ecCodes: Using BUFR Tools Part 2

Computer User Training Course 2018

Shahram Najm

Development Section
Forecast Department

Contents

- Comparing messages
- Copying messages
- Setting header data

bufr_compare – compare BUFR messages

- Use bufr_compare to compare the BUFR 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 bufr_compare
  - switches to a key-based mode to find out which coded keys are different
  - Data section is unpacked
  - fails returning a non-zero exit code

bufr_compare – basic usage

```
bufr_compare [options] bufr_file bufr_file
```

- Options

`-b key,key,…`	All keys in this list are skipped when comparing files
`-H`	Compare message headers only
`-w key[:{s|i|d}]{=|!=}value,…`	Where option
`-f`	Do *not* fail on error
`-v`	Verbose
`...`	

bufr_compare – a simple example

- Check if the BUFR messages in f1.bufr and f2.bufr differ and if so which keys

```
> bufr_compare f1.bufr f2.bufr
== 1 == DIFFERENCE == long
   [#3#channelQualityFlagsForAtovs]: [0] != [1]
== 1 == DIFFERENCE == long
   [#5#channelQualityFlagsForAtovs]: [0] != [2]
> echo $?
1
```

- There are two differences both in message ‘1’
- The keys are integers (type = long)
- The exit code is set to 1 because the comparison failed

bufr_compare – a simple example

- Blacklist the key channelQualityFlagsForAtovs and compare the files again

```
> bufr_compare -b channelQualityFlagsForAtovs f1.bufr
  f2.bufr

> echo $?
0
```

- The exit code is set to 0 because the comparison is successful according to the blacklist. We have blacklisted ALL instances of the key ‘channelQualityFlagsForAtovs’

bufr_compare – a simple example

- Now blacklist just the key #3#channelQualityFlagsForAtovs (rank 3) and compare the files again

```
> bufr_compare -b '#3#channelQualityFlagsForAtovs'
  f1.bufr f2.bufr
== 1 == DIFFERENCE == long
  [#5#channelQualityFlagsForAtovs]: [0] != [2]
> echo $?
1
```

- The exit code is set to 1 because the comparison failed for the 5th instance of channelQualityFlagsForAtovs

bufr_compare – verbose output

- The verbose option shows details of all keys being compared

```
> bufr_compare -v f1.bufr f2.bufr

    comparing edition as long
    comparing masterTableNumber as long
    comparing bufrHeaderCentre as long
    comparing bufrHeaderSubCentre as long
    comparing updateSequenceNumber as string
    comparing dataCategory as long
    comparing internationalDataSubCategory as long
    ...
    comparing #3#channelQualityFlagsForAtovs as long
== 1 == DIFFERENCE == long
   [#3#channelQualityFlagsForAtovs]:[0] != [1]
    comparing #3#channelQualityFlagsForAtovs->units as string
    ...
```

bufr_compare – compare headers only

- To compare only the headers of two BUFR messages use the **-H** option

```
> bufr_compare f1.bufr f2.bufr
== 1 == DIFFERENCE == long [relativeHumidity]: ...
== 1 == DIFFERENCE == double [horizontalVisibility]:
   ...
> bufr_compare -H f1.bufr f2.bufr
```

- The **-H** option does not unpack the data section (so is quicker)

bufr_compare – summary of differences

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

```
> bufr_compare -f f1.bufr f2.bufr
== 1 == DIFFERENCE == long [stationType->percentConfidence]:
   [70] != [71]
== 2 == DIFFERENCE == long [stationType->percentConfidence]:
   [70] != [74]
== 3 == DIFFERENCE == long [stationType->percentConfidence]:
   [70] != [72]

## ERRORS SUMMARY #######
##
## Summary of different key values
## stationType->percentConfidence ( 3 different )
##
## 3 different messages out of 3
...
```

bufr_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`**

bufr_compare – setting the tolerance

- Comparison of the values in two files shows that the local longitude is different with the default absolute error tolerance of zero

```
> bufr_compare f1.bufr f2.bufr
== 1 == DIFFERENCE == double [localLongitude]: [1.5183000000000012506e+02]
   != [1.5183099999999988859e+02]
        absolute diff. = 0.001, relative diff. = 6.58627e-06
        tolerance=0
```

- Set the absolute error tolerance to 0.001 and the comparison is successful

```
> bufr_compare -A 0.001 f1.bufr f2.bufr
```

bufr_compare – setting the tolerance

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

```
> bufr_compare f1.bufr f2.bufr
== 1 == DIFFERENCE == double [localLongitude]: [1.5183000000000012506e+02]
   != [1.5183099999999988859e+02]
        absolute diff. = 0.001, relative diff. = 6.58627e-06
        tolerance=0
```

- Set a relative error of 6.59e-06 as the tolerance for localLongitude

```
> bufr_compare -R localLongitude=6.59e-06 f1.bufr f2.bufr
```

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

Practical: using bufr_compare

```
cd $SCRATCH
cp -r ~trx/ecCodes/2018/bufr_tools_compare ./
cd bufr_tools_compare
```

1. Use bufr_compare to compare the BUFR messages contained in the files syno_1.bufr and syno_2.bufr
   - Which keys does bufr_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 bufr_compare to compare the messages again
   - Are any keys reported as different ? What is the exit code ?
3. Remove the blacklist and try adjusting the tolerances (absolute and relative)

bufr_copy – copy contents of BUFR files

- Use bufr_copy to copy selected messages from BUFR files
- The selection works on header keys (not data section)
- Header key values can be used to specify the output file names
- bufr_copy fails if a key is not found
  - Use the `-f` option to force bufr_copy not to fail on error

bufr_copy – usage

```
bufr_copy [options] bufr_file bufr_file … out_bufr_file
```

- Options

`-p key[:{s|l|d}],…`	Keys to print (only with **`-v`**)
`-w key[:{s|l|d}]{=|!=}value,…`	Where option
`-v`	Verbose
`-f`	Do *not* fail on error
`...`	

bufr_copy – examples

- Copy only the messages whose centre is 80 from a file

```
> bufr_copy -w centre=80 in.bufr out.bufr
```

- Copy only the messages whose centre is not 80 from a file

```
> bufr_copy -w centre!=80 in.bufr out.bufr
```

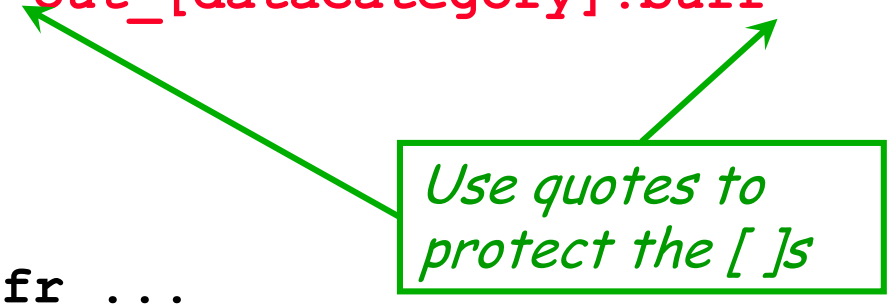
bufr_copy – using key values in output file

- Header key values can be used to specify the output file name

```
> bufr_copy in.bufr "out_[dataCategory].bufr"

> ls out_*

out_0.bufr  out_2.bufr ...
```



- Stop the UNIX shell from interpreting the square brackets!
- This provides a convenient way to filter BUFR messages into separate files

bufr_set – set header key / value pairs

- Not the right tool for making changes in the data section. We will cover that later
- Use bufr_set to
  - Make simple changes to header key/value pairs in the input file(s)
- Each BUFR message is written to the output file
- bufr_set fails when an error occurs
  - e.g. when a key is not found

bufr_set – usage

```
bufr_set [options] bufr_file bufr_file … out_bufr_file
```

- Options

`-s key[:{s|i|d}]=value,…`	List of header key/values to set
`-p key[:{s|i|d}],…`	Keys to print (only with `-v`)
`-w key[:{s|i|d}]{=|!=}value,…`	Where option
`-f`	Do *not* fail on error
`-v`	Verbose
`-S`	Strict
`...`	

bufr_set – examples

- Set the key bufrHeaderCentre in the header and print its value after the change:

```
> bufr_set -v -p bufrHeaderCentre
           -s bufrHeaderCentre=222 in.bufr out.bufr
```

- Note: bufr_set never changes the input file

bufr_set – examples

- Convert a BUFR message from edition 3 to edition 4:

```
> bufr_set -s edition=4 in.bufr out.bufr
```

bufr_set – examples

- Add a local section (section 2) to a BUFR message

```
> bufr_set -s section2Present=1 in.bufr out.bufr
```

- Please note: According to the WMO, the use of local sections in messages intended for non-local or international exchange is strongly discouraged
- The local section is only defined for bufrHeaderCentre=98 i.e. ECMWF
- A local section for other centres requires modification of the definition files
- Adding a local section means the message has extra keys