GRIB API: Advanced Topics Part I

Shahram Najm
Development Section
Forecast Department

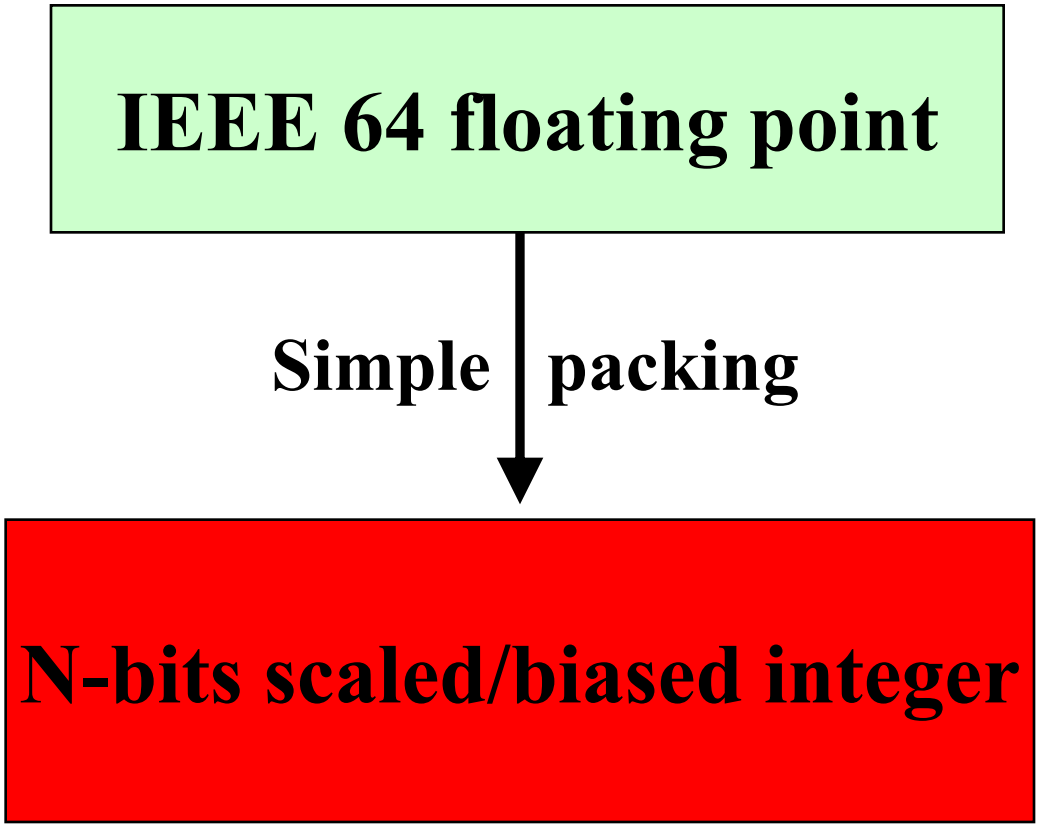
GRIB API

All you wouldn't like to know, but you must know

Overview

- Simple Packing
- Constant fields
- Bitmap
- Multi fields

Simple packing: Loss of information

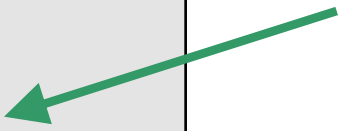


Usually N = 8, 10, 16, 24

Simple packing: Keys

- values
- decimalPrecision
- changeDecimalPrecision
- packingError (read only)

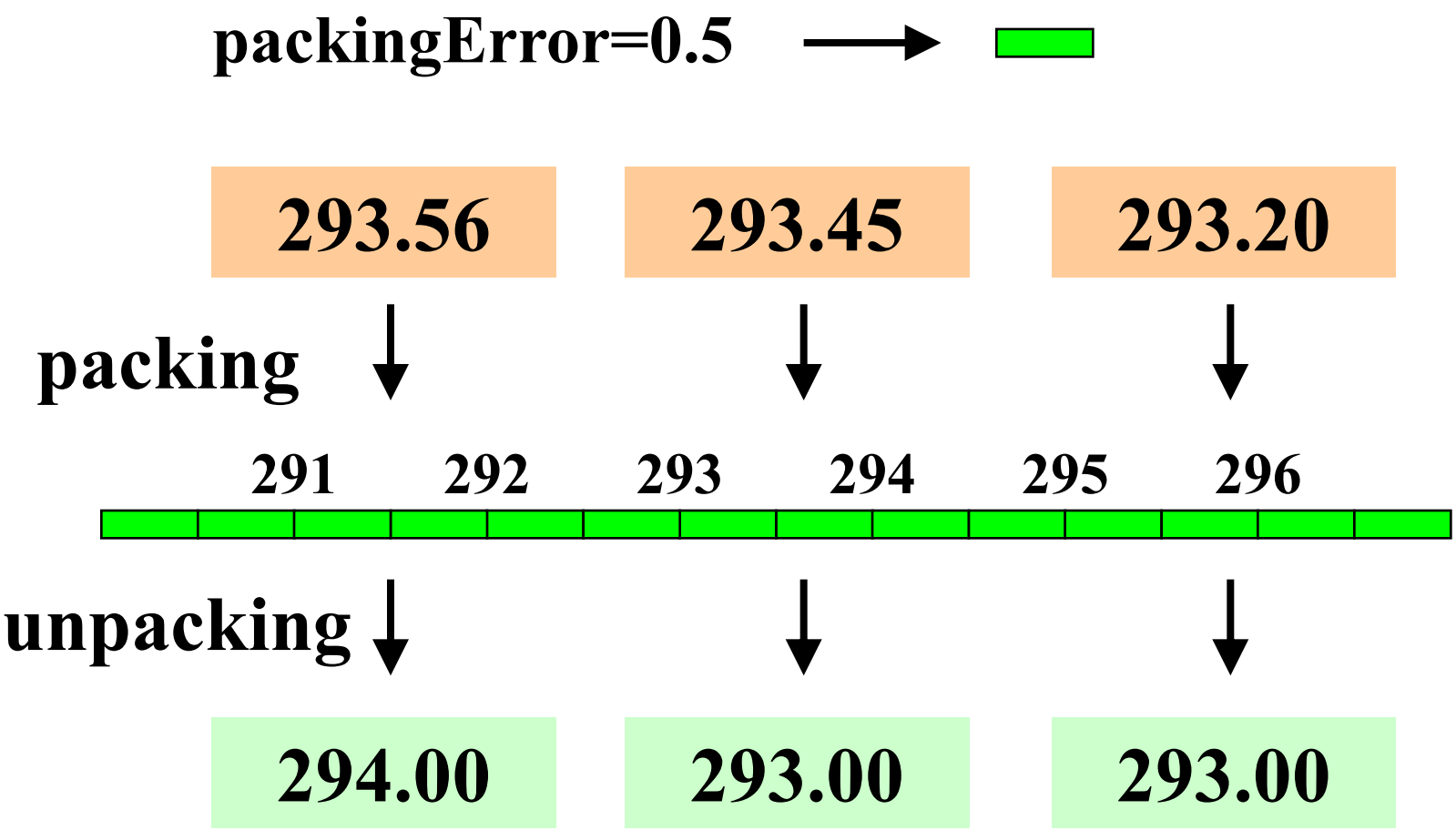
- referenceValue (read only)
- bitsPerValue
- decimalScaleFactor
- binaryScaleFactor (read only)



Use these keys only if you know how packing works

Note: setting “decimalPrecision” does not repack data but setting “changeDecimalPrecision” does!

Simple packing = discretization



Simple packing

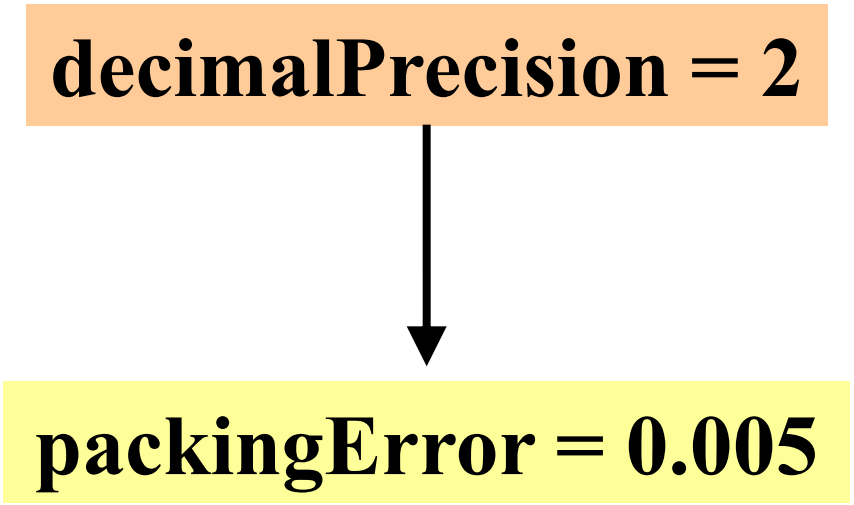
Original value = **Unpacked value** ± **packingError**

Packing error depends on the packing parameters:

bitsPerValue, decimalScaleFactor, binaryScaleFactor, referenceValue

Decimal precision

Decimal precision = decimal digits to be preserved



Simple packing: Example

- Imagine a hypothetical 12-hour 500 hPa geopotential height forecast with values ranging from 5340 to 5460 gpm
- For a decimal precision of 1 we scale all values by 10 so now they will range from 53400 to 54600
- The “decimalScaleFactor” D is chosen such that when the original data is multiplied by 10^D, the integer part of the result will have enough precision to contain all the information
- The “referenceValue” is the minimum (i.e. 53400) . Subtract this from all values to leave non-negative residuals ranging from 0 to 1200
- The calculated bit-length for this range is 11 bits
- All values are now packed into words 11 bits long

Constant fields

- In a constant field all the values are the same
- Repeating the same value N times is very inefficient
- The constant value is the only value stored and the data section is empty.
- Constant fields are very small and they are very precisely encoded
- A constant field can be easily created with

```
grib_set -d 1 in.grib out.grib
```

- In a constant field the packing parameters are not defined (bitsPerValue=0)

Constant fields

WARNING

At this point the packing parameters are not known.

We load a constant field

```
grib_new_from_file(infile,igrib)
```

We set some non-constant values

```
grib_set(igrib,'values',values)
```

We write the field

```
grib_write(igrib,outfile)
```

What packingError can we expect?

In the constant field the packing parameters are not set.
GRIB API doesn't know what precision we require.
A safe choice is made **bitsPerValue=24.**

Constant fields

It is better practice to set decimalPrecision or bitsPerValue before packing the values

```
grib_new_from_file(infile,igrib)
grib_set(igrib,'decimalPrecision',4)
grib_set(igrib,'values',values)
grib_write(igrib,outfile)
```

```
grib_new_from_file(infile,igrib)
grib_set(igrib,'bitsPerValue',16)
grib_set(igrib,'values',values)
grib_write(igrib,outfile)
```

Constants and precision: Practicals

To get the practicals:

```
tar -xvf  ~trx/grib_api/grib_packing.tar
cd grib_packing/constant
```

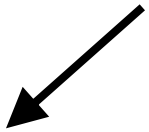
1. You have a GRIB file constant.grib.
2. Set values = 23.26, 42.51, 61.22, 45.95, and print packingError and bitsPerValue
3. Set decimalPrecision=1 and set the same values. Print again packingError and bitsPerValue
4. Compare file sizes and packingErrors.

Bitmap

- The bitmap is an array of binary values. Its size is the number of points in the grid (numberOfPoints)
  - 0 –> value is missing
  - 1 –> value is present
- missingValue = 9999.00

bitmap	values	values
1	2.25	2.12
0	9999	9999
0	9999	9999
1	0.63	0.33
…	…	…

Without using a bitmap



Bitmap

- In order to conserve space, the bitmap is used to efficiently indicate those data points that do appear in the Data Section
- Those data points for which the bit is set to zero will not have a corresponding value in the Data Section

0	0	0	0
0	1	1	0
0	0	1	0

Bitmap section

	2.45	4.67	
		9.11	

Data section

Bitmap: Practicals

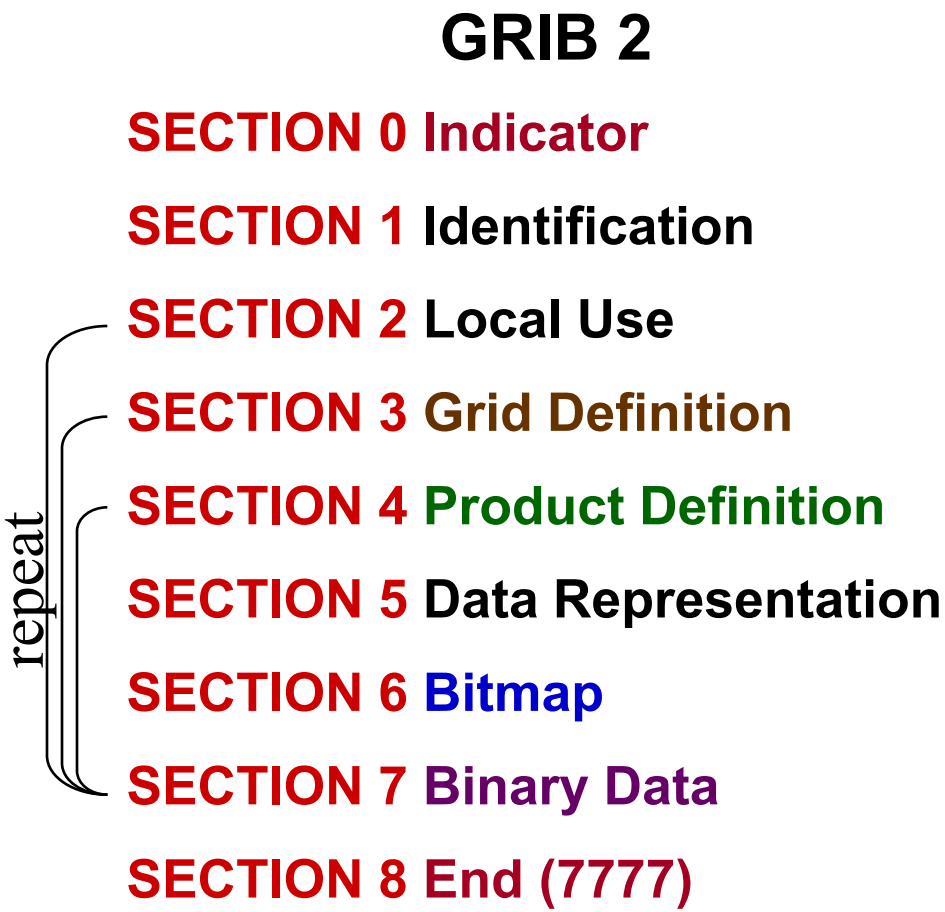
To get the practicals:

tar -xvf ~trx/grib_api/grib_packing.tar

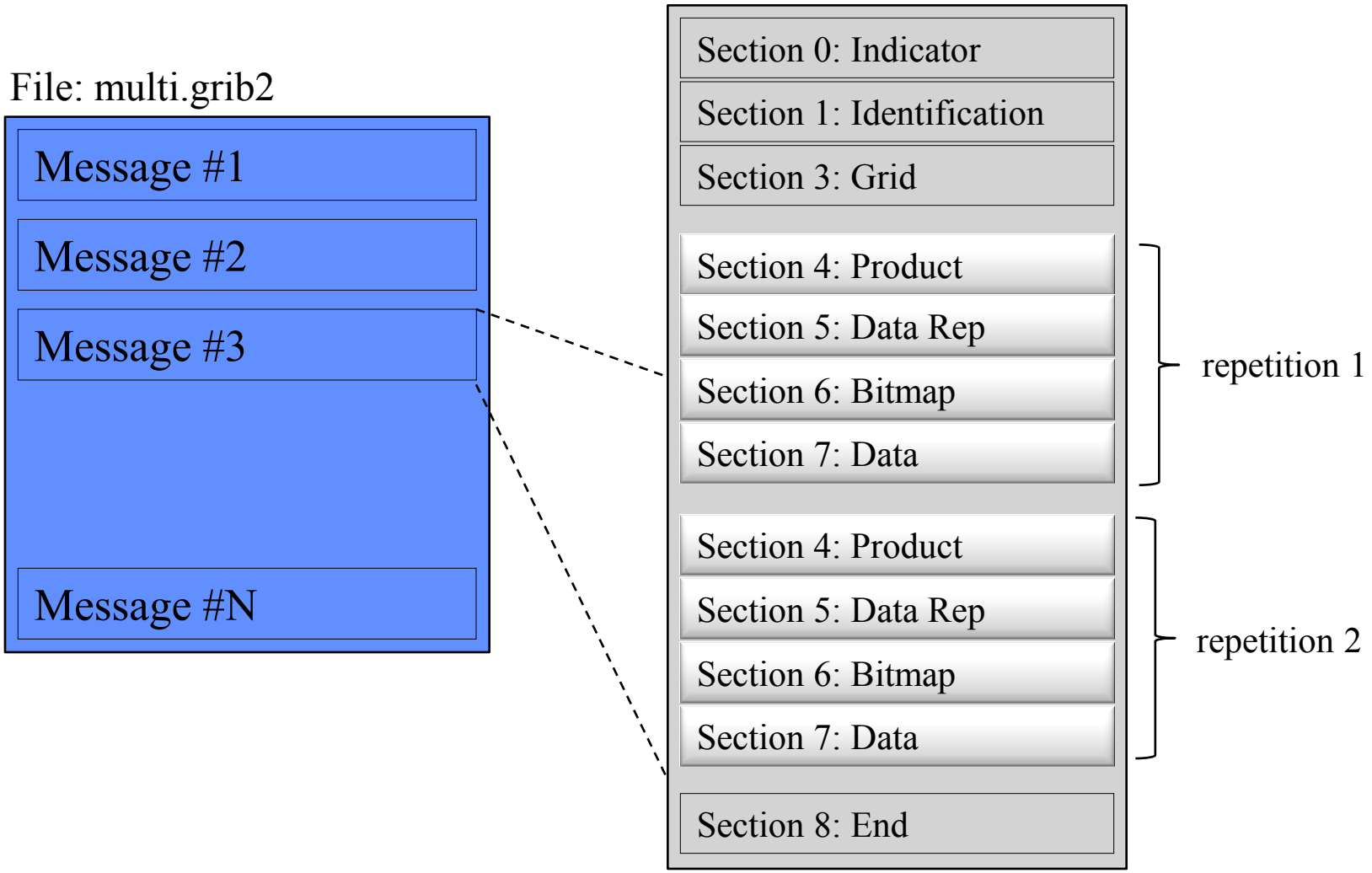
cd grib_packing/bitmap

1. You have a GRIB start.grib with 4 messages. Set
    1. bitsPerValue=8, bitmapPresent=0 in the first message
    2. bitsPerValue=16 , bitmapPresent=0 in the second message
    3. bitsPerValue=24 , bitmapPresent=0 in the third message
    4. bitsPerValue=8, bitmapPresent=1 in the fourth message
2. Set values = 0.2, 0.4, 0.6, 0.7, 9999.
3. Print the values.

Multi field


GRIB 2
SECTION 0 Indicator
SECTION 1 Identification
SECTION 2 Local Use
SECTION 3 Grid Definition
SECTION 4 Product Definition
SECTION 5 Data Representation
SECTION 6 Bitmap
SECTION 7 Binary Data
SECTION 8 End (7777)
repeat

Multi field


File: multi.grib2
Message #1
Message #2
Message #3
Message #N
Section 0: Indicator
Section 1: Identification
Section 3: Grid
Section 4: Product
Section 5: Data Rep
Section 6: Bitmap
Section 7: Data
repetition 1
Section 4: Product
Section 5: Data Rep
Section 6: Bitmap
Section 7: Data
repetition 2
Section 8: End

Multi field: example

- **Consider 500 hPa height field forecasts produced by a numerical model at forecast hours 12 and 24.**

```
Section 0: Indicator Section
Section 1: Identification Section
Section 2: Local Use Section (optional)
Section 3: Grid Definition Section

Section 4: Product Definition Section    (hour = 12)| repetition 1
Section 5: Data Representation Section              |
Section 6: Bit-Map Section                          |
Section 7: Data Section                             |

Section 4: Product Definition Section    (hour = 24)| repetition 2
Section 5: Data Representation Section              |
Section 6: Bit-Map Section                          |
Section 7: Data Section                             |

Section 8: End Section
```

- **Note that since the Grid Definition Section is not repeated, it remains in effect for all forecast hours**

Multi field (example multi.f90)

```
! turn on support for multi field messages
call grib_multi_support_on()

! turn off support for multi field messages
!call grib_multi_support_off()
```

```
call grib_new_from_file(ifile,igrib, iret)
! Loop on all the messages in a file.
do while (iret /= GRIB_END_OF_FILE)
   call grib_new_from_file(ifile,igrib, iret)
end do
```

Multi field (example write_multi.f90)

```
sec=4
do step=0,240,12
  call grib_set(in_gribid,"step",step)
  ! Append in_gribid to multi_gribid
  ! Start from section sec
  call grib_multi_append(in_gribid,sec,multi_gribid)
 enddo
! write messages to a file
call grib_multi_write(multi_gribid,outfile)
```

Multi field: Practicals

To get the practicals:

```
tar -xvf  ~trx/grib_api/grib_multi.tar
```

1. Compile the Fortran program write_multi.f90 and run it to produce a multi field message multi.grib.
2. Using grib_copy, copy multi.grib to copied.grib.
3. Do a grib_count on multi.grib and copied.grib.
4. Compile the Fortran program multi.f90 and run it using multi.grib as input. Turn the multi support on and run it again.
5. Repeat step4 replacing multi.grib with copied.grib as the input file.

Questions ?