BUFRextract USER GUIDE

Author: Francis Breame G8ISI
vf0123@btinternet.com

Version 3.0 180911

Please note that BUFR is an all-encompassing format and there are certainly files which this program won't handle. **BUFRextract** is under constant development so things will undoubtedly change. Please email me at vf0123@btinternet.com with any bugs, comments, or suggestions.

Applies to BUFRextract v0.3.1 and subsequent.

1. Introduction

BUFRextract is a program to decode BUFR data files.

It can produce:

- Detailed information about the structure and contents of the file.
- All or selection portions of the decoded data in CSV file format, suitable for feeding into other software for further processing.

It operates entirely from a command line invocation, thereby making it useful for calling from other programs requiring decoded data.

It is a companion to **BUFRdisplay**, which uses the same decode engine but includes extensive GUI facilities for investigating BUFR file contents, as well as displaying the data in a variety of geographical projections. **BUFRdisplay** can produce the same output as **BUFRextract**, but has to be invoked via the GUI.

2. Usage

BUFRextract is invoked as follows:

- Windows: `BUFRextract.exe <BUFR file to be decoded> <options>`
- Linux: `BUFRextract.exe <BUFR file to be decoded> <options>`

The program is written in Perl. However, it is packaged into an executable file including all the needed Perl runtimes. It therefore isn't necessary to install Perl separately, although it won't matter if it is.

The first execution of **BUFRextract** will cache the file's contents (it's actually a self-extracting zip file). Thus the first start will take longer - thereafter the cache is used.

For information, the location of the cache is:

- Windows XP:
  `%SystemDrive%\Documents and Settings\<userid>\Local Settings\Temp\par-<userid>\cache-<instance id>`
- Vista/Windows 7:
  `%SystemDrive%\Users\<userid>\AppData\Local\Temp\par-<userid>\cache-<instance id>`
- Linux:
  `/tmp/par-<userid>/cache-<instance id>`

where `<instance-id>` is a hex number like
`f5c1c961a82f4938342e`e86d6b454bbf3d55e740

Since the instance id changes with each executable released, the cache is automatically checked and flushed of old versions, if necessary, each time a new version of **BUFRextract** is run.

Progress and error messages are output to STDOUT and STDERR. If you are calling the program from another you may wish to sink or otherwise handle this output (e.g. `>nul 2>&1` from a DOS shell).

BUFR uses a set of fixed table files in the decoding process (see Section 3.2.2). **BUFRextract** uses its own formats for the these. If you therefore wish to use a special table of your own, please contact me so that I can convert it into the appropriate format.

3. BUFR file basics

For more details on the BUFR file format, see [2] and [3].

BUFR is a complex format which has grown up over the years. Although the original intentions were good, the following are some of the drawbacks:

- A reluctance to use established or even de facto standards, e.g. using its own form of data compression.
- An obsession with data compression, much less important with today's networks.
- An intention was to make the data self-defining by the use of tables (again using their own methods), somewhat negated by the fact the tables are separate from the data. Whilst the WMO publishes standard tables, these were only, until recently, issued in Word and pdf format. Hence much work was required in transcription before they could be used by en/decoding software, leading to errors.
- As was the intention, user organisations produced their own tables on an ad hoc basis. However, there is no central registration, coordination, or checking organisation, so the required tables may well not be readily available.
- Lack of resiliency. Because everything is reduced to a bit stream, a single transmission error or, worse, table error will render the entire file unusable.
- Many processing rules were added over the years to meet assorted requirements. This has led to a mass of special cases with which decoding software has to (or should) cope.

I am not a fan of BUFR.

However, to continue, the basic structure of a BUFR file is:

3.1. BUFR file

Consists of 1 or more **MESSAGES**.

3.2. MESSAGE

A message is made up of:

3.2.1. HEADERS (BUFR sections 0,1,2)

General information about the message.

3.2.2. DATA DESCRIPTORS (BUFR section 3)

Define the data format.

The type of a descriptor is identified by a number of the form n-nn-nnn, e.g. 0-01-007. This id is used to access a BUFR table, which in turn defines how the data is encoded. This means that new data definitions can be easily (!) added by updating the tables, rather than by changing the BUFR decoder itself. Some descriptors perform operations on the data (e.g. 'delayed replication'), rather than simply describing data values.

3.2.3. SUBSETS (BUFR section 4)

Hold the data as defined by the data descriptors.

There are one or more data subsets. Usually there are quite a few, e.g. a number of observations. In **BUFRextract**, a subset is referred to by a number starting from 1.

Each subset consists of a number of **DATA FIELDS**

3.2.4. DATA FIELDS (BUFR section 4)

Note: this is terminology which I have introduced - it is not part of the BUFR specification.

A DATA FIELD corresponds to a data descriptor which is capable of holding data (not all are).

For a start, we describe the data structure extracted from a **COMPRESSED** BUFR file (more on **UNCOMPRESSED** files later).

In **BUFRextract**, data fields are identified by a number, within each message, starting from 1. I've tried to keep the numbering in line with the ECMWF bufr_toolkit [4] (unfortunately the people who specified BUFR numbered nearly everything - but not all - from 1 rather than 0, which would have been more sensible in computer terms). To confuse the issue further, some fields hold data which is only used internally by the decoder, but were nonetheless retained in the field numbering scheme by ECMWF. They are omitted from the list of fields which provide data externally, so that this list may have gaps in its numbering sequence. The highest field number would therefore be greater than the total number of fields output which carry data (see?). I call the fields which supply "real" data "external fields".

Each subset therefore holds a row of data, of length equal to the number of external fields.

So, conceptually, we have, where:
s = total number of subsets
n = total number of external fields

		FIELDS		
	1	2	...	n
S	1	val 11	val 12	...	val 1n
U				
B	2	val 21	val 12	...	val 2n
S				
E	...			
T				
S	s	val s1	val s2	...	val sn

Note that fields can be empty if the data is not available.

When BUFRextract decodes data into a CSV file, it includes a section which defines each field number in terms of the data name (e.g. Brightness temperature) and data units (e.g. K) - see Section 8.

Values are usually numeric, which will be decoded by BUFRextract into their base value (e.g. degrees, m/s). However. numerics can also refer to code values which require lookup in yet another BUFR table to discover the meaning of the number. For example, descriptor 0-01-007 is "Satellite Identifier". A value of 209, say, if looked up in the code/flag table for 0-01-007 will translate to "NOAA 18". Flags are similar, but the number is treated as a string of binary bits. A "1" in a particular location in the number will yield a translation from the code/flag table. Thus several flags con be incorporated within a single number.

The trouble with field numbering is that, until you've read the BUFR file, you don't know which fields relate to what data. They can change if 'delayed replicators' are used, so they are not necessarily the same in each file type from day-to-day.

However, as usual with BUFR, there is a further complication; BUFR data files can be either **COMPRESSED** or **UNCOMPRESSED**. This mainly affects the internal representation - however, it does impact on field numbering.

For **COMPRESSED** files (which comprise the vast majority, at least on EUMETCast), all subsets within a message are of the same size, i.e. have the same field numbering, as described above. (Theoretically, numbering could change between messages, but I always assume for display purposes that it doesn't, which has been safe so far. YMMV.) Thus only one section in the CSV file is required per message to identify field numbers in terms of their data contents.

However, for **UNCOMPRESSED** files, each subset can be, and usually is, of a different size which means that the field numbering changes for each subset. Hence the field definition section in the CSV file has to be repeated for every single subset.

See section 7 "A note on data field numbering" below for more information.

3.2.5. END MARKER (BUFR section 5)

4. Output

BUFRextract output is in two formats:

4.1. Structure text

The detailed structure of the BUFR file may be listed in text format using the **-ot** option (see Section 6.1). This is particularly useful in looking at the data descriptors to see exactly what is in the file, and what the format of the values is. Additionally, some or all data values can be included in text format (**-a**, **-c**, and **-s** options – see Section 6.2).

4.2. Data CSV

The data values (or selected parts) may be decoded to a CSV file, together with the field definitions as derived from the data descriptor, by using the **-od** option (see Section 6.1). This is a more useful (if bulkier) form for processing by other software. The file format is specified in Section 8.

BUFRextract, at the moment, only outputs numeric values to the CSV file. This means that code/flag values will have to be externally translated via the table to discover their meaning. However, it is possible to output decoded values to the structure text file by using the **-a** and **-c** options (see Section 6.3). Sometime I may fix this, but it isn't really as useful as one might think.

5. Table versions

One of the banes of BUFR is the question of table versions, in that each centre/subcentre can produce local versions of the master tables from the WMO, and there is no central registry of them. These local versions can be difficult to get hold of, and in any case require translation to BUFRextract's internal format.

For straightforward data, the local differences are often not significant, so I put an entry in an internal file which tells the program to use an equivalent master version

Because this happens frequently and therefore requires my issuing a new version of the tables each time, BUFRextract now does this automatically. If it finds that it need a local table version which does not exist within those supplied with the program, it will automatically use an equivalent WMO master version. There is always the danger that the local alterations will be significant, in which case the decode will fail and the full local version will have to be obtained and input. I therefore output a warning to this effect if it does an automatic equivalence.

6. Command line options

Note that there must not be a space between the option and its argument, if any,
e.g. use
-odmyfile.csv
and not
-od myfile.csv

Live with it.

6.1. Output options:

-ot<file> Structure text output file.
To append to an existing file, prepend with >
Default: no text output

-od<file> Data CSV output file.
To append to an existing file, prepend with >
Default: no CSV output

6.2. Options controlling the textual output to the -ot file:

-d<format> Print BUFR data descriptors.
Format = 0 - off
= 1 - full
= 2 - descriptive (fields only)
Default: 0

-e Print data descriptors expansion during initial walk (only useful for debugging).
Default: off

-a Print data values for those subsets selected by **-s** option.
Default: off

-c Print decoded code/flag table values if -a selected, as opposed to numeric values.
Default: off

-s<list> Print values of specified subsets only (numbering from 1).
Only works if **-a** option is also specified.
List is a comma-separated set of values.
This does not affect the CSV file output, where all subset value are always output.
A value can be:

- A number, e.g. `7`
- A range, e.g. `2-8`
- ***** which signifies the last subset - can also be used in a range

e.g. `1,5-7,10-*`
`1-8`
Default: all subsets

6.3. Options controlling the CSV output to the -od file:

-q Include quality info in output (data description operators 2-22 to 2-37)
Unlikely to be of general use as quality info is not (yet) processed by the program.
Default: off - processing stops when quality info is met (so far, all quality info is at the end of the data).

6.4. Options controlling output to both the -ot and -od files:

-m<list> Process specified messages only (numbering from 1).
The list format is as defined for **-s**.
Default: all

6.5. Miscellaneous options:

-x Enable trace output to STDOUT for debugging.

6.6. Examples

- `BUFRextract.exe toz.bfr -d1 -a -s1,2,* -odtoz.csv -ottoz.txt`

  will produce a text listing of all the data descriptors + the values for subsets 1,2,*, as well as a dump of all values to the CSV file.

- `BUFRextract.exe toz.bfr -odtoz.csv`

  will just dump the data.

7. A note on data field numbering

When using BUFR files, extracting the correct parts of the data is one of the most awkward aspects, because of the flexibility of the data descriptors.

For compressed files, the only ways of finding out the numbers of the required data fields are:

- Output the structure and look at the data descriptors. Use options "`-a -s1`" (the `-s1` reduces the amount if output). (Note - using `-d1` to list the data descriptors won't necessarily work because a facility called 'delayed replication' means that sequences of data descriptors can be repeated an arbitrary number of times, which is specified within the data. Thus `-d1` will show that delayed replication is in use, but can't work out the repeats. Use `-a` actually to decode the data.

- Run a full data CSV extract and look at the field definitions.

Then hope that the numbers don't change between successive files, or even messages (in practice they don't seem to, but there is no guarantee).

For uncompressed files, the numbering changes for each subset, let alone message, so the only way is to look at the CSV field definitions for each subset.

If you are automatically looking for data descriptor names in the CSV in order to determine the field numbers, remember that the same data descriptors often repeat, e.g.

```
Field 50 TOVS/ATOVS/AVHRR instrumentation channel number
Field 51 Brightness temperature
Field 52 TOVS/ATOVS/AVHRR instrumentation channel number
Field 52 Brightness temperature
```

In this case, you'll have to be a bit cleverer and look at the values to find the channel number you want, before picking the temperature.

Some field numbers may appear to be missing, as mentioned in Section 3.2.4. These refer to data descriptors which carry data which is only used internally during the decode process, e.g. delayed replicators, and do not produce actual external data.

8. CSV file format

At last what you actually wanted to know...

This file contains the dump if data values. Note that all subsets are always dumped, regardless of the **-s** setting. Only messages selected by **-m** will be dumped however.

The file contains a number of BUFR messages, each with a number of subsets or observations. The number of subsets can vary from message to message.

There are two formats, one for compressed BUFR files (the majority), and one for uncompressed files. In compressed files the field definitions are the same for all subsets within a message. (Although the field definitions for each message within a file are generally the same, they needn't be.)

In uncompressed files, the field definitions can and do change from subset to subset.

8.1. Format 1 - compressed files

In this format, a set of field definitions is given in the header, followed by data rows.

Note that, for backwards compatibility, there is no format identifier.

Each message consists of:

8.1.1. Comments

Start with # and are to be ignored.

8.1.2. Headers

Headers always start with a 0.

`0,"file","<filename>"`
: Source BUFR filename

`0,"message","<message no>"`
: Message number, as specified in the -m option.

`0,"field-<i>-<n>","<name for data field n>","<unit for data field n>","<data descriptor reference>" (1 <= n <= i)`

: where

: i = field number

: Note that some may appear to be missing - these are fields which are used internally during the decode process, as explained in Section 3.2.4

.

: n = position of the data on the data row (see below).

These will always count sequentially, and be the same as i unless there are missing field numbers because they are internal fields.

This is the key bit which gives the correspondence between the data field number, the CSV data row index, the field name & unit text, and the data descriptor (should you wish to do your own BUFR table lookup).

There will be one field cross-reference line for each external field

e.g. `0,"field-32-31","Brightness temperature","K","0-12-163"`

`0,<other headers tbd>`

Any other lines starting with 0 should be ignored.

8.1.3. Data rows

Data rows always start with a number >= 1, representing the subset number.

`<1>,<value 1>,<value 2>,...<value x>` Data row for subset 1
...
`<s>,<value 1>,<value 2>,...<value x>` Data row for subset s

where

s = total number of subsets.

x = total number of external fields

<value n> corresponds to n in the "field-<i>-<n>" header so that the data field information can be extracted from the header. n is the actual position on the data row, numbering from 0 (the subset number will always be position 0).

There will always be s data rows, where s is the total number of subsets.

Data values are blank if undefined.

e.g. `31,224,160,0,621,3,2012,10,8,9,34,58.019,4909,.....`
contains the data for subset 31.

8.1.4. Example file

Shortened in the interests of readability.

```
# Running: BUFRextract v0.2.0 121013, BUFRdecode v0.1.8 121008,
data_BUFRextract v0.2.1 121012
# © Francis Breame 2008-2012
# 2012/10/13 16:55:15
0,"file","W_XX-EUMETSAT-
Darmstadt,SOUNDING+SATELLITE,NPP+ATMS_C_EUMP_20121008093458_v1r0_SDR.
bin"
0,"message","1"
0,"field-1-1","Satellite identifier","Codetable","0-01-007"
0,"field-2-2","Identification of originating/generating
centre","Commoncodetable","0-01-033"
0,"field-3-3","Identification of originating/generating sub-
centre","Commoncodetable","0-01-034"
```

```
0,"field-4-4","Satellite instruments","Codetable","0-02-019"
0,"field-5-5","Satellite classification","Codetable","0-02-020"
0,"field-6-6","Year","A","0-04-001"
0,"field-7-7","Month","Mon","0-04-002"
0,"field-8-8","Day","D","0-04-003"
0,"field-9-9","Hour","H","0-04-004"
0,"field-10-10","Minute","Min","0-04-005"
0,"field-11-11","Second","S","0-04-006"
0,"field-12-12","Orbit number","Numeric","0-05-040"
0,"field-13-13","Scan line number","Numeric","0-05-041"
0,"field-14-14","Field of view number","Numeric","0-05-043"
0,"field-15-15","Granule level quality flags","Flagtable","0-33-079"
0,"field-16-16","Scan level quality flags","Flagtable","0-33-080"
0,"field-17-17","Geolocation quality","Codetable","0-33-078"
0,"field-18-18","Latitude (high accuracy)","Deg","0-05-001"
0,"field-19-19","Longitude (high accuracy)","Deg","0-06-001"
0,"field-20-20","Height or altitude","M","0-07-002"
0,"field-21-21","Satellite zenith angle","Deg","0-07-024"
0,"field-22-22","Bearing or azimuth (degree true)","Deg","0-05-021"
0,"field-23-23","Solar zenith angle","Deg","0-07-025"
0,"field-24-24","Solar azimuth (degree true)","Deg","0-05-022"
0,"field-25-25","Satellite antenna corrections version
number","Numeric","0-25-075"
0,"field-27-26","Channel number","Numeric","0-05-042"
0,"field-28-27","Satellite channel centre frequency","Hz","0-02-153"
0,"field-29-28","Satellite channel band width","Hz","0-02-154"
0,"field-30-29","Antenna polarization","Codetable","0-02-104"
0,"field-31-30","Antenna temperature","K","0-12-066"
0,"field-32-31","Brightness temperature","K","0-12-163"
0,"field-33-32","Noise-equivalent delta temperature while viewing
cold target","K","0-12-158"
.....
1,224,160,0,621,3,2012,10,8,9,34,58.019,4909,1,1,2,0,0,12.52229,-
109.31795,829350,63.86,281.75,142.42,83.69,,1, .....
2,224,160,0,621,3,2012,10,8,9,34,58.019,4909,1,2,2,0,0,12.64277,-
109.9173,829340,62.14,281.62,142.99,83.31,,1,  .....
3,224,160,0,621,3,2012,10,8,9,34,58.019,4909,1,3,2,0,0,12.74974,-
110.45616,829330,60.5,281.5,143.5,82.97,,1,2,  .....
4,224,160,0,621,3,2012,10,8,9,34,58.019,4909,1,4,2,0,0,12.84656,-
110.94949,829320,58.91,281.39,143.96,82.65,,1, .....
5,224,160,0,621,3,2012,10,8,9,34,58.019,4909,1,5,2,0,0,12.93525,-
111.40637,829310,57.35,281.29,144.39,82.34,,1, .....
.....
```

Repeats for 7 messages.

Looking up "Satellite identifier" value 224 in code table 0-01-007 gives "NPP".
Looking up "Originating centre" value 160 in code table 0-01-033 gives "US NOAA/NESDIS".
Looking up "Satellite instruments" value 621 in code table 0-02-019 gives "ATMS".

8.2. Format 2 - uncompressed files

In this format, each data row contains its own field definition.

Each subset consists of:

8.2.1. Comments

Start with # and are to be ignored.

8.2.2. Headers

Headers always start with a 0.

`0,"file","<filename>"`
Source BUFR filename

`0,"format","2"`
Indicates an uncompressed file CSV format

`0,"message","<mn>","<m>"(1 <= mn <= m)`
Message number (mn) and total messages (m) in this file.

`0,"subset","<sn>","<s>" (1 <= sn <= s)`
Subset number (sn) and total subsets (s) in this message.

`0,<other headers tbd>`
Any other lines starting with 0 should be ignored.

8.2.3. Data rows

```
<index>,"<name for data field>","<unit for field>","<data descriptor
for field>",<field data value>
```

Index just numbers sequentially from 1 for the total number of data fields in this subset.
Data values are blank if undefined.

e.g. `64,"Total precipitation/total water equivalent","kg/m**2","0-13-011",9`

8.2.4. Example file

Shortened in the interests of readability.

```
# Running: BUFRextract v0.2.0 121013, BUFRdecode v0.1.8 121008,
data_BUFRextract v0.2.1 121012
# © Francis Breame 2008-2012
# 2012/10/13 16:58:01
0,"file","Z__C_EDZW_20121008080910_bda01,synop_bufr_999999_999999__MW
_608.bin"
```

```
0,"format","2"
0,"message","1","11"
0,"subset","1","3"
1,"WMO block number","Numeric","0-01-001",1
2,"WMO station number","Numeric","0-01-002",199
3,"Station or site name","CCITTIA5","0-01-015",SIHCCAJAVRI
4,"Type of station","Code table","0-02-001",0
5,"Year","Year","0-04-001",2012
6,"Month","Month","0-04-002",10
7,"Day","Day","0-04-003",8
8,"Hour","Hour","0-04-004",8
9,"Minute","Minute","0-04-005",0
10,"Latitude (high accuracy)","Degree","0-05-001",68.7553
11,"Longitude (high accuracy)","Degree","0-06-001",23.5387
12,"Height of station ground above mean sea level (see note
3)","M","0-07-030",382
13,"Height of barometer above mean sea level (see note 4)","M","0-07-
031",383
14,"Pressure","Pa","0-10-004",95740
15,"Pressure reduced to mean sea level","Pa","0-10-051",100320
.....
0,"file","Z__C_EDZW_20121008080910_bda01,synop_bufr_999999_999999__MW
_608.bin"
0,"format","2"
0,"message","1","11"
0,"subset","2","3"
1,"WMO block number","Numeric","0-01-001",1
2,"WMO station number","Numeric","0-01-002",262
3,"Station or site name","CCITTIA5","0-01-015",NORDOYAN LH
4,"Type of station","Code table","0-02-001",0
5,"Year","Year","0-04-001",2012
6,"Month","Month","0-04-002",10
7,"Day","Day","0-04-003",8
8,"Hour","Hour","0-04-004",8
9,"Minute","Minute","0-04-005",0
10,"Latitude (high accuracy)","Degree","0-05-001",64.7977
11,"Longitude (high accuracy)","Degree","0-06-001",10.5493
12,"Height of station ground above mean sea level (see note
3)","M","0-07-030",33
13,"Height of barometer above mean sea level (see note 4)","M","0-07-
031",
14,"Pressure","Pa","0-10-004",
15,"Pressure reduced to mean sea level","Pa","0-10-051",
.....
```

Repeats for 3 subsets, and then for 11 messages with however many subsets each message contains.

Looking up "Type of station" value 0 in code table 0-02-001 gives "Automatic".

9. Legal Stuff

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

For details of the GNU General Public License see
http://perldoc.perl.org/perlgpl.html
or write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

10. Links and References

[1] My www site
http://www.elnath.org.uk/

[2] WMO Documentation, especially 306 Manual On Codes, vol 1.2
http://www.wmo.int/pages/themes/wmoprod/manuals.html

[3] A Guide To The WMO Code Form FM 94 BUFR - very useful if you want to write a decoder
dss.ucar.edu/docs/formats/bufr/bufr.pdf

[4] European Centre for Medium-Range Weather Forecasts (ECMWF) - Unix BUFR decoding software
http://www.ecmwf.int/products/data/software/bufr.html