Migration GRIB1 -> GRIB2: Short summary of important modifications

(last update 8/2018)

0. Introduction

GRIB2 is more complex than GRIB1 offering a basis to represent a lot more products in WMO standard (e.g. ensemble products) than GRIB1.

GRIB_API/EcCodes is a GRIB coding software (from ECMWF) with a key – value approach, enabling the user to code GRIB without detailed knowledge about its structure. Moreover the creation of edition independent keys makes it easier to migrate to GRIB2.

This document collects the important differences between GRIB1 and GRIB2 in GRIB_API (EcCodes) coding terms and helps to “understand” GRIB2 and to identify traps in the usage of GRIB_API.

It is beneficial if the reader has basic knowledge of GRIB_API (EcCodes) and GRIB(2) code. GRIB_API keys are written *cursive* and their values **bold**.

1. Basic differences of GRIB1 and GRIB2

Element/Parameter coding

GRIB1: *indicatorOfParameter, table2Version* (ee, tab)
GRIB2*: discipline* pdis*, parameterCategory* pcat*, parameterNumber* pnum (triple)

WMO regulation 92.6.2: To maintain orthogonal structure of GRIB Edition 2, parameter names in Code table 4.2 should **not** contain **surface type** and **statistical process** as part of the name.
Therefore variables do not get own triple (pdis, pcat, pnum) as in GRIB1, but the setting of level codes and statistical process codes make them distinguishable.
Examples:
+ Cloud cover CLCH, CLCM, CLCL (different level coding for same triple)
+ Precipitation amounts are coded as “rates” (triple) with statistical process “accumulation”

New product definition templates to code additional information for special variables (examples):
+ Satellite templates: observed (4.31; *shortNames* start with **OBS…**), synthetical (4.32; shortNames **SYN**….)

+ Chemical templates 4.40-4.43:	aerosols, nuclide
+ Distribution template 4.57, 4.58, 4.67, 4.68:	volcanic ash, mineral dust

Latitude, longitude and angles

Presentation of horizontal grids in higher precision (10**-3 -> 10**-6 degree) for
+ start and end points of grid
+ increments of grid
+ Geogr. latitude: +90 -> -90
+ Geogr. longitude: 0 -> 360 (GRIB1: -180 -> +180)

Therefore the grid increments are now representable in GRIB2.

Usage of units

Only SI units are used for GRIB2.
This is valid for parameter definitions and it is true for vertical coordinates as well, e.g. cm -> m, hPa -> Pa

Scale factor and scaled value

To satisfy the SI standards, scale factor and scaled value are defined for meta data according to **WMO regulation 92.1.12**: A scale factor **F** and a scaled value **V** are related to the original value **L** as follows: **$L \times 10^F = V$**

Vertical grid: Representation of level and layer

Differentiation of level and layer in GRIB2 is not realized with own code entry as in GRIB1, but with double setting of entries (Keys in GRIB2: *typeOf**First**FixedSurface, typeOf**Second**FixedSurface*).
Examples:

+ model levels/layers:	GRIB1 109 / 110	->	GRIB2 105 / 2*105
+ depth below land (level/layer):	GRIB1 111 / 112	->	GRIB2 106 / 2*106

Introduction of general vertical coordinate (*typeOfLevel=*‘**generalVertical/Layer**‘) with new keys *nlev, numberOfVGridUsed* and *uuidOfVGrid* (see Chapter 3.)

New meta data in GRIB2 (describing the product)

The new structure of GRIB2 allows the coding of a variety of new information not known in GRIB1.
Examples: GRIB_API keys with reference to WMO-Table (partly with local additions)

Key	**Description/Remark**	**Table**
significanceOfReferenceTime	Analysis (0), start of forecast (1), …	WMO: 1.2
productionStatusOfProcessedData	Entries for “operations”(0), parallel suite (1) and experiments (2)	WMO: 1.3
typeOfProcessedData	Classification of products : rough subdivision in analysis (0/an), forecast (1/fc), …	WMO: 1.4
typeOfGeneratingProcess	Detailed definition of generation, e.g. ‘intialization‘ (1) including local entries as ‘nudging‘ (202), ‘nvariant data‘ (196), …	WMO.: 4.3
backgroundGeneratingProcessIdentifier	Discrimination between main run (0), assimilation (2), pre-assimilation (1)	Local table
Information content of GRIB1 *localVersionNumber* = ipds(47) is put in *productionStatusOfProcessedData, backgroundGeneratingProcessIdentifier* and *localExperimentNumber* in GRIB2.		

In most cases changes in meta data concerning the variable definition will be unseen by usage (grib_get/grib_set) of “*shortName*” as *shortName* is edition independent.

2. Specific features and characteristics of GRIB_API

Basics (from ECMWF Training Course) for GRIB API keys

+ Each key has a native type (real, integer, string) and conversions are provided from one type to another when possible. So it is possible to represent the keys as string key:s, as INTEGER key:i or REAL key:d.
+ The set of keys available changes from one message to another as it depends on the content of the message and GRIB edition.
+ Changing the value of some keys can cause some other keys to disappear and new keys to be available.
+ The value of a key is not always coded in the GRIB message because it can be the result of the combination of several other keys through a given algorithm or just temporary (transient). Therefore we talk about
    ++ CODED keys (coded in the message as they are)
    ++ COMPUTED keys (temporary or computed from other keys)

Listing tools **grib_ls** and **grib_dump**

grib_ls (list of content of grib file): controlled by version and file content, e.g. selection of keys given out depends on the first field and is different for GRIB1 and GRIB2 and may change with GRIB_API version. Better: **grib_get**.

Summary of content with **grib_dump:** **grib_dump –O gribfile** for octet mode (WMO documentation style dump)

level

This is not a coded key (original in grib message), but an edition independent (valid for GRIB1 AND GRIB2) computed key generated by GRIB_API :
"level": *level = topLevel=bottomLevel*
"layer": *topLevel ≠ bottomLevel*
GRIB1: *level =bottomLevel* GRIB2: *level=topLevel*
problem: *typeOfLevel=***depthBelowLand /depthBelowLandLayer**
As in GRIB2 the unit „meter" (GRIB1: cm) is prescribed, values < 1m or real values are not presentable with *level(:i)* or *top/bottomLevel(:i)*, because GRIB_API converts to „meters" and presents integer values.
Here you have to take the "real" keys ***level:d; topLevel:d*** and ***bottomLevel:d*** or you can use the following GRIB2 coded keys:
scaleFactorOfFirstFixedSurface / scaledValueOfFirstFixedSurface
scaleFactorOfSecondFixedSurface / scaledValueOfSecondFixedSurface

*typeOfLevel=***isobaricInhPa /isobaricInPa**
In GRIB2 for pressure levels the SI unit 'Pa' is prescribed. GRIB_API defines two different level types (*typeOfLevel*) with level units in 'Pa' or 'hPa', so the user has the opportunity to choose.
The listing can be steered by setting the key *pressureUnits* to '**Pa**' or '**hPa**':
e.g. grib_get **–s *pressureUnits='Pa'*** -p *topLevel,bottomLevel* gribfile

step

The keys *stepType, stepUnits, step, startStep, endStep, stepRange* define the „type" (instant or statistical process), time unit and times related to reference time of one GRIB message. These are "computed" edition independent keys valid for GRIB1 and GRIB2:

stepType='instant': *step=endStep=startStep*

else *step=endStep* *with* *stepRange='endStep – startStep'* problem : Default for ***stepUnits*** in GRIB_API is **h**=‚hour‘; therefore ‚minutes‘ are not representable unless the time unit is set to **m**=‚minutes‘: **grib_get** –s ***stepUnits=m*** –p *step* gribfile

New with EcCodes (DWD implementation): Default for *stepUnits* is the inside the GRIB2 message coded key ***indicatorOfUnitOfTimeRange!!***

latitude and longitude

As mentioned before, coding of longitude and latitude will be in milli-degrees in GRIB1 and micro-degrees in GRIB2. GRIB_API provides edition independent keys, ending with **…InDegrees**, representing the values in degrees (real values), for example:

latitudeOfFirstGridPointInDegrees, iDirectionIncrementInDegrees

constant data

Handling of constant fields in GRIB_API: no packing of data values, reference value = constant value, the following keys are set:

isConstant *= 1* ***bitsPerValue*** *= 0*

Using the following environment variable, this behavior can be suppressed (for GRIB1 and GRIB2):

export **GRIB_API_LARGE_CONSTANT_FIELDS**=1

DWD will not use this for GRIB2, so for constant GRIB2 fields you will find fields with shorter grib length and *isConstant* = 1 and *bitsPerValue* = 0 !

3. Changes in COSMO output

For “single level fields” no more unnecessary vertical coordinates are coded, e.g. only variables on model levels are coded with the new “general vertical coordinate”.

See http://www.cosmo-model.org/content/model/modules/coding/grib/gribVerticalCoordinates.pdf and Release Notes COSMO 4.28, 5.01 and INT2LM 1.22, 2.01, 2.05.

New GRIB2 typeOfLevel "generalVertical" and "generalVerticalLayer"

The main difference to the other vertical coordinate types 'hybrid' and 'hybridLayer' is, that now there is no computational algorithm how the vertical layers are constructed, but a full 3D file (called a HHL-file for the height of half levels in COSMO) is provided, that has to be processed together with other 3D variables defined on these levels.
With this new vertical level type, also some new keys were introduced for the WMO GRIB2 standard, namely:

=> *nlev* (alias for numberOfVerticalCoordinateValues)
nlev is ke+1 in COSMO, because the number of half levels are specified here

=> *numberOfVGridUsed*
represents the vertical coordinate type ivctype

=> *uuidOfVGrid*
is a unique identifier for a special HHL-file

These keys replace the vertical coordinate parameters, which are present in data with level types 'hybrid' or 'hybridLayer'.

HHL-files can only be created by the INT2LM, not by the COSMO-Model.

In connection with the introduction of the new generalized vertical coordinate the fields **HHL** (height of model half levels) and **P** (pressure) are packed with **24** instead of 16 bits per value in GRIB2.

W_SO / W_SO_ICE: Coding of a layer is now possible in GRIB2, so *typeOfLevel* is changing from **depthBelowLand** to **depthBelowLandLayer** including new level/layer specifications.

Special heights above „mean sea level“ (MSL) are coded with t*ypeOfSecondFixedSurface*=**101** (e.g. HHL, HSURF, HBAS_CON, HTOP_CON, HTOP_DC)

4. Local configuration: Comments on GRIB_DEFINITION_PATH

The general (WMO standard) parameter definition (shortName.def, name.def, units.def, paramId.def) for GRIB2 is in

    /BASE_DIR/definitions/grib2/…

whereas the local definition for each centre is provided in

    /BASE_DIR/definitions/grib2/localConcepts/<centre>   , <centre> = ecmf, edzw etc.

A search for a definition file is done first in the "general" definition directory and second in the "localConcepts".
The installation of a local configuration is possible by setting the environment variable

    GRIB_DEFINITION_PATH

to use own local definitions instead of the definition files provided by ECMWF distribution. This procedure is recommended by ECMWF.

If a local configuration is installed by GRIB_DEFINITION_PATH=/BASE_DIR/definitions.edzw:/grib_api/definitions the search for each required definition file starts in /BASE_DIR/definitions.edzw. If the file is found there it will be used. Otherwise the search will be done in /grib_api/definitions.

5. References – Further Reading

WMO GRIB2:
http://www.wmo.int/pages/prog/www/WMOCodes.html

Wiki for GRIB API or EcCodes:
https://software.ecmwf.int/wiki/display/GRIB
https://confluence.ecmwf.int/display/ECC