BUFR format and graphical format for polar volume data submitted to and composites produced by the OPERA Odyssey

B. Urban

Meteo France

12 March 2012

Release notes

12 March 2012

Increased length for “how” string values.

18 November 2011

Adding descriptors for coding flexibly ODIM 'how' attribute; implies creating local BUFR tables 9; upgrade to bufr-opera-mf-1.19 tools

24 February 2011

Adding mandatory list of radar used in composites; various clarifications; upgrade to bufr-opera-mf-1.18 tools

9 June 2010

First official release under name WD_2010_10_ODC_Polar_and_Composite_Data_in_BUFR

Design choices

Main design choice was to reuse existing BUFR descriptors as much as technically possible, while staying within the framework of the ODIM specifications. Best understanding of what follows requires that you have a look at that ODIM specification: http://www.knmi.nl/opera/opera3/OPERA_2008_18_WP2.1b_ODIM_UML.pdf

Compression

The amount of data to be coded will increase hugely to handle polar volume and a very large compositing domain, so an efficient compression method is needed. The way it was done so far was twofold:

1. There is a first slice level encoding step for reflectivity, which is alas lossy
2. Then there is RLE encoding of the resulting integer array

The method was successful so far for reflectivity data, but appears cumbersome to use for new parameters, such as rainfall accumulation.

It is here proposed to handle directly the array of raw values (be it a 2D array of nbins*nrays polar measurements or a cartesian array in some geographic projection), but to compress it in the message using standard tools like zlib. This will be valid for reflectivity, rainfall accumulation or radial wind. The raw values are floating point IEEE 64 bits, and the above 2D arrays will be actually 1D arrays, organized in chronological order for polar data and as video scanning order for cartesian data.

New BUFR descriptors and new tables for mandatory ODIM products

There are 2 set of local tables defined below:

1. Release 8 is enough for coding/decoding for the Odyssey data center of OPERA 3
2. Release 9 adds flexibility to code data which is not mandatory for the Odyssey data center of OPERA 3, but which is expected to be so in the near future

BUFR templates and tables content for mandatory ODIM products (table 8)

The italic comments refers to the HDF5 equivalent element. The template for mandatory polar volume is:

ODIM template for mandatory polar volume			
3	21	204	ODIM additional station identifiers (*/what/source*, except WMO type coded below)
3	1	31	WMO station number, nominal date/time and station position (*/what/date, /what/time, /where/lon, /where/lat, /where/height*)
3	21	203	ODIM polar volume mandatory content

The template for mandatory composite is given below.

ODIM template for mandatory composites products			
3	1	11	Date (year, month, day) (*/what/date*)
3	1	13	Time (hour, minute, second) (*/what/time*)
3	21	204	ODIM compositing center identifiers (*/what/source*); it should be at least ORG:247 for Odyssey products
0	29	205	Geographic projection as PROJ initialization string */where/projdef*)
0	5	33	Pixel size on horizontal between columns *(/where/xscale)*
0	6	33	Pixel size on horizontal between rows *(/where/yscale)*
2	1	129	We need more bits to code large products
0	30	21	Number of pixels per row *(/where/xsize)*
0	30	22	Number of pixels per column *(/where/ysize)*
2	1	0	
3	1	21	Latitude/longitude of top left corner of the image *(/where/UL_lon, /where/UL_lat)*
3	1	21	Latitude/longitude of top right corner of the image *(/where/UR_lon, /where/UR_lat)*
3	1	21	Latitude/longitude of bottom right corner of the image *(/where/LR_lon, /where/LR_lat)*
3	1	21	Latitude/longitude of bottom left corner of the image *(/where/LL_lon, /where/LL_lat)*
3	21	8	Z-R law used *(only for rainfall intensity and accumulation)*
3	21	204	List of radars contributing to the composite as their node name (see table 3 of ODIM 2.1 specification); type of station identifier is always NOD
1	4	0	Delayed replication of 4 descriptors
0	31	1	Number of parameters

ODIM template for mandatory composites products			
3	21	205	Time period *(/dataset?/what/startdate,/dataset?/what/starttime, /dataset?/what/enddate and /dataset?/what/endtime)*
0	30	31	Picture type *(/dataset?/what/product)*
0	30	196	Type of product *(/dataset?/what/quantity)*
3	21	206	ODIM compressed array

The new element descriptors and sequence descriptors introduced are summarized below, they can simply be added to our current tables 6 to create a full working set of tables for the specification described here.

BUFR TABLES B
(Version 8 - originating center 247)

TABLE REFERENCE			**TABLE ELEMENT NAME**	**BUFR**			
				UNIT	**SCALE**	**REFERENCE VALUE**	**DATA WIDTH (Bits)**
f	*x*	*y*		**SCALE**			
0	1	192	Type of station identifier	CCITT IA5	0	0	24
0	1	193	Station identifier	CCITT IA5	0	0	128
0	30	197	Compression method	Code-Table	0	0	8
0	30	198	Byte element of a compressed array	Numeric	0	0	8
0	29	205	Geographic projection as PROJ initialization string	CCITT IA5	0	0	800

CODE AND FLAG TABLES ASSOCIATED WITH BUFR TABLE B
(version 8 – originating center 247)

0 30 197

Type of compression

Code	
0	zlib compression
1-254	Reserved
255	Missing value

BUFR TABLES D

(Version 8 - originating center 247)

						ODIM polar volume mandatory content
3	21	203	1	12	0	Delayed replication of 12 descriptors
			0	31	1	Number of scans
			3	21	205	Time period *(/dataset?/what/startdate, /dataset?/what/starttime, /dataset?/what/enddate and /dataset?/what/endtime)*
			0	30	196	Type of product *(/dataset?/what/product)*
			0	2	135	Antenna elevation *(/dataset?/where/elangle)*
			0	30	194	Number of bins along the radial *(/dataset?/where/nbins)*
			0	21	201	Range-bin size *(/dataset?/where/rscale)*
			0	21	203	Range-bin offset[1] *(/dataset?/where/rstart)*
			0	30	195	Number of azimuths *(/dataset?/where/nrays)*
			0	2	134	Antenna beam azimuth (of first ray) *(/dataset?/where/a1gate)*
			1	2	0	Delayed replication of 2 descriptors
			0	31	1	Number of parameters
			0	30	196	Type of product *(/dataset?/data?/what/quantity)*
			3	21	206	ODIM compressed array
						ODIM additional station identifier
3	21	204	1	2	0	Delayed replication of 2 descriptors
			0	31	1	Delayed descriptor replication factor
			0	1	192	Type of station identifier *(RAD, ORG, PLC, NOD, CTY or CMT from ODIM 2.1 specification table 3)*
			0	1	193	Station identifier
						ODIM times stamps
3	21	205	1	2	2	Replication of 2 descriptors (start time stamp and end time stamp)
			3	1	11	Date (year, month, day)
			3	1	13	Time (hour, minute, second)
						ODIM compressed array
3	21	206	0	30	197	Compression method
			1	3	0	Compressed array of IEEE double floats
			0	31	2	Number of chunks
			1	1	0	Delayed replication of 1 descriptor
			0	31	2	Size of chunk

1 this is here in meters, contrary to ODIM specification which requires km

						ODIM polar volume mandatory content
			0	30	198	Byte element of a compressed array

Remark: notice the use of 2 levels of delayed replication in the compressed array, as 0 31 2 is limited to 16 bits, which is by far not enough.

Link with ODIM specification when using tables 8

These are additional coding requirements to fill properly descriptor 0 30 196 and 0 30 31.

1. Polar data: the type of product 0 30 196 is used as 'product' field in the ODIM and for the 'quantity' field of a complete scan at a given elevation. Corresponding values are:
   - value 90 corresponds to 'SCAN' for the 'product' field in the ODIM
   - value 0 corresponds to 'DBZH' for the 'quantity' field in ODIM
   - value 40 corresponds to 'VRAD' for the 'quantity' field in ODIM
2. Composite products: type of product 0 30 196 is used for the 'quantity' ODIM field; values are:
   - value 4 is for reflectivity ('DBZH' in ODIM specification)
   - value 20 is for rainfall intensity ('RATE' in ODIM specification)
   - value 21 is for rainfall accumulation ('ACRR' in ODIM specification)
   - value 255 (missing value) is for quality ('QIND' in ODIM specification)
3. Composite products: picture type 0 30 31 is used for 'product' ODIM field:
   - value 1 is for composite ('COMP' in ODIM specification)

BUFR templates and tables content for extended ODIM products (tables 9)

The red color will be used to indicate differences from tables 8. The template for mandatory polar volume is:

ODIM template for mandatory polar volume with extensions			
3	21	204	ODIM additional station identifiers (*/what/source*, except WMO type coded below)
3	1	31	WMO station number, nominal date/time and station position (*/what/date, /what/time, /where/lon, /where/lat, /where/height*)
3	21	207	ODIM polar volume mandatory content with extensions

The template for mandatory composite is given below. Notice that the Z-R law parameters can now be included via descriptor 3 21 210.

ODIM template for mandatory composites products with extensions			
3	21	208	ODIM composites mandatory content with extensions

BUFR TABLES B

(Version 9 - originating center 247)

TABLE REFERENCE			TABLE ELEMENT NAME	BUFR			
f	*x*	*y*		UNIT SCALE	SCALE	REFERENCE VALUE	DATA WIDTH (Bits)
0	30	199	ODIM product (table 14)	CCITT IA5	0	0	48
0	30	200	ODIM quantity (table 16)	CCITT IA5	0	0	48
0	30	201	ODIM how attribute name	CCITT IA5	0	0	128
0	30	202	ODIM how attribute string value	CCITT IA5	0	0	128
0	30	203	ODIM how attribute double value	CCITT IA5	0	0	64

Notices:

1. 0 30 199 and 0 30 200 will allow for 6 characters field length and 0 30 201 and 0 30 202 will allow for 16 characters field length.
2. The 8 bytes of 0 30 203 are to be interpreted as an IEEE double float, ordered most significant bit first (sign bit, then exponent bits, and finally mantissa bits); program 'decbufr' won't print out correctly 0 30 203. The reason for this was to avoid having a descriptor length greater than 32 bits, which most BUFR software won't grok, OPERA 3.1 included.

BUFR TABLES D

(Version 9 - originating center 247)

						ODIM polar volume mandatory content with extensions (evolution of 3 21 203)
3	21	207	3	21	209	ODIM how attributes set */how*
			1	14	0	Delayed replication of 14 descriptors
			0	31	1	Number of scans
			3	21	209	ODIM how attributes set */dataset?/how*

			3	21	205	Time period *(/dataset?/what/startdate, /dataset?/what/starttime, /dataset?/what/enddate and /dataset?/what/endtime)*
			0	30	199	ODIM product *(/dataset?/what/product)*
			0	2	135	Antenna elevation *(/dataset?/where/elangle)*
			0	30	194	Number of bins along the radial *(/dataset?/where/nbins)*
			0	21	201	Range-bin size *(/dataset?/where/rscale)*
			0	21	203	Range-bin offset[2] *(/dataset?/where/rstart)*
			0	30	195	Number of azimuths *(/dataset?/where/nrays)*
			0	2	134	Antenna beam azimuth (of first ray) *(/dataset?/where/a1gate)*
			1	3	0	Delayed replication of 3 descriptors
			0	31	1	Number of parameters
			3	21	209	ODIM how attributes set */dataset?/data?/how*
			0	30	200	ODIM quantity *(/dataset?/data?/what/quantity)*
			3	21	206	ODIM compressed array
						ODIM composites mandatory content with extensions (evolution of composite template with tables 8)
3	21	208	3	21	209	ODIM how attributes set */how*
			3	1	11	Date (year, month, day) (*/what/date*)
			3	1	13	Time (hour, minute, second) (*/what/time*)
			3	21	204	ODIM compositing center identifiers (*/what/source*); it should be at least ORG:247 for Odyssey products
			0	29	205	Geographic projection as PROJ initialization string */where/projdef*)
			0	5	33	Pixel size on horizontal between columns *(/where/xscale)*
			0	6	33	Pixel size on horizontal between rows *(/where/yscale)*
			2	1	129	We need more bits to code large products
			0	30	21	Number of pixels per row *(/where/xsize)*
			0	30	22	Number of pixels per column *(/where/ysize)*
			2	1	0	
			3	1	21	Latitude/longitude of top left corner of the image *(/where/UL_lon, /where/UL_lat)*
			3	1	21	Latitude/longitude of top right corner of the image *(/where/UR_lon, /where/UR_lat)*
			3	1	21	Latitude/longitude of bottom right corner of the image *(/where/LR_lon, /where/LR_lat)*
			3	1	21	Latitude/longitude of bottom left corner of the image *(/where/LL_lon, /where/LL_lat)*

2 this is here in meters, contrary to ODIM specification which requires km

			3	21	204	List of radars contributing to the composite as their node name (see table 3 of ODIM 2.1 specification); type of station identifier is always NOD
			1	5	0	Delayed replication of 5 descriptors
			0	31	1	Number of parameters
			3	21	209	ODIM how attributes set */dataset?/how*
			3	21	205	Time period *(/dataset?/what/startdate,/dataset?/what/starttime, /dataset?/what/enddate and /dataset?/what/endtime)*
			0	30	199	ODIM product (table 14)
			0	30	200	ODIM quantity (table 16)
			3	21	206	ODIM compressed array
						ODIM how attributes set
3	21	209	1	2	0	Delayed replication of 2 descriptors
			0	31	1	Delayed descriptor replication factor
			0	30	201	ODIM how attribute name
			0	30	202	ODIM how attribute string value
			1	2	0	Delayed replication of 2 descriptors
			0	31	1	Delayed descriptor replication factor
			0	30	201	ODIM how attribute name
			0	30	203	ODIM how attribute double value

Other coding recommendations

Section 1 encoding

The following values are recommended:

	Polar data	Reflectivity composite	Rainfall intensity composite	Rainfall accumulation composite
BUFR edition	4	4	4	4
Originating center	247	247	247	247
Originating sub-center	Country code[3]	74 if created by UKMO, 85 if created by Meteo	74 if created by UKMO, 85 if created by	74 if created by UKMO, 85 if created by

3 which can be found in table C-11 of that document:

ftp://www.wmo.int/Documents/MediaPublic/Publications/CodesManual_WMO_no_306/WMO306_Vol_I.2_2010_en.pdf

		France	Meteo France	Meteo France
Data category	6	6	6	6
Local data sub-category	Free	Free	Free	Free
International data sub-category	0 if only reflectivity, 2 if other data present (like wind)	0	2	2

Other coding remarks

1. The new compression scheme being quite good (see tests below), it didn't seem there is a need to have 'gain' and 'offset' values different from 1 and 0 in the ODIM BUFR version. Nevertheless, if for instance there is input data in a reflectivity composite with a 0.5 dBZ resolution at most, it doesn't make sense to keep a higher resolution for the composite values, which should then be rounded to the nearest 0.5 dBZ, enhancing compression performance substancially. Same remark can be made for quality QIND, which is a float value between 0 and 1; it makes currently little sense here to have values finer than 0.05.
2. 'nodata' is in the BUFR version the IEEE value called (in C language) DBL_MAX; ditto for 'undetect', taken as -DBL_MAX.
3. Bits of the IEEE float values used in the compressed array have to be ordered most significant bit first (sign bit, then exponent bits, and finally mantissa bits) before and after compression; this is coherent with how BUFR orders bits.

Tests

Polar data

The best way to test the validity of the proposed coding is to create samples of polar volume data. Better, successful conversion without loss between the HDF5 and BUFR version will prove that the formats specifications are complete and inter-operable. This was done for some Meteo France polar data of some medium rainy case (three scans, one reflectivity parameter); the data was 'randomized' so as to fake 0.5 dBZ resolution; even the large slice of undetected measurements below 8dBZ was handled that way, which will stress the compression algorithm performance.

There are two programs.

1. 'bufr2hdf5' converts from the above BUFR polar format to HDF5
2. 'hdf2bufr' converts HDF5 to the above BUFR polar format

The important property of idem-potence is verified by applying 'hdf2bufr' to the result of 'bufr2hdf5'. Performances for size and speed:

	BUFR (zlib compression level 6)	HDF5 (zlib compression level 6)
Size for 3 scans 720 rays * 256 bins	579540 bytes	640314 bytes
Time for reading in memory (Intel Xeon 3.20GHz)	real 0m0.446s user 0m0.380s sys 0m0.004s	real 0m0.205s user 0m0.116s sys 0m0.004s

Composites products

The same strategy (successful conversion without loss between the HDF5 and BUFR) was applied on a Meteo France european domain reflectivity composite; it was originally coded with slice levels, so randomization to 0.5 dBZ resolution was also applied. The associated quality flag array was also randomly simulated, at a resolution of 0.05[4]. Image size depends heavily on the amount of rainy pixels; the picture at the end of the document is the result of the processing of the sample composite used, and gives an idea of that amount.

There are two programs.

3. 'comp2hdf5' converts from the above BUFR composite format to HDF5
4. 'hdf2comp' converts HDF5 to BUFR composite format

The important property of idem-potence is verified by applying 'hdf2comp' to the result of 'comp2hdf5'. Performances for size and speed:

	BUFR (zlib compression level 6)	HDF5 (zlib compression level 6)
Size for 1 reflectivity parameter and 1 associated quality flag; 2200 rows * 1700 columns; 2km resolution	2065884 bytes	2184845 bytes
Time for reading in memory (Intel Xeon 3.20GHz)	real 0m2.508s user 0m2.344s sys 0m0.048s	real 0m1.357s user 0m1.120s sys 0m0.112s

Comments about the tests

1. The "data randomizing" gives a worst case scenario for files size; notice that even without that, due to the lack of level slicing in the input polar data and output composites, the Odyssey products will be large
2. HDF5 has been designed to handle huge files, probably at the expense of miss-handling smaller ones
3. BUFR obviously loses some time manipulating bits instead of bytes or machine words, but there is a win in compression ratio

4 ODIM quality flags take real values between 0 and 1

4. Endianness compatibility in the decoded compressed array is to be verified; ditto for the fact that IEEE is really encoded in the BUFR and HDF5 files
5. 'bufr2hdf5' and 'hdf2bufr' are very close to generic converters between BUFR and HDF5 for polar volume data; ditto for 'hdf2comp' and 'comp2hdf5' for composite products

How to run the tests yourself

Tests are in file 'bufr-opera-mf-1.19.tar.gz', to be downloaded at the OPERA ftp site. The file includes BUFR library version 3.1beta. Requirements: Unix like system, with zlib, PROJ and HDF5 development libraries installed. Commands to run:

```
tar xzf bufr-opera-mf-1.19.tar.gz
cd  bufr-opera-mf-1.19
# You may have to prefix the following with values for CFLAGS and LDFLAGS
# if some libraries are not at the standard location
./configure
make
cd tests
# don't run 'make check' in the top directory! see last section for explanations
make check
# All comments should read "PASS", not "FAIL"
make odim # to build the ODIM BUFR file 'odim-polar.bfr' from Meteo France data
make bufr2hdf # to build the equivalent HDF5 file 'odim-polar.hdf5'
make hdf2bfr # to build back 'odim-polar.bfr' called 'odim-polar.bfr2'
cmp  odim-polar.bfr odim-polar.bfr2 # no error should be reported
make compo # to build the ODIM BUFR file 'odim-compo.bfr' from Meteo France data
make comp2hdf # to build the equivalent HDF5 file 'odim-compo.hdf5'
make hdf2com # to build back 'odim-compo.bfr' called 'odim-compo.bfr2'
cmp  odim-compo.bfr odim-compo.bfr2 # no error should be reported
```

Low level message coding recommendations

It is assumed you will use a C program to code your message. Your coding program has to fill with polar data an '`odim_polar_t`' structure. That structure is described extensively in file '`bufr-opera-mf-1.19/tests/polar.h`'(see above); reading that file along with the ODIM specification should be enough to allow you to fill the structure.

When the structure is filled, you should call a C function such as '`recode_opera_local()`' (see file '`bufr-opera-mf-1.19/tests/hdf2bufr.c`') to create the final message.

Low level composite decoding recommendations

Still assuming you use a C program, you have to use a main function such as the one at the end of file '`bufr-opera-mf-1.19/tests/comp2hdf5.c`', but which will have the part starting with '`write_hdf5()`' replaced with your own application. At that point in the program '`comp2hdf5`', you will have the global variable '`odim_data`' of type '`odim_comp_t`' filled with all composite data. That '`odim_comp_t`' structure is described in details in file '`bufr-opera-mf-1.19/tests/compo.h`', and mimics closely the composite ODIM specification.

Sweets

Directory `'bufr-opera-mf-1.19/view'` contains converters from BUFR to PNG graphical files, similar to the one available in the Graz compositing software. The programs are built when running `'make check'` in that directory. This will obviously work only if you have the PNG library development files; but you will also need to install the modified PROJ library which has the 'polar' projection; see file `'Software/odim/proj-4.4.9polar.tar.gz'` at the OPERA ftp site. Currently, only reflectivity is handled, and the color table is built-in, but this can be adapted easily.

Raw visualization

To convert the first elevation of reflectivity included in a polar ODIM file in BUFR to a PNG image, with range as the x-axis and azimuth as the y-axis:

```
./bufr2view -d ../tables ../tests/odim-polar.bfr polar.png
```

To convert an ODIM reflectivity composite file in BUFR to PNG:

```
./comp2view -d ../tables ../tests/odim-compo.bfr compo.png
```

Elaborated visualization

To convert all reflectivity elevations of a polar ODIM file in BUFR to PNG, in the azimuthal equidistant projection:

```
./bufr2view-cart -d ../tables ../tests/odim-polar.bfr local_radar
```

The last argument is the root for the file names of the generated PNG images, which will have elevation information and the suffix `'.png'` appended.

Super elaborated visualization

To convert all reflectivity elevations of a polar ODIM file in BUFR to PNG, in the Google Maps or Google Earth projection:

```
./bufr2view-cart -k -d ../tables ../tests/odim-polar.bfr radar
```

Files have the same naming convention as for the azimuthal equidistant case. It is then easy to include these images as overlays in Google Maps or Google Earth, as they have already transparency built-in, and a KML wrapper with the right boundaries is created along. The name of the wrapper is the last argument with suffix `'.kml'` added.

To convert an ODIM reflectivity composite file in BUFR to PNG, in the Google Maps or Google Earth projection:

```
./comp2view -k -d ../tables ../tests/odim-compo.bfr composite
```

As above, the image has already transparency built-in and is ready to use as an overlay in Google Maps or Google Earth, with a KML wrapper created along (file name wrapper is last argument with suffix `'.kml'` added).

Graphical format proposal

It is proposed that PNG is the graphical format to be used for the Odyssey output.

Graphical output will mainly be used to spot visually interesting weather patterns in relation to surface places. So visualizing some surface background along with radar data is the way to go. Most OPERA members have adequate tools to do that, reading directly BUFR or HDF5 files; but external users probably won't have them.

One of the cheapest way to offer a usable service for them is to use tools like Google Maps (which can be used purely as a Web app) or Google Earth to visualize the produced graphical format. The only request is to have some transparency in the image and to reproject it to latitude/longitude (respectively from polar or Lambert projection). Programs `'bufr2view-cart'` and `'comp2view'` with option `-k` create exactly what is needed for that, so it is suggested they serve as the basis for graphical output production.

A sample of how it could look like is given below: