Michael J. Beckerle, Tresys Technology

Data Format Description Language (DFDL) v1.0

Experience Document 3

Bit Order and Encodings

Status of This Document

Grid Working Document (GWD)

Abstract

This document provides experience information to the OGF community on the original Data Format Description Language (DFDL) 1.0 specification (GFD-P-R.174).

Modeling a MIL-STD-2045 header in DFDL v1.0 is not possible without the addition of new capabilities for specifying bit order and non-standard encodings. There are many related military-standard binary data formats which are similar, and so cannot be modeled in DFDL.This document describes the new properties and property values that are required to successfully model this format.

All resulting errata have been incorporated into a revised Data Format Description Language (DFDL) 1.0 specification (GFD-P-R.207) which obsoletes GFD-P-R.174.

Michael J. Beckerle, Tresys Technology

Contents

1. Introduction 3
2. DFDL property dfdl:bitOrder 4
2.1 New Section 11.3 - Byte Order and Bit Order 5
2.2 New Section 11.4 - dfdl:bitOrder Example 5
2.3 New Section 11.4.1 - Example Using Right-to-Left Display for 'leastSignificantBitFirst' ...6
2.4 New Section 11.4.2 - dfdl:bitOrder and Grammar Regions 7
2.5 New Section 12.1.4 - Example: AlignmentFill....7
2.6 Change to Section 22 - Precedence Rules....8
2.7 Changes to Section 13.7 - Properties Specific to Number with Binary Representation....9
2.8 Change to Section 13.7.1.4.1 - Add Example of LSB 10
3. DFDL Standard Encodings 11
3.1 Changes to Section 3 - Glossary....11
3.2 Changes to Section 11 - Encoding....11
3.3 New Section 34 – Appendix D 11
4. Security Considerations....19
5. Authors and Contributors....20
6. Intellectual Property Statement....21
7. Disclaimer 22
8. Full Copyright Notice....23
9. References 24

Michael J. Beckerle, Tresys Technology

1. Introduction

There are a number of military standard binary data formats. The definitions of these formats are not generally available to the public. However, MIL-STD 2045 is a publicly available standard for a binary header used in conjunction with many other binary data formats and it illustrates most of the modeling complexities of the general family of MIL-STD binary formats [MILSTD2045].

Modeling the MIL-STD-2045 header, as well as the related binary data formats, is not possible without the addition of additional new capabilities for DFDL to allow:

1. The filling of bytes least significant bit first;
2. Encodings that are neither an IANA name nor a CCSID.

DFDL Working Group Action 233 was raised in January 2014 to address this. This document records the conclusions of this Action and should be treated as a companion document to DFDL 1.0 Experience Document 1 [DFDLX1]. Specifically it provides the detailed content for these errata:

- Erratum **4.24**, which introduces a new DFDL property dfdl:bitOrder.
- Erratum **2.107**, which introduces the concept of DFDL standard encodings and defines two such encodings.

To prove feasibility, the new dfdl:bitOrder property and the DFDL standard encodings have been implemented as part of the Daffodil open-source DFDL project [Daffodil].

2. DFDL property dfdl:bitOrder

The new dfdl:bitOrder property is added to Section 11 along with new sub-sections.

Property Name	Description
bitOrder	Enum Valid values 'mostSignificantBitFirst', 'leastSignificantBitFirst'. The bits of a byte each have a place value or numeric significance. Hence, the byte value 255 = $2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$. A bit of a byte can always be unambiguously identified as the 2^n-bit. Meaning the bit which, if set, contributes 2^n to the magnitude of the number. The bit order is the correspondence of a bit's numeric significance to the bit position within the byte. As with all other numbering in DFDL and XML Schema, positions begin at 1, so the bits of a byte are numbered 1 to 8. See Section 11.4 below for an illustration. Value 'mostSignificantBitFirst' means: • The bit having numeric significance 2^7 is first, i.e., has bit position 1. • In general the 2^n bit has position 8 - *n.* • The least significant bits of byte N are considered to be adjacent to the most significant bits of byte N+1. Value 'leastSignificantBitFirst' means: • The bit having numeric significance 2^0 is first, i.e., has bit position 1. • In general the 2^n bit has position *n* + 1. • The most significant bits of byte N are considered to be adjacent to the least significant bits of byte N+1.

	This property applies to all content and framing since it determines which bits of a byte occupy what bit positions. Content and framing are defined in terms of regions of the data stream, and these regions are defined in terms of the starting bit position and ending bit position; hence, dfdl:bitOrder is relevant to determining the specific bits of any grammar region (see Section 9.2) when the region's starting bit position or ending bit position are not on a byte boundary. The bit order can only change on byte boundaries, and alignment of up to 7 bits will be inserted to ensure byte-alignment whenever the bit order changes. When unparsing, this alignment region is filled using bits from the dfdl:fillByte. Annotation: dfdl:element, dfdl:simpleType, dfdl:sequence, dfdl:choice, dfdl:group

2.1 New Section 11.3 - Byte Order and Bit Order

The material here is inserted as a new Section 11.3, including the footnote.

Byte order and bit order are separate concepts. However, of the possible combinations, only the following are allowed:

1. 'bigEndian' with 'mostSignificantBitFirst'
2. 'littleEndian' with 'mostSignificantBitFirst'
3. 'littleEndian' with 'leastSignificantBitFirst' [1]

The remaining combination is a schema definition error.

2.2 New Section 11.4 - dfdl:bitOrder Example

The material here is inserted as a new Section 11.4.

---

[1] Used by data format MIL-STD-2045

Consider a structure of 4 logical elements. The total length is 16 bits. Assume dfdl:lengthUnits is 'bits', dfdl:representation is 'binary', dfdl:binaryNumberRep is 'binary':

```
<element name="A" type="xs:int" dfdl:length="3"/> <!-- having value 3 -->
<element name="B" type="xs:int" dfdl:length="7"/> <!-- having value 9 -->
<element name="C" type="xs:int" dfdl:length="4"/> <!-- having value 5 -->
<element name="D" type="xs:int" dfdl:length="2"/> <!-- having value 1 -->
```

The above are colorized so as to highlight the corresponding bits in the data below.

In a format where dfdl:bitOrder is 'mostSignificantBitFirst':

```
              01100010 01010101
              AAABBBBB BBCCCCDD
Significance  M      L M      L
Bit Position  12345678 12345678
Byte Position ----1--- ----2---
```

As presented here, the bits corresponding to each element appear left to right, and all bits for an individual element are adjacent. Within the bits of an individual element the most significant bit is on the left, least significant on the right, consistent with the way the bytes themselves are presented.

In contrast, in a format where dfdl:bitOrder is 'leastSignificantBitFirst':

```
              01001011 01010100
              BBBBBAAA DDCCCCBB
Significance  M      L M      L
Bit Position  87654321 87654321
Byte Position ----1--- ----2---
```

In the above presentation note how the bits of the element 'B' do not appear adjacent to each other. The most significant bits of byte N are adjacent to the least significant bits of byte N+1.

2.3 New Section 11.4.1 - Example Using Right-to-Left Display for 'leastSignificantBitFirst'

The material here is inserted as a new Section 11.4.1.

When working exclusively with data having dfdl:bitOrder 'leastSignificantBitFirst', it is useful to present data with bytes Right to Left. That is, with the bytes starting at byte 1 on the right, and increasing to the left.

```
              01010100 01001011
              DDCCCCBB BBBBBAAA
Significance  M      L M      L
Bit Position  87654321 87654321
Byte Position ----2--- ----1---
```

With this reorientation, the bits of the element 'B' are once again displayed adjacently. Within the bits of an individual element the most significant bit is on the left, least significant on the right, consistent with the way the bytes themselves are presented.

Often the specification documents for data formats that with least-significant-bit-first bit order will describe data using this Right-to-Left presentation style.

2.4 New Section 11.4.2 - dfdl:bitOrder and Grammar Regions

The material here is inserted as a new Section 11.4.2.

When any grammar region appears before (to the left of) or after (to the right of) another grammar region in the grammar rules of Section 9.2, and the boundary between the two falls within a byte rather than on a byte boundary, then the dfdl:bitOrder determines which bits are occupied by the regions.

In general, the notion of *before* means occupying lower-numbered bit positions, and the bit positions are numbered according to dfdl:bitOrder. Hence, when dfdl:bitOrder is 'mostSignificantBitFirst', grammar regions that are before, will occupy more-significant bits, and when dfdl:bitOrder is 'leastSignificantBitFirst', grammar regions that are before will occupy less-significant bits.

2.5 New Section 12.1.4 - Example: AlignmentFill

The material here is inserted as a new Section 12.1.4.

When dfdl:alignmentUnits is 'bits', and the dfdl:alignment is not a multiple of 8, then the dfdl:bitOrder property affects the alignment by controlling which bits are skipped as part of the grammar ***AlignmentFill*** region.

In general, the ***AlignmentFill*** region is *before* the regions it is aligning, and within a byte, the meaning of *'before'* is interpreted with respect to the dfdl:bitOrder.

When dfdl:bitOrder is 'mostSignificantBitFirst', then bits with more significance are before bits with less significance, so the ***AlignmentFill*** region occupies more significant bits of the byte relative to the bits of the element itself. When dfdl:bitOrder is 'leastSignificantBitFirst', then bits with less significance are before bits with more significance, so the ***AlignmentFill*** region occupies less significant bits of the byte relative to the bits of the element itself.

Consider a structure of 2 logical elements. Assume dfdl:lengthUnits='bits', dfdl:representation='binary', dfdl:binaryNumberRep='binary', dfdl:alignmentUnits='bits', and assume the data is at the begining of the data stream.

```
<element name="A" type="xs:int" dfdl:length="2" dfdl:alignment='8'/>
<!-- having value 1 -->
<element name="B" type="xs:int" dfdl:length="4" dfdl:alignment='4'/>
<!-- having value 5 -->
```

The above are colorized so as to highlight the corresponding bits in the data below. The total length due to the alignment region appearing before element 'B' will be 8 bits.

In a format where dfdl:bitOrder is 'mostSignificantBitFirst' the data can be visualized as:

```
               01000101
               AAxxBBBB
Significance   M      L
Bit Position   12345678
```

In the above, the ***AlignmentFill*** region is marked with 'x' characters, and contains all 0 bit values.

In a format where dfdl:bitOrder is 'leastSignificantBitFirst' the presentation is different:

```
               01010001
               BBBBxxAA
Significance   M      L
Bit Position   87654321
```

In the above the ***AlignmentFill*** region still appears before element 'B', and in this case that is in less significant bits of the byte than the bits of content of element 'B', and these bits are displayed to the right of the bits of element 'B'.

2.6 Change to Section 22 - Precedence Rules

The dfdl:bitOrder property needs adding to the property precedence tables. Specifically, add into all 'Parsing: Common' and 'Unparsing: Common' sub-sections ahead of dfdl:encoding.

2.7 Changes to Section 13.7 - Properties Specific to Number with Binary Representation

Uses of the term "left" to be replaced with "most significant", "more significant", or "before" depending on the context.

Uses of the term "right" to be replaced with "least significant", "less significant", or "after" depending on the context.[2]

Specifically:

2.7.1 **dfdl:binaryNumberRep**

Revise bullets containing "left"

- A negative number is indicated by digits with the most significant nibble being xD.
- If a positive or negative value packs to an odd number of nibbles, an extra xF nibble is added as the most significant nibble

In the description of the dfdl:binaryNumberRep property, the paragraph:

> For all values, the dfdl:byteOrder property is used to determine the numeric significance of the bytes making up the representation.

Should be extended to say:

> For all values, the dfdl:byteOrder property is used to determine the numeric significance of the bytes making up the representation, and the dfdl:bitOrder property is used to determine the numeric significance of the bits within a byte.

2.7.2 **dfdl:binaryVirtualDecimalPoint.**

Revise paragraphs using "left" and "right" to:

> If you specify a positive integer, the position of the decimal point is moved from the least-signficant side of the number toward the most-significant side of the number. For

[2] There are other uses of "left" and "right" in the specification but they are not problematic (e.g., LeftFraming and RightFraming refer to position within the text of the grammar rule).

example, if 3 is specified then, the integer value 1234 represents 1.234. This is equivalent to dividing by $10^{3.}$

If you specify a negative integer, the position of the decimal point is moved from the least significant side of the number further in the less-significant direction. For example, if you specify -3, the integer value 1234 represents 1 234 000.This is equivalent to multiplying by 10^{3}.

2.8 Change to Section 13.7.1.4.1 - Add Example of LSB

This snippet is a reproduction from the specification, for reference.

```
Positions:
00000000 01111111 11122222
12345678 90123456 78901234
Bits:
01011010 10010010 00000000
Hex values
   5   A    9   2    0   0
```

The material below here is added to the end of Section 13.7.1.4.1 as a new example.

Now let us examine the 13 bits beginning at position 2, in the context where dfdl:byteOrder is 'littleEndian' and dfdl:bitOrder is leastSignificantBitFirst and dfdl:binaryNumberRep is 'binary'.

In this case, the bit positions are assigned differently. Below the bytes are shown left-to-right:

```
Positions:
00000000 11111110 22222111
87654321 65432109 43210987
Bits:
01011010 10010010 00000000
Hex values
   5   A    9   2    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:
22222111 11111110 00000000
43210987 65432109 87654321
Bits:
00000000 10010010 01011010
Hex values
   0   0    9   2    5   A
```

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

```
 0  1001 0010 1101
```

or the value 0x92D.

3. DFDL Standard Encodings

3.1 Changes to Section 3 - Glossary

A definition of DFDL standard encoding is added.

DFDL Standard Encoding - A character set for which there is no IANA name or CCSID but the name and definition of which DFDL implementations must agree on. See Section 34: Appendix D: DFDL Standard Encodings.

3.2 Changes to Section 11 - Encoding

The property definition for dfdl:encoding is changed to distinguish DFDL standard encodings from other implementation-defined encoding names.

DFDL standard encoding names are defined in Section Section 34: Appendix D: DFDL Standard Encodings. When supported, a conforming DFDL implementation must implement them in a uniform manner so that they are portable across all DFDL implementations that implement them.

Additional implementation-defined encoding names may be provided only for character set encodings for which there is no IANA name standard nor CCSID standard nor DFDL standard encoding. These implementation-defined encodings must have "X-" as a prefix to their name, as they are subject to being superseded by IANA or DFDL standard encoding names.

3.3 New Section 34 – Appendix D

The material below is added as a new Appendix D to the DFDL specification.

34. Appendix D: DFDL Standard Encodings

This appendix provides a specification template for the definition of DFDL standard encodings, and defines two such encodings for use in DFDL schemas.

These encodings are optional - DFDL implementations are not required to support them to be considered to be in conformance with the DFDL specification.

However, if implemented, then a conformant DFDL implementation must follow the specification in this appendix.

34.1 Purpose

Additional encoding names are needed for several reasons:

1. Incomplete Information: the definitions of IANA [IANA] and CCSID [CCSID] character set encodings are not sufficiently specific about the encoding, omitting important characteristics such as alignment or bit order. An example of this is the encoding X-DFDL-US-ASCII-7-BIT-PACKED, defined herein.

2. Unnamed or Legacy Encodings: There are many legacy encodings that have no IANA or CCSID identifiers. For example ITA2 [ITA2] is a 5-bit-per character encoding that is very old, yet still has modern uses.

34.2 Conventions

All indexing of positions are 1-based meaning the first position is position 1.

Bytes are unsigned integers with values from 0 to 255 (decimal).

Hexadecimal numbers have the prefix "0x".

Decimal numbers have no prefix when the context is clear, and are explicitly labeled with (decimal) to resolve ambiguity.

34.3 Specification Template

A DFDL standard encoding specification must include:

1. *Name*: All DFDL standard encoding names will begin with the prefix "X-DFDL-". Note that encoding names are not case-sensitive.
2. *Translation table:* An encoding/decoding algorithm that provides a complete bi-directional mapping between encoded characters and UTF-16 character codes. Many encodings share character code definitions and describe only different ways of representing those values; hence, this section may reference other IANA, CCSID or DFDL standard encoding definitions.
3. *Width*: Whether the character set is fixed or variable width. If fixed width, the size of a code unit in bits or bytes.
4. *Alignment:* Alignment and alignment units for a code unit.
5. *Byte order:* When a code unit consists of more than one byte of data, the byte order of the code units. Note that the dfdl:byteOrder property does not apply to text.

34.4 Encoding X-DFDL-US-ASCII-7-BIT-PACKED

This encoding is used by MIL-STD-2045 [MILSTD2045] and a variety of related binary data standards.

34.4.1 Name

X-DFDL-US-ASCII-7-BIT-PACKED

34.4.2 Translation table

The characters are exactly the 128 US-ASCII characters with the same code unit values.

34.4.3 Width

Fixed width. Each code unit is 7-bits wide.

This encoding differs from US-ASCII in that it specifically uses only 7-bits per character within each byte[3]. A subsequent character or the next data field may begin in the very next bit after a 7-bit character code of this encoding.

Hence, 8 characters will fit exactly into 7 bytes of data as no bits are wasted.

34.4.4 **Alignment**

1 bit. That is, a code unit may begin on any bit boundary.

34.4.5 **Byte Order**

Not applicable - code units are always smaller than 1 byte.

34.4.6 **Example 1**

The table below shows how the 8 character string 'UNIT1234' is encoded into 7 bytes, when the bit order is least-signficant-bit first.

Logical character	**7-bit code unit**	**Bit placement** 2^7 2^0	**Byte value**	**Byte number**
U	1010101	X**1010101**		
N	1001110	**0**XXXXXXX XX**100111**	0x55	1
I	1001001	**01**XXXXXX XXX**10010**	0x67	2
T	1010100	**100**XXXXX XXXX**1010**	0x92	3
1	0110001	**0001**XXXX XXXXX**011**	0x1A	4

[3] Though US-ASCII contains only 128 characters DFDL assumes the US-ASCII encoding to use one character per 8-bit byte.

2	0110010	**10010**XXX XXXXXX**01**	0x93	5
3	0110011	**110011**XX XXXXXXX**0**	0xCD	6
4	0110100	**0110100**X	0x68	7

Table 1: Example of DFDL Standard Encoding X-DFDL-US-ASCII-7-BIT-PACKED

Below is the same data shown horizontally with the bytes numbered from right to left. The bits corresponding to the characters are shown in distinct colors. The characters that each bit contributes to are written below each of the bits and in the corresponding color.

```
     01010101 01100111 10010010 00011010 10010011 11001101 01101000
Byte -----1-- -----2-- -----3-- -----4-- -----5-- -----6-- -----7--
Char NUUUUUUU IINNNNNN TTTIIIII 1111TTTT 22222111 33333322 44444443
```

This kind of data is often better visualized by numbering the bytes in the opposite direction that is *right to left*, which leads to this presentation:

```
     01101000 11001101 10010011 00011010 10010010 01100111 01010101
Byte -----7-- -----6-- -----5-- -----4-- -----3-- -----2-- -----1--
Char 44444443 33333322 22222111 1111TTTT TTTIIIII IINNNNNN NUUUUUUU
```

In the above, the bits corresponding to each character code unit are more easily recognized, but the characters appear right-to-left (i.e., backward for English).

34.4.7 **Example 2**

The bits below represent a 3-bit unsigned integer containing value 7, followed by the ASCII string 'ABC' followed by the ASCII DEL character (character code 0x7F). This illustrates a string not beginning on a byte boundary. Again the bit ordering is least-signficant-bit first.

The bits are written in increasing position and place value from *right-to-left*:

```
 1111111 1000011 1000010 1000001 111
 DEL     C       B       A       7
```

In the above example, if we number the bits from the right, starting with position 1, the character code for 'A' occupies bits 4 to 10. If we divide the data above into bytes with vertical bars we must start on the right to get:

```
 char:      DEL       C       B         A
 bits:     01111111 | 1000011 1 |000010 10 | 00001 111
 byte:     7F         87         1C          0F
 byte pos: 4          3          2           1
```

34.5 Encoding X-DFDL-US-ASCII-6-BIT-PACKED

This encoding is used by MIL-STD-6016D (See [MILSTD6016]) and a variety of related binary data standards.

34.5.1 Name

X-DFDL-US-ASCII-6-BIT-PACKED

34.5.2 Translation Table

The characters are a subset of the 128 US-ASCII characters. The assignment of numeric code units to the characters is different for some of the characters and the same for others.

The correspondence of ASCII characters to their code point values is given by the table below.

Character	Code Point Value (Decimal)	Notes
@	0	These characters' code points are the same as the US-ASCII code point, minus 64 (decimal)
A to Z	1 through 26	
[	27	
\	28	
]	29	
^	30	
_	31	
SPACE	32	These characters' code points are the same as in US-ASCII.
!	33	
"	34	
#	35	
$	36	
%	37	
&	38	

' APOSTROPHE	39	
(	40	
)	41	
*	42	
+	43	
, COMMA	44	
-	45	
. PERIOD	46	
/	47	
0 to 9	48 through 57	
:	58	
;	59	
<	60	
=	61	
>	62	
?	63	

Table 2: Translation Table for DFDL Standard Encoding X-DFDL-US-ASCII-6-BIT-PACKED

34.5.3 **Width**

Fixed width. Each code unit is 6-bits wide. A subsequent character or the next data field may begin in the very next bit after a 6-bit character code of this encoding.

Hence, 4 characters will fit into 3 bytes of data as no bits are wasted.

34.5.4 **Alignment**

1 bit. That is, a code unit may begin on any bit boundary.

34.5.5 **ByteOrder**

Not applicable. Code units are always less than 1 byte in size.

34.5.6 **Example 1**

The table below shows how the 8 character string '1234' is encoded into 3 bytes. The bit order is least-significant-bit first:

Logical character	**6-bit code unit**	**Bit placement** 2^7 2^0	**Byte value**	**Byte number**
1	110001	XX**110001**		
2	110010	**10**XXXXXX XXXX**1100**	0xB1	1
3	110011	**0011**XXXX XXXXXX**11**	0x3C	2
4	110100	**110100**XX	0xD3	3

Table 3: Example of DFDL Standard Encoding X-DFDL-US-ASCII-6-BIT-PACKED

4. Security Considerations

Security considerations are dealt with in the corresponding sections of the DFDL 1.0 specification [DFDL].

No additional security issues have been raised.

5. Authors and Contributors

Michael J. Beckerle (Corresponding Author)

Tresys Technology

Columbia, MD

USA

Email: mbeckerle@tresys.com

We greatly acknowledge the contributions made to this document by the following and all the other people who provided constructive and valuable input in the group discussions.

Stephen M Hanson, IBM Software Group, Hursley, UK

Alex Wood, IBM Software Group, Hursley, UK

Mark Frost, IBM Software Group, Hursley, UK

Andrew Edwards, IBM Software Group, Hursley, UK

6. 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.

7. 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.

8. Full Copyright Notice

9. References

[Daffodil] https://opensource.ncsa.illinois.edu/confluence/display/DFDL/Daffodil%3A+Open+Source+DFDL

[DFDL] Michael J Beckerle, Steven M Hanson, Alan W Powell. Data Format Description Language (DFDL) v1.0 Specification. Open Grid Forum. (http://redmine.ogf.org/dmsf/dfdl-wg)

Forthcoming Update: GFD-P-R.207 (2014)

Obsolete: GFD-P-R.174. January 2011.

[DFDLX1] DFDL Experience Document 1 (http://redmine.ogf.org/dmsf/dfdl-wg)

[IANA] IANA, "Character Sets,"

http://www.iana.org/assignments/character-sets

[ITA2] Wikipedia: Baudot code

http://en.wikipedia.org/wiki/Baudot_code#ITA2

[MILSTD2045] CONNECTIONLESS DATA TRANSFER APPLICATION LAYER STANDARD, MIL-STD-2045-47001D w/CHANGE 1, 23 June 2008 (available publicly from US Dept. of Defense at http://assistdocs.com/)

[MILSTD6016] Tactical Data Link Message Standard MIL-STD-6016

This standard requires authorization to access.

http://en.wikipedia.org/wiki/Link_16 provides some information about the standard.

http://www.assistdocs.com/search/document_details.cfm?ident_number=123964 provides links for those with authorization to access the various versions of the standard.