Mixed-Endian Byte Orders

Status of This Document

Grid Working Document (GWD)

Document Change History

2014-07-24 Created. Broken out from bit order documents.

Copyright Notice

Copyright © Open Grid Forum (2014). Some Rights Reserved. Distribution is unlimited.

Abstract

Discusses mixed-endian byte orders for possible future inclusion into DFDL.

.

Contents

[Mixed-Endian Byte Orders 1](#_Toc393987642)

[1. Introduction 3](#_Toc393987643)

[2. A General Naming Scheme for Mixed-Endian Byte Orders 4](#_Toc393987644)

[3. Lengths and Mixed-Endian Byte Orders 5](#_Toc393987645)

[3.1 Allowing Any Length for Mixed-Endian Byte Orders 5](#_Toc393987646)

[4. Historical 8](#_Toc393987647)

[4.1.1 Mixed-endian byte order 8](#_Toc393987648)

[4.1.2 The byteOrder 'bigEndianIn16BitLittleEndianWords' 8](#_Toc393987649)

[5. Back Matter 9](#_Toc393987650)

[5.1 Security Considerations 9](#_Toc393987651)

[5.2 Contributors 9](#_Toc393987652)

[5.3 Intellectual Property Statement 9](#_Toc393987653)

[5.4 Disclaimer 9](#_Toc393987654)

[5.5 Full Copyright Notice 9](#_Toc393987655)

# Introduction

There are several mixed-endian byte orders. This document discusses topics around possible addition of them to the DFDL language.

# A General Naming Scheme for Mixed-Endian Byte Orders

The general scheme here is that the data is chunked into words of some size. The words appear in either big/little endian position. The bytes within each word appear within that word in either big/little endian position.

So a general naming scheme:

* <bytes within word order> [ In <word size> <word order> Words]

The complete set of possibilities:

* bigEndian - no word size to consider
* littleEndian - no word size to consider
* bigEndianIn16BitLittleEndianWords[[1]](#footnote-2) - 16-bit words are in little-endian order, but within words the bytes are in big-endian order
* bigEndianIn32BitLittleEndianWords - 32 bit words in little-endian order, but within words the bytes are in bigEndian order.
* littleEndianIn16BitBigEndianWords[[2]](#footnote-3) - 16-bit words are in big-endian order, but within words the bytes are in little-endian order
* littleEndianIn32BitBigEndianWords - 32 bit words in big-endian order, but within words the bytes are in littleEndian order.

It is unclear if there were ever machines creating some of these. Some may simply not be needed. (The 32-bit word variants are particularly suspect. There is no mention of them in the Wikipedia article on Endianness for example.)

Note also that combinations where the byte order and word order are the same order are not required. E.g., littleEndian16BitWordsLittleEndian would be equivalent to just littleEndian. These additional enums are only needed for mixed-endian situations.

# Lengths and Mixed-Endian Byte Orders

The lengths could be restricted to be either a multiple of the word size, or exactly 2 times the word size.

A more flexible length capability might work like this:

* For any length <= word size, behaves as just the byte order bigEndian or littleEndian. Word aspect is ignored.
  + Example: for byte order bigEndianIn16BitLittleEndianWords, if the length is 11 bits, it behaves exactly like bigEndian with length 11 bits.
* The only length greater than the word size allowed is 2 times the word size.
  + Example: for byte order bigEndianIn16BitLittleEndianWords, if the length is 32 bits, that is acceptable, but any other length > 16 is a schema definition error.

The alternative is to determine the right semantics for bit-fields in these mixed endian schemes, and in that case any length (perhaps up to 2 time the word size as maximum) would be allowed.

## Allowing Any Length for Mixed-Endian Byte Orders

Consider this example. The context is where dfdl:byteOrder is 'bigEndianIn16BitLittleEndianWords' and dfdl:bitOrder is 'leastSignificantBitFirst' and dfdl:binaryNumberRep is 'binary'.

The example includes 4 bytes of data. Let us examine the 25 bits beginning at position 2. Below the bytes are shown left-to-right:

Positions:  
00000000 11111110 22222111 33322222  
87654321 65432109 43210987 21098765  
Bits:  
01011010 10010010 01110100 00000000

Hex values  
 5 A 9 2 7 4 0 0

The bits of interest are highlighted above. If we redisplay this same data, but reversing the order of the bytes to right-to-left, then we get:

Positions:  
33322222 22222111 11111110 00000000  
21098765 43210987 65432109 87654321  
Bits:  
00000000 01110100 10010010 01011010

Hex values  
0 0 7 4 9 2 5 A

The above shows more clearly that we are looking at a contiguous region of bits containing

0 0011 1010 0100 1001 0010 1101

However, this must now be interpreted using the byte order 'bigEndianIn16BitLittleEndianWords'. So first the data is grouped into its bytes:

xxxx xxx0 0001 1101 0100 1001 0010 1101

byte 4 byte 3 byte 2 byte 1

In the above, the 'x' bits are so-indicated so as to illustrate that they did not come from the data stream. These 'x' bits are not used.

Next, reorder the bytes into most-significant byte first and leftmost order

In this byte order, byte 3 is most significant, followed in decreasing significance by byte 4, byte 1, and byte 2.

0001 1101 xxxx xxx0 0010 1101 0100 1001

byte 3 byte 4 byte 1 byte 2 (original byte positions)

In the above note the 'x' characters which represent the bits that were not in the original 25 bits. Since these 'x' are in the middle of the bits, we shift the data from byte 3 downward into byte 4 (sign extending if the most-significant bit is 1 and the number is signed.)

*It is not at all clear this is the correct semantics. I suspect there is only general agreement what this byte order means when the number of bits is exactly 32.*

*The critical property the algorithm must preserve here is that it doesn't matter where the bits are located. Shifting them by inserting a prior field doesn't change the value.*

0000 0000 0011 1010 0010 1101 0100 1001

byte 3 byte 4 byte 1 byte 2 (original byte positions)

0 0 3 A 4 D 4 9

In the above, the bits that were originally in byte 3 are red. The green bits are 0 in this example, but in general they would be either 0 or 1 depending on whether sign-extension was required.

The value is 0x3A2D49.

# Historical

This material is here purely to preserve the comments should they be revisited in the future:

### Mixed-endian byte order

This new enum for dfdl:byteOrder would only be usable for types xs:int, ... xs:decimal.

### The byteOrder 'bigEndianIn16BitLittleEndianWords'

The name of this mixed-endian byte order follows a scheme where it first describes the byte order within words, followed by the size of the words (in bits), followed by the order of the words.

See the Appendix about the general naming scheme for mixed-endian byte orders.

*Must confirm that this byte order really is needed for link16/TDLs, and if so, is this actually the right one.*

*Link 16 requires that an entire message body be decoded aka re-ordered by swapping adjacent 16-bit word pairs, and possibly byte order within those words.*

*If that is the only use of the 16-bit word swapping, and it is not needed as well for these 'mixed endian' integers, then this new byte order is not needed (for link16 anyway)*

*CONFIRMED: This swapping is a bulk decode pre-proccessing, not a per-field byte order. (2014-07-24)*

*In that case new mixed-endian byte orders could just be x- extensions until a later point release of the DFDL standard.*

In formats related to MIL-STD-2045 some (but not all) integers are stored in a mixed byte order.

In this format the integer must have a multiple of 16-bits as its length.

*We should consider requiring the length to be exactly 16 or exactly 32 bits.*

The data is stored littleEndian, but the units are 16-bit words, not individual bytes.

Each 16-bit word is stored as a pair of bytes; however, this pair is stored big-endian.

For example, consider the integer 0x1A2B3C4D. In each byte order this is stored as the given byte sequence:

* bigEndian: 1A, 2B, 3C, 4D
* littleEndian: 4D, 3C, 2B, 1A
* littleEndian16Bit: 3C, 4D, 1A, 2B

This new enum for dfdl:byteOrder would only be usable for types xs:int, xs:long, xs:unsignedInt, xs:unsignedLong, xs:integer, xs:nonNegativeInteger, and xs:decimal. Furthermore this new byte order would only be allowed when dfdl:binaryNumberRep is 'binary'.

As with the other enum values for dfdl:byteOrder, it is necessary to determine the byte order at parse time, or via external variable bindings.

This new enum can also be used in conjunction with both bit order mostSignificantBitFirst, and bit order leastSignificantBitFirst.

# Back Matter

## Security Considerations

TDML does not address security issues directly, but correctness of a DFDL implementation is critical to system security of any system using DFDL when accessing data reading or writing. Some security considerations for DFDL are described in [DFDL].

## Contributors

Michael J. Beckerle (Corresponding Author)

Tresys Technology

Columbia, MD

USA

Email: mbeckerle@tresys.com

## Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

## Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

## Full Copyright Notice

Copyright (C) Open Grid Forum (2014). Some Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included as references to the derived portions on all such copies and derivative works. The published OGF document from which such works are derived, however, may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing new or updated OGF documents in conformance with the procedures defined in the OGF Document Process, or as required to translate it into languages other than English. OGF, with the approval of its board, may remove this restriction for inclusion of OGF document content for the purpose of producing standards in cooperation with other international standards bodies.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

1. This byte order is called "little endian with atomic size 16-bit" in Wikipedia. [↑](#footnote-ref-2)
2. Wikipedia describes this byte order as 'PDP-Endian' since it was produced by that family of computers. [↑](#footnote-ref-3)