|  |  |  |
| --- | --- | --- |
| WORLD METEOROLOGICAL ORGANIZATION  COMMISSION FOR BASIC SYSTEMS  -----------------------------  THIRD MEETING OF  INTER-PROGRAMME EXPERT TEAM ON DATA REPRESENTATION MAINTENANCE AND MONITORING  BEIJING, CHINA, 20 - 24 JULY 2015 |  | IPET-DRMM-III / Doc.3.2(5)  (29.06.2015)  -------------------------    ITEM 3.2    ENGLISH ONLY |

**Encoding elements with a large range and limited precision in BUFR**

*Submitted by Charles Sanders and Weiqing Qu (Australia)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Summary and Purpose of Document**

This document proposes some solutions for handling elements with large range and limited precision in BUFR

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACTION PROPOSED**

The meeting is requested to review and discuss the proposed solutions.

# Background

Attempts to validate the encoding of wave observations in BUFR have revealed that BUFR does not have adequate facilities for encoding elements which have a large range and limited relative precision. Examples of such elements include, amongst many others, water wave spectral energy densities, pressure in some vertical profiles, concentrations of some atmospheric constituents, and air density.

Several possible solutions have been identified, including

1. Use a scale and magnitude representation to allow the required precision for small values without having an excessively large data width.
2. Use existing BUFR operators
3. Define a new “Delayed change of scale” operator.
4. Encode the logarithm of the element’s value instead of the value.
5. Use a standard IEEE floating point format.
6. Use a “floating point” format, with separate sign, exponent and significand sub-fields.

# General discussion

The problem arises because obtaining the required range for large values and the required precision for small values results in very large data widths, wasting a lot of space. In addition, if the software (as much does) uses IEEE 32 bit floating point for intermediate values, problems can arise for data widths in excess of 24. Other software, using 32 bit integers for the bit manipulation involved in the extraction of the bit fields is likely to fail for data widths in excess of 32.

Option 1 (scale-magnitude) has been used for some elements already. It is possible that some software could apply the scaling internally, and other software could use the scale and scaled magnitude representation, resulting in confusion as to whether the scaling has been applied.

Option 2 (using existing operators) is almost certainly not workable in reality. The problem is not the existence of some subsets with much larger values for an element than other subsets, but the wide range of values for an element within a single repetition in a single subset. Unless the replications are removed and replaced with a long sequence of repeated descriptors, the BUFR operators will only allow a single set of precision and data width changes for each repetition Thus, covering the wide range still requires excessively large data widths. I will not discuss this further.

Option 3, a delayed change of scale operator, is workable, but is a big change from previous practice. It also introduces a new operator, which would require software changes and would almost certainly require a new edition of BUFR.

Option 4, logarithmic encoding, is used for some elements already, such as 0 13 099 through 0 13 101 (cloud particle density, area and volume) 0 15 011 (electron density) and some others. Simple logarithmic encoding cannot be used for zero, but this can be handled by either having a special value for zero, or by adding a small offset before taking the logarithm.

Option 5, an IEEE floating point encoding, will normally easily handle the required ranges and precisions, but will (due the limited choice of data widths) usually waste considerable space This option would almost certainly require a new edition.

Option 6, a custom floating point format, requires an extra column or columns in BUFR table B, and also changes to the encoding and decoding software. This would almost certainly require a new edition.

# An example of encoding data with each option

Consider an element that has the following requirements:

Name: Arbitrary element

Unit: Some unit

Range: 0 to 105

Absolute precision for small values: 10-7

Relative precision for large values: 3 decimal digits

In conventional BUFR representation, this would require a scale of 7, a reference value of 0, and the encoded integers would range from 0 to 1012 (105\*107). The required data width would be 40 bits, with the actual representable range being 0 to (approximately) 109951.16. Assuming 0 aa bbb was the element descriptor assigned, the table B entry would be something like

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Scale | Reference | Data width |
| aa | 0 aa bbb | Arbitrary element | Some unit | 7 | 0 | 40 |

and the list of descriptors, either in section 3 or in a table D sequence would be “0 aa bbb”. The large data width (40) wastes a lot of space and may cause software issues.

### Option 1: Scale and Magnitude

With option 1, separate scale and magnitude, a data width of 10 would be adequate. Assuming descriptor 0 08 090 was used for the scale, numbers near zero could be represented with a scale of 7 (range 0 to 1022\*10-7=1.022\*10-4 with a precision of 1.0-7) while numbers near 105 would be represented with a scale of -2 (range 0 to 1022\*102=1.022\*105 with a precision of 102). The actual space occupied would be either 18 or 26 bits (8 bits for the 0 08 090 to set the required scale and possibly another 8 for the reset back to missing, and the 10 bits for the data). Defining additional descriptors for smaller (or larger) scale ranges could minimise the required space, as could use of the change data width operator on 0 08 090. The table B entry would be something like

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Scale | Reference | Data width |
| aa | 0 aa bbb | Scaled value of arbitrary element | Some unit | 0 | 0 | 10 |

and the list of descriptors would be either 0 08 090 followed by 0 aa bbb if the next element also used scaling, or 0 08 090, 0 aa bbb, and 0 08 090 with a value of zero or missing if the next element did not use scaling. Since 0 08 090 is in class 8 and thus applies until redefined, use of it would require the inclusion of a missing or zero valued scale at the end of a sequence of scaled descriptors.

### Option 3: Delayed change of scale

With option 3, there would have to be an associated data item to describe the delayed change of scale. Assuming the change of scale operator was 2 xx yyy, and the data item to describe the scale change was 0 31 zzz then one possible set of additions could be

Add to BUFR table C

|  |  |  |  |
| --- | --- | --- | --- |
| Table reference |  |  |  |
| F X | Operand | Operator name | Operator definition |
| 2 xx | YYY | Delayed change of scale | Change the scale of the next YYY data elements in class B, excluding CCITT IA5 data and code and flag tables by the value in the data stream specified by a increment scale data descriptor from class 31, such as 0 31 zzz |

Add to BUFR table B

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Scale | Reference | Data width |
| aa | 0 aa bbb | Arbitrary element | Some unit | 2 | 0 | 10 |
| 31 | 0 31 zzz | Increment to be made to scale | Numeric | 0 | -7 | 4 |

With these values, the representable range would be from 0 to (210-2) \*10-(2-7) (1.022\*108), with a resolution of 10-(2+7) (10-9) near zero. The sequence of descriptors for a single instance using the same scale change would be 2 xx 001, 0 31 zzz, and 0 aa bbb . The data stream would consist of 14 bits. If multiple data items used the same delayed change of scale, the overheads would be smaller still.

### Option 4: Logarithmic encoding

There are at least 3 variations of this, depending on how zero is handled.

1. Ignore zero
2. Use a small offset.
3. Use a special value for zero

In case a), the table B entry would be something like

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Scale | Reference | Data width |
| aa | 0 aa bbb | Base 10 logarithm of arbitrary element in units of some unit before taking the logarithm | log10(Some unit) | 4 | -80000 | 17 |

This would allow the representation of values between 10-8 and approximately 1.28\*105 with a resolution of more than 3 digits across the entire range.

In case b) the table B entry would be something like

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Scale | Reference | Data width |
| aa | 0 aa bbb | Base 10 logarithm of arbitrary element plus 10-7 in units of some unit before taking the logarithm | log10(Some unit) | 4 | -70000 | 17 |

This would allow the representation of values between 0 and approximately 1.28\*106 with a resolution of approximately 2.3\*10-11 near zero and 2.9\*102 near the maximum.

In case c) the table B entry would be something like

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Scale | Reference | Data width |
| aa | 0 aa bbb | Base 10 logarithm of arbitrary element in units of some unit before taking the logarithm (see note x) | log10(Some unit) | 4 | -70001 | 17 |

Note x: values of zero shall be encoded by setting all bits zero. The resulting decoded value before checking for zero would be approximately 9.9978\*10-8.

This would allow the representation of values between 0 (represented by a special value) and approximately 1.28\*106 with a resolution of approximately 2.3\*10-11 near 10-7 and 2.9\*102 near the maximum. The smallest non-zero representable value would be 10-7.

In all these cases, the sequence of descriptors would contain just the table B reference 0 aa bbb. Also, there are obviously many other choices of the exact encoding. Base 2 (or even base e natural logarithms) logarithms instead of base 10 are also a possibility. The data stream would contain 17 bits.

### Option5: IEEE floating point

With this option, either different columns would be required in table B, or the meaning of the existing columns would have to change. One possibility would be to change the meaning of the scale column to allow for a keyword value “IEEE float” or similar, with appropriate notes added to the description of the table columns in the regulations, including specifying that the data width for IEEE float must be one of 16, 32, 64 or 128. The available data widths are 16 (approximately 3 decimal digits, -3\*104 to 3\*104), 32 (7 digits, -1039 to +1039), 64 (16 digits, -10308 to +10308) or 128 (34 digits, -104932 to +104932). In this case, the IEEE 16 bit type has inadequate range, so 32 would be required and the table B entry would be something like

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Scale | Reference | Data width |
| aa | 0 aa bbb | Arbitrary element | Some unit | IEEE float | Not applicable | 32 |

The sequence of descriptors would continue to contain just the table B reference. The data stream would contain 32 bits.

### Option 6: Custom floating point

Table B would need additional columns. For a custom floating point format the columns would be sign bits (0 or 1), exponent bits, and significand bits. A column to distinguish between normal (Fixed point) and floating point formats would also be needed. One possibility would be to replace the Scale, Reference and Data width columns with Format, Scale or sign bits, Reference value or exponent bits, and Data width or significand bits. There are many other possibilities. Regulations and notes would have to be added to describe the format and how it should be expanded.

These regulations and notes could include:

1. If the format is “Fixed”, then the remaining columns have meaning Scale, Reference and Data width. If the format is “Float: the remaining columns are Sign bits, Exponent bits, and Significand bits.
2. The number of bits allocated to the sign bit will be 0 or 1. Zero bits for the sign will mean that the values are all positive. One bit for the sign will be interpreted as 0 being positive and one being negative.
3. An exponent value of zero will indicate a denormal value.
4. For exponent values other than zero, If the widths of the exponent and significand are e and m respectively, and the values for the sign (0 if the width of the sign bit field is 0), exponent and significand fields are S, E and M respectively, then the value will be (-1\*\*S)\*2\*\*(E+1-2\*\*(e-1))\*(1+M/2\*\*m)
5. For exponent values of 0 (denormal values), using the same symbols as above, the value will be (-1\*\*S)\*2\*\*(2-2\*\*(e-1))\*M/2\*\*m

Note that \*\* is used for exponentiation to avoid the issues with nested exponents.

It may also be considered desirable for a maximal valued exponent to mean either infinity or NaN, as is done in IEEE floating point.

With a system like that above, one possibility for a table B entry to handle the requirements above would be

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Table Reference |  |  |  |  |  |  |
| Class | F X Y | Element name | Unit | Format | Sign bits | Exponent bits | Significand bits |
| aa | 0 aa bbb | Arbitrary element | Some unit | Float | 0 | 6 | 10 |

These values would give a range of 0 to approximately 4.3\*109. The precision near the maximum would be approximately 2.1\*106. If denormal values are not permitted then the smallest non-zero value would be approximately 9.3\*10-10 with a precision near this of approximately 9.1\*10-13. If denormal values are allowed, the smallest non-zero value would be approximately 9.1\*10-13 with the precision being the same.

The sequence of descriptors would continue to contain just the table B reference. The data stream would contain 16 bits.