid_simple gridType=reduced_gg --
double [packedValues]: 1 out of 7 different
   max absolute diff. = 2.0000000000000000e+00, relative diff. = 0.4
       max diff. element 2: 3.0000000000000000e+00
       5.0000000000000000e+00
       tolerance=0.0000000000000000e+00 packingError: [0.0625005]
       [0.0625005]
       values max= [70] [70]          min= [1] [1]
```

- Set the absolute error tolerance to 2.0 and the comparison is successful

```
> grib_compare -A 2.0 -c data:n f1.grib1 f2.grib1
```


grib_compare – setting the tolerance

- We can also set a relative error as tolerance for each key

```
> grib_compare -c data:n f1.grib1 f2.grib1
-- GRIB #1 -- shortName=2t paramId=167 stepRange=0 levelType=sfc level=0
   packingType=grid_simple gridType=reduced_gg --
double [packedValues]: 1 out of 7 different
   max absolute diff. = 2.0000000000000000e+00, relative diff. = 0.4
       max diff. element 2: 3.0000000000000000e+00
       5.0000000000000000e+00
       tolerance=0.0000000000000000e+00 packingError: [0.0625005]
       [0.0625005]
       values max= [70] [70] min= [1] [1]
values max= [70] [70] min= [1] [1]
```

- Set a relative error of 0.4 as the tolerance for packedValues

```
> grib_compare -R packedValues=0.4 -c data:n f1.grib1 f2.grib1
```

- The comparison is successful because the relative tolerance is greater than the relative difference

grib_compare – setting the tolerance

- Different packing precision can give different data values

```
> grib_compare -c data:n f1.grib1 f3.grib1
-- GRIB #1 -- shortName=2t paramId=167 stepRange=0 levelType=sfc level=0
   packingType=grid_simple gridType=reduced_gg --
double [packedValues]: 1 out of 7 different
   max absolute diff. = 5.0000000000000000e-01, relative diff. = 0.166667
       max diff. element 1: 2.5000000000000000e+00
       3.0000000000000000e+00
       tolerance=0.0000000000000000e+00 packingError: [0.0625005] [0.5]
       values max= [70] [70] min= [1] [1]
values max= [70] [70] min= [1] [1]
```

- Here we can use the packing error as the tolerance

```
> grib_compare -P -c data:n f1.grib1 f3.grib1
```

- The comparison is successful because the maximum absolute difference is within the larger of the two packing errors – only the packing precision has changed

Practical: using grib_compare

1. Use `grib_compare` to compare the GRIB messages contained in the files `file1.grib` and `file2.grib`
 - Which keys does `grib_compare` report as different ? What is the exit code returned ?
2. Now use the `-b` option to 'black list' the keys that you know are different and use `grib_compare` to compare the messages again
 - Are any keys reported as different ? What is the exit code ?
3. Compare the data namespaces for `file1.grib` and `file2.grib`. What values need to be set for the absolute (with `-A`) and relative (with `-R`) error tolerances for the comparison to be successful ?

`grib_copy` – copy contents of GRIB files

- Use `grib_copy` to copy selected messages from 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 – usage

```
grib_copy [options] grib_file grib_file ... out_grib_file
```

- Options

- p `key[:{s|l|d}] ,...` Keys to print (only with `-v`)
- w `key[:{s|l|d}]{=/{!}=}value ,...` Where option
- B `"key asc, key desc"` Order by: "step asc, centre desc"
- v Verbose
- f Do *not* fail on error
- ...

`grib_copy` – examples

- To copy only the analysis fields from a file

```
> grib_copy -w dataType=an in.grib1 out.grib1
```

- To copy only those fields that are not analysis fields

```
> grib_copy -w dataType!=an 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_ms1.grib  ...
```

*Use quotes to
protect the []s*

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

Practical: using `grib_copy`

1. The file `tz_an_pl.grib1` contains parameters T and Z on a set of pressure levels
 - Use `grib_copy` to create two files, one containing only the parameter T, the other containing the parameter Z
 - Check the content of the new files with `grib_ls`
 - Repeat, but output the messages so that the pressure levels in the new files are in increasing numerical order
2. Use `grib_copy` to split `tz_an_pl.grib1` into separate files for each parameter/level combination
 - Create files named `t_500.grib1`, `z_500.grib1`, etc

`grib_set` – set key / value pairs

- Use `grib_set` to
 - Set key / value pairs in the input GRIB file(s)
 - Make simple changes to key / value pairs in the input GRIB file(s)
- Each GRIB message is written to the output file
 - By default this includes messages for which no keys are changed
 - With `-s` (strict) option **only** messages matching **all constraints** in the where option are copied
- An option exists to repack data
 - Sometimes after setting some keys involving properties of the packing algorithm the data needs to be repacked
- `grib_set` **fails** when an error occurs
 - e.g. when a key is not found

grib_set – usage

```
grib_set [options] grib_file grib_file ... out_grib_file
```

- Options

<code>-s key[:{s l d}]=value,...</code>	List of key / values to set
<code>-p key[:{s l d}],...</code>	Keys to print (only with <code>-v</code>)
<code>-w key[:{s l d}]{=/{!}=}value,...</code>	Where option
<code>-d value</code>	Set all data values to value
<code>-f</code>	Do <i>not</i> fail on error
<code>-v</code>	Verbose
<code>-S</code>	Strict
<code>-r</code>	Repack data
<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 GRIB edition dependent !
 - `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
```

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

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

`grib_set` – set key values to missing

- When a key is not used all the bits of its value should be set to 1 to indicate that it is ‘missing’
- Different keys have different lengths so the value that needs to be coded for missing keys is not unique
- To set a key to missing a string "missing" or "MISSING" is accepted by `grib_set`

```
> grib_set -s Ni=missing in.grib2 out.grib2
```

- Note that some values cannot be set to “missing” !

```
> grib_set -s dataDate=missing file1.grib2 file2.grib2
GRIB_API ERROR      :  unable to set dataDate=missing (Value cannot
                        be missing)
GRIB_API ERROR      :  grib_set_values[0] dataDate (7) failed: Value
                        cannot be missing
```

`grib_set` – changing decimal precision

- To pack a temperature expressed in Kelvin with 1 digit of precision after the decimal point we can set `changeDecimalPrecision=1`
 - N.B. this is different to setting the number of significant digits !

```
> grib_set -s changeDecimalPrecision=1 T.grib1 T1.grib1
```

- Use `grib_compare` to see the differences

```
> grib_compare -c data:n T.grib1 T1.grib1
-- GRIB #1 -- shortName=2t paramId=167 stepRange=0 levelType=sfc level=0
  packingType=grid_simple gridType=reduced_gg --
double [packedValues]: 2132215 out of 2140702 different
  max absolute diff. = 5.0000000000011369e-02, relative diff. = 0.000207239
    max diff. element 17: 2.41216796875000000000e+02
      2.41266796875000011369e+02
        tolerance=0.0000000000000000e+00 packingError: [0.000984192]
          [0.0500122]
            values max= [315.447] [315.467]          min= [216.967] [216.967]
```

grib_set – changing the packing algorithm

- `grib_set` can be used to change the packing algorithm used from `grid_simple` (simple packing) to `grid_second_order` (2nd order packing)

```
> grib_set -r -s packingType=grid_second_order f1.grib2 \  
  f1_packed.grib2
```

- This can provide a very efficient level of compression

```
> ls -s f1.grib2 f1_packed.grib2 f1.grib2.bz2 | sort
```

```
1000 f1_packed.grib2  
1116 f1.grib2.bz2  
2616 f1.grib2
```

`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 name

```
> 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
- **The GRIB tools cannot be used to interpolate the data**

Practical: using grib_set

1. The GRIB messages in file.grib1 have been encoded with an incorrect date and time
 - Use [grib_set](#) to change the date to 24 February 2015 and the time to 00UTC. Check the new files with [grib_ls](#)
 - Repeat but change the date and time for T at 500hPa **only**
 - Repeat so that T at 500hPa **only** is written to the output file
2. An SST field has been created by masking the Soil Temperature at Level 1 (STL1) with the Land-Sea Mask and is included with other messages in the file surface.grib
 - Use [grib_set](#) to change the parameter for the field from STL1 to SST and level type to 'surface'
 - Be careful not to change the other parameters !
 - Repeat with each different message output to a separate file