

# [MC-NBFX]: .NET Binary Format: XML Data Structure

---

## Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft [Open Specification Promise](#) or the [Community Promise](#). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

## Revision Summary

| Date       | Revision History | Revision Class | Comments                                     |
|------------|------------------|----------------|--|
| 08/10/2007 | 0.1              | Major          | Initial Availability                         |
| 09/28/2007 | 0.2              | Minor          | Updated the technical content.               |
| 10/23/2007 | 0.2.1            | Editorial      | Revised and edited the technical content.    |
| 11/30/2007 | 0.3              | Minor          | Updated the technical content.               |
| 01/25/2008 | 0.3.1            | Editorial      | Revised and edited the technical content.    |
| 03/14/2008 | 0.3.2            | Editorial      | Revised and edited the technical content.    |
| 05/16/2008 | 1.0              | Major          | Updated and revised the technical content.   |
| 06/20/2008 | 2.0              | Major          | Updated and revised the technical content.   |
| 07/25/2008 | 2.0.1            | Editorial      | Revised and edited the technical content.    |
| 08/29/2008 | 2.0.2            | Editorial      | Revised and edited the technical content.    |
| 10/24/2008 | 2.0.3            | Editorial      | Revised and edited the technical content.    |
| 12/05/2008 | 2.1              | Minor          | Updated the technical content.               |
| 01/16/2009 | 2.1.1            | Editorial      | Revised and edited the technical content.    |
| 02/27/2009 | 2.1.2            | Editorial      | Revised and edited the technical content.    |
| 04/10/2009 | 2.1.3            | Editorial      | Revised and edited the technical content.    |
| 05/22/2009 | 2.2              | Minor          | Updated the technical content.               |
| 07/02/2009 | 2.2.1            | Editorial      | Revised and edited the technical content.    |
| 08/14/2009 | 2.2.2            | Editorial      | Revised and edited the technical content.    |
| 09/25/2009 | 2.3              | Minor          | Updated the technical content.               |
| 11/06/2009 | 2.3.1            | Editorial      | Revised and edited the technical content.    |
| 12/18/2009 | 2.3.2            | Editorial      | Revised and edited the technical content.    |
| 01/29/2010 | 2.4              | Minor          | Updated the technical content.               |
| 03/12/2010 | 2.4.1            | Editorial      | Revised and edited the technical content.    |
| 04/23/2010 | 3.0              | Major          | Updated and revised the technical content.   |
| 06/04/2010 | 3.0.1            | Editorial      | Revised and edited the technical content.    |
| 07/16/2010 | 4.0              | Major          | Significantly changed the technical content. |

| <b>Date</b> | <b>Revision History</b> | <b>Revision Class</b> | <b>Comments</b>  |
|-------------|-------------------------|-----------------------|--|
| 08/27/2010  | 4.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 10/08/2010  | 4.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 11/19/2010  | 4.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 01/07/2011  | 4.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 02/11/2011  | 4.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 03/25/2011  | 4.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 05/06/2011  | 4.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 06/17/2011  | 4.1                     | Minor                 | Clarified the meaning of the technical content.                              |
| 09/23/2011  | 4.1                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011  | 5.0                     | Major                 | Significantly changed the technical content.                                 |
| 03/30/2012  | 5.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 07/12/2012  | 5.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 10/25/2012  | 5.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 01/31/2013  | 5.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |
| 08/08/2013  | 5.0                     | No change             | No changes to the meaning, language, or formatting of the technical content. |

# Contents

|  |           |
|--|-----------|
| <b>1 Introduction</b> .....                                      | <b>6</b>  |
| 1.1 Glossary .....   | 6         |
| 1.2 References.....  | 6         |
| 1.2.1 Normative References.....                                  | 7         |
| 1.2.2 Informative References .....                               | 7         |
| 1.3 Overview .....   | 8         |
| 1.4 Relationship to Protocols and Other Structures .....         | 8         |
| 1.5 Applicability Statement.....                                 | 8         |
| 1.6 Versioning and Localization .....                            | 9         |
| 1.7 Vendor-Extensible Fields.....                                | 9         |
| <b>2 Structures</b> .....  | <b>10</b> |
| 2.1 Common Definitions.....                                      | 10        |
| 2.1.1 Record .....   | 10        |
| 2.1.2 MultiByteInt31 .....                                       | 13        |
| 2.1.2.1 MultiByteInt31-(1 Byte).....                             | 14        |
| 2.1.2.2 MultiByteInt31-(2 Bytes) .....                           | 14        |
| 2.1.2.3 MultiByteInt31-(3 Bytes) .....                           | 15        |
| 2.1.2.4 MultiByteInt31-(4 Bytes) .....                           | 16        |
| 2.1.2.5 MultiByteInt31-(5 Bytes) .....                           | 17        |
| 2.1.3 String .....   | 18        |
| 2.1.4 DictionaryString .....                                     | 18        |
| 2.2 Records .....  | 19        |
| 2.2.1 Element Records .....                                      | 19        |
| 2.2.1.1 ShortElement Record (0x40).....                          | 19        |
| 2.2.1.2 Element Record (0x41) .....                              | 20        |
| 2.2.1.3 ShortDictionaryElement Record (0x42).....                | 20        |
| 2.2.1.4 DictionaryElement Record (0x43) .....                    | 21        |
| 2.2.1.5 PrefixDictionaryElement[A-Z] Record (0x44-0x5D) .....    | 21        |
| 2.2.1.6 PrefixElement[A-Z] Record (0x5E-0x77).....               | 22        |
| 2.2.2 Attribute Records .....                                    | 23        |
| 2.2.2.1 ShortAttribute Record (0x04) .....                       | 23        |
| 2.2.2.2 Attribute Record (0x05).....                             | 23        |
| 2.2.2.3 ShortDictionaryAttribute Record (0x06).....              | 24        |
| 2.2.2.4 DictionaryAttribute Record (0x07) .....                  | 25        |
| 2.2.2.5 ShortXmlnsAttribute Record (0x08) .....                  | 25        |
| 2.2.2.6 XmlnsAttribute Record (0x09) .....                       | 26        |
| 2.2.2.7 ShortDictionaryXmlnsAttribute Record (0x0A) .....        | 26        |
| 2.2.2.8 DictionaryXmlnsAttribute Record (0x0B) .....             | 27        |
| 2.2.2.9 PrefixDictionaryAttribute[A-Z] Records (0x0C-0x25) ..... | 27        |
| 2.2.2.10 PrefixAttribute[A-Z] Records (0x26-0x3F) .....          | 27        |
| 2.2.3 Text Records .....   | 28        |
| 2.2.3.1 ZeroText Record (0x80) .....                             | 28        |
| 2.2.3.2 OneText Record (0x82) .....                              | 28        |
| 2.2.3.3 FalseText Record (0x84).....                             | 29        |
| 2.2.3.4 TrueText Record (0x86) .....                             | 29        |
| 2.2.3.5 Int8Text Record (0x88) .....                             | 29        |
| 2.2.3.6 Int16Text Record (0x8A) .....                            | 29        |
| 2.2.3.7 Int32Text Record (0x8C) .....                            | 30        |
| 2.2.3.8 Int64Text Record (0x8E) .....                            | 30        |

|            |  |           |
|------------|--|-----------|
| 2.2.3.9    | FloatText Record (0x90)                          | 31        |
| 2.2.3.10   | DoubleText Record (0x92)                         | 31        |
| 2.2.3.11   | DecimalText Record (0x94)                        | 32        |
| 2.2.3.12   | DateTimeText Record (0x96)                       | 33        |
| 2.2.3.13   | Chars8Text Record (0x98)                         | 34        |
| 2.2.3.13.1 | Character Escaping                               | 35        |
| 2.2.3.14   | Chars16Text Record (0x9A)                        | 36        |
| 2.2.3.15   | Chars32Text Record (0x9C)                        | 36        |
| 2.2.3.16   | Bytes8Text Record (0x9E)                         | 36        |
| 2.2.3.17   | Bytes16Text Record (0xA0)                        | 37        |
| 2.2.3.18   | Bytes32Text Record (0xA2)                        | 37        |
| 2.2.3.19   | StartListText / EndListText Records (0xA4, 0xA6) | 38        |
| 2.2.3.20   | EmptyText Record (0xA8)                          | 38        |
| 2.2.3.21   | DictionaryText Record (0xAA)                     | 38        |
| 2.2.3.22   | UniqueIdText Record (0xAC)                       | 39        |
| 2.2.3.23   | TimeSpanText Record (0xAE)                       | 39        |
| 2.2.3.24   | UuidText Record (0xB0)                           | 40        |
| 2.2.3.25   | UInt64Text Record (0xB2)                         | 41        |
| 2.2.3.26   | BoolText Record (0xB4)                           | 42        |
| 2.2.3.27   | UnicodeChars8Text Record (0xB6)                  | 42        |
| 2.2.3.28   | UnicodeChars16Text Record (0xB8)                 | 43        |
| 2.2.3.29   | UnicodeChars32TextRecord(0xBA)                   | 43        |
| 2.2.3.30   | QNameDictionaryTextRecord(0xBC)                  | 43        |
| 2.2.3.31   | *TextWithEndElement Records                      | 44        |
| 2.3        | Miscellaneous Records                            | 44        |
| 2.3.1      | EndElement Record (0x01)                         | 45        |
| 2.3.2      | Comment Record (0x02)                            | 45        |
| 2.3.3      | Array Record (0x03)                              | 45        |
| <b>3</b>   | <b>Structure Examples</b>                        | <b>48</b> |
| <b>4</b>   | <b>Security Considerations</b>                   | <b>58</b> |
| <b>5</b>   | <b>Appendix A: Product Behavior</b>              | <b>59</b> |
| <b>6</b>   | <b>Change Tracking</b>                           | <b>60</b> |
| <b>7</b>   | <b>Index</b>                                     | <b>61</b> |

# 1 Introduction

This specification defines the .NET Binary Format: XML Data Structure, which is a binary format that can represent many **XML** documents, as specified in [\[XML1.0\]](#).

This purpose of the format is to reduce the processing costs associated with XML documents by encoding an XML document in fewer bytes than the same document encoded in **UTF-8**, as specified in [\[RFC2279\]](#).

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

## 1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

**base64**  
**little-endian**  
**.NET Framework**  
**Unicode**  
**universally unique identifier (UUID)**  
**UTC (Coordinated Universal Time)**  
**UTF-8**  
**UTF-16**  
**XML**

The following terms are specific to this document:

**DictionaryString:** A structure defined in section [2.1.4](#) that uses a **MultiByteInt31** to refer to a **string**.

**MultiByteInt31:** A structure defined in section [2.1.2](#) that encodes small integer values in fewer bytes than large integer values.

**record:** The fundamental unit of information in the .NET Binary Format: XML Data Structure encoded as a variable length series of bytes. Section [2](#) specifies the format for each type of **record**.

**String:** A structure defined in section [2.1.3](#) that represents a set of characters.

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as specified in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [\[Windows Protocol\]](#).

## 1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[IEEE854] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE 854-1987, October 1987, <http://ieeexplore.ieee.org/iel1/2502/1121/00027840.pdf?tp=&arnumber=27840&isnumber=1121>

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO/IEC 8601:2004, December 2004, <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30>

**Note** There is a charge to download the specification.

[MS-OAUT] Microsoft Corporation, "[OLE Automation Protocol](#)".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/rfc/rfc2119.txt>

[RFC2279] Yergeau, F., "UTF-8, A Transformation Format of ISO10646", RFC 2279, January 1998, <http://www.ietf.org/rfc/rfc2279.txt>

[RFC2781] Hoffman, P., and Yergeau, F., "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000, <http://www.ietf.org/rfc/rfc2781.txt>

[RFC3548] Josefsson, S., Ed., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003, <http://www.ietf.org/rfc/rfc3548.txt>

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, <http://www.ietf.org/rfc/rfc4122.txt>

[UNICODE] The Unicode Consortium, "Unicode Home Page", 2006, <http://www.unicode.org/>

[XML1.0] Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>

## 1.2.2 Informative References

[IEEE754] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, <http://ieeexplore.ieee.org/servlet/opac?punumber=2355>

[MC-NBFS] Microsoft Corporation, "[.NET Binary Format: SOAP Data Structure](#)".

[MC-NBFSE] Microsoft Corporation, "[.NET Binary Format: SOAP Extension](#)".

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

[XML-INFOSET] Cowan, John, and Tobin, Richard, "XML Information Set (Second Edition)", W3C Recommendation, February 2004, <http://www.w3.org/TR/2004/REC-xml-infoset-20040204>

### 1.3 Overview

The .NET Binary Format: XML Data Structure is used to efficiently represent XML 1.0 documents, as specified in [\[XML1.0\]](#).

### 1.4 Relationship to Protocols and Other Structures

The .NET Binary Format: XML Data Structure is extended by the NET Binary Format: SOAP Data Structure, as described in [\[MC-NBFS\]](#), and the .NET Binary Format: SOAP Extension, as described in [\[MC-NBFSE\]](#).

### 1.5 Applicability Statement

The .NET Binary Format: XML Data Structure is a general-purpose way to represent an XML document that offers many benefits in terms of reduced size and processing costs, but at the expense of human readability. However, the .NET Binary Format: XML Data Structure is capable of representing only a subset of information described by an XML information set (infoset), as described in [\[XML-INFOSET\]](#). It does not represent all syntactic aspects of an XML document encoded textually.

Some constructs have more than one form, of which the .NET Binary Format for XML Data Structure supports one form. For example, the standard (short) form of an empty element is not supported, but the more general form (with open and close tags) is supported.

```
<element/>                                <!-- Not supported -->
<element></element>                       <!-- Supported -->
```

Other constructs are not supported, although a functionally equivalent construct is supported by the .NET Binary Format for XML Data Structure. For example, a CDATA section cannot be encoded; however, a semantically equivalent construct can be encoded.

```
<element><![CDATA[hello world]]></element> <!-- Not supported -->
<element>hello world</element>           <!-- Supported -->
```

Character references are necessary in textual XML in order to disambiguate document structure from document content. The .NET Binary Format: XML Data Structure uses **records** to distinguish between structure and content, making character references unnecessary.

Insignificant spaces in an element or end element are not supported.

```
<element a = "value" ></element >       <!-- Not supported -->
```

Processing instructions, data type definitions (DTDs), and declarations are not supported and cannot be represented by this format.

The following table identifies the items that are not available in the .NET Binary Format for XML Data Structure.

| Unsupported construct | Example              |
|-----------------------|----------------------|
| Xml Declaration       | <?xml version="1.0"> |



| Unsupported construct                               | Example   |
|---|---|
| Processing Instruction                              | <code>&lt;?pi?&gt;</code>   |
| DTD   | <code>&lt;!DOCTYPE ...</code>   |
| Character Reference                                 | <code>&lt;element&gt;&amp;amp;&lt;/element&gt;</code>                     |
| Empty Element (short form)                          | <code>&lt;element/&gt;</code>   |
| CDATA Section                                       | <code>&lt;element&gt;&lt;![CDATA[hello world]]&gt;&lt;/element&gt;</code> |
| Insignificant White Space (in or around an element) | <code>&lt; element a = "value" &gt;&lt;/element &gt;</code>               |

## 1.6 Versioning and Localization

The .NET Binary Format: XML Data Structure has no versioning mechanism. The format contains both **UTF-16** [\[RFC2781\]](#)-encoded and UTF-8 [\[RFC2279\]](#)-encoded **strings**, and their use is described in section [2](#).

## 1.7 Vendor-Extensible Fields

Records in the .NET Binary Format: XML Data Structure that contain **DictionaryString** structures use integers to represent strings. The producer and consumer of a document encoded in this format must agree on how to map these integers to strings. This specification does not prescribe how the producer and consumer agree upon or learn about this mapping. Furthermore, the format does not provide a way to encode such information. Any specification that defines this mapping **MUST** be considered a different format.

## 2 Structures

The .NET Binary Format: XML Data Structure is composed of zero or more records, each of which represents some characters in the XML document. The complete XML document represented by the format is simply the concatenation of the characters represented by each of the records. The resulting document is not necessarily a valid XML document.

Unless otherwise noted, records may appear in any order.

### 2.1 Common Definitions

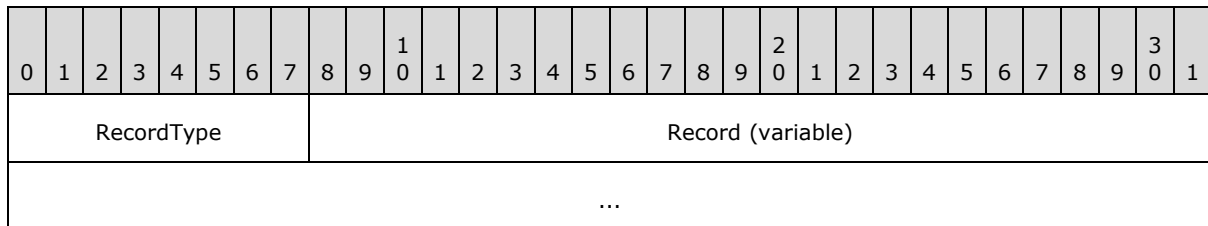
This section specifies the basic record structure and commonly used structures within those records.

Unless otherwise noted, all values MUST be encoded in **little-endian** format.

Unless otherwise noted, the alignment of a record or any of the fields in the record MUST NOT be assumed to be any particular value. The bit position diagrams are provided to indicate relative positions and sizes of fields, but do not indicate alignment.

#### 2.1.1 Record

Each record is encoded as follows.



**RecordType (1 byte):** A single byte that identifies the type of record.

**Record (variable):** Dependent upon RecordType.

The following table shows the mapping for each RecordType. The RecordType MUST be one of the values listed in this table. The format for each record is further detailed after the table.

| RecordType | Record                   |
|------------|--------------------------|
| 0x00       | Reserved                 |
| 0x01       | EndElement               |
| 0x02       | Comment                  |
| 0x03       | Array                    |
| 0x04       | ShortAttribute           |
| 0x05       | Attribute                |
| 0x06       | ShortDictionaryAttribute |
| 0x07       | DictionaryAttribute      |

| <b>RecordType</b>          | <b>Record</b>  |
|----------------------------|--|
| 0x08                       | ShortXmlnsAttribute  |
| 0x09                       | XmlnsAttribute   |
| 0x0A                       | ShortDictionaryXmlnsAttribute  |
| 0x0B                       | DictionaryXmlnsAttribute   |
| 0x0C 0x0D ... 0x24<br>0x25 | PrefixDictionaryAttributeA PrefixDictionaryAttributeB ... PrefixDictionaryAttributeY<br>PrefixDictionaryAttributeZ |
| 0x26 0x27 ... 0x3E<br>0x3F | PrefixAttributeA PrefixAttributeB ... PrefixAttributeY PrefixAttributeZ  |
| 0x40                       | ShortElement   |
| 0x41                       | Element  |
| 0x42                       | ShortDictionaryElement   |
| 0x43                       | DictionaryElement  |
| 0x44 0x45 ... 0x5C<br>0x5D | PrefixDictionaryElementA PrefixDictionaryElementB ... PrefixDictionaryElementY<br>PrefixDictionaryElementZ         |
| 0x5E 0x5F ... 0x76<br>0x77 | PrefixElementA PrefixElementB ... PrefixElementY PrefixElementZ  |
| 0x78 0x79 ... 0x7E<br>0x7F | Reserved   |
| 0x80                       | ZeroText   |
| 0x81                       | ZeroTextWithEndElement   |
| 0x82                       | OneText  |
| 0x83                       | OneTextWithEndElement  |
| 0x84                       | FalseText  |
| 0x85                       | FalseTextWithEndElement  |
| 0x86                       | TrueText   |
| 0x87                       | TrueTextWithEndElement   |
| 0x88                       | Int8Text   |
| 0x89                       | Int8TextWithEndElement   |
| 0x8A                       | Int16Text  |
| 0x8B                       | Int16TextWithEndElement  |
| 0x8C                       | Int32Text  |
| 0x8D                       | Int32TextWithEndElement  |

| <b>RecordType</b> | <b>Record</b>                |
|-------------------|------------------------------|
| 0x8E              | Int64Text                    |
| 0x8F              | Int64TextWithEndElement      |
| 0x90              | FloatText                    |
| 0x91              | FloatTextWithEndElement      |
| 0x92              | DoubleText                   |
| 0x93              | DoubleTextWithEndElement     |
| 0x94              | DecimalText                  |
| 0x95              | DecimalTextWithEndElement    |
| 0x96              | DateTimeText                 |
| 0x97              | DateTimeTextWithEndElement   |
| 0x98              | Chars8Text                   |
| 0x99              | Chars8TextWithEndElement     |
| 0x9A              | Chars16Text                  |
| 0x9B              | Chars16TextWithEndElement    |
| 0x9C              | Chars32Text                  |
| 0x9D              | Chars32TextWithEndElement    |
| 0x9E              | Bytes8Text                   |
| 0x9F              | Bytes8TextWithEndElement     |
| 0xA0              | Bytes16Text                  |
| 0xA1              | Bytes16TextWithEndElement    |
| 0xA2              | Bytes32Text                  |
| 0xA3              | Bytes32TextWithEndElement    |
| 0xA4              | StartListText                |
| 0xA5              | Reserved                     |
| 0xA6              | EndListText                  |
| 0xA7              | Reserved                     |
| 0xA8              | EmptyText                    |
| 0xA9              | EmptyTextWithEndElement      |
| 0xAA              | DictionaryText               |
| 0xAB              | DictionaryTextWithEndElement |

| RecordType                 | Record                            |
|----------------------------|-----------------------------------|
| 0xAC                       | UniqueIdText                      |
| 0xAD                       | UniqueIdTextWithEndElement        |
| 0xAE                       | TimeSpanText                      |
| 0xAF                       | TimeSpanTextWithEndElement        |
| 0xB0                       | UuidText                          |
| 0xB1                       | UuidTextWithEndElement            |
| 0xB2                       | UInt64Text                        |
| 0xB3                       | UInt64TextWithEndElement          |
| 0xB4                       | BoolText                          |
| 0xB5                       | BoolTextWithEndElement            |
| 0xB6                       | UnicodeChars8Text                 |
| 0xB7                       | UnicodeChars8Text WithEndElement  |
| 0xB8                       | UnicodeChars16Text                |
| 0xB9                       | UnicodeChars16TextWithEndElement  |
| 0xBA                       | UnicodeChars32Text                |
| 0xBB                       | UnicodeChars32TextWithEndElement  |
| 0xBC                       | QNameDictionaryText               |
| 0xBD                       | QNameDictionaryTextWithEndElement |
| 0xBE 0xBF ... 0xFE<br>0xFF | Reserved                          |

### 2.1.2 MultiByteInt31

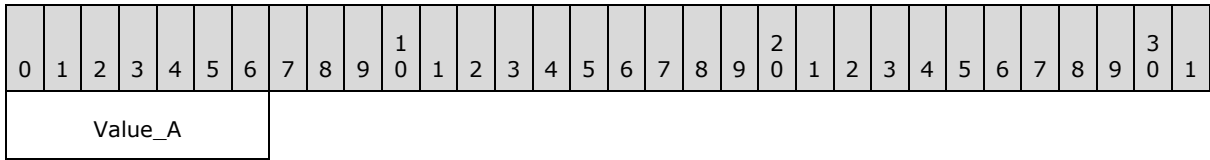
This structure describes an unsigned 31-bit integer value in a variable-length packet. The size of the number to be stored determines the size of the packet according to the following mapping.

| Unsigned integer range       | Packet size | Packet reference                         |
|------------------------------|-------------|--|
| 0x00 to 0x7F                 | 1 byte      | <a href="#">MultiByteInt31-(1 Byte)</a>  |
| 0x0080 to 0x3FFF             | 2 bytes     | <a href="#">MultiByteInt31-(2 Bytes)</a> |
| 0x004000 to 0x1FFFFF         | 3 bytes     | <a href="#">MultiByteInt31-(3 Bytes)</a> |
| 0x00200000 to 0x0FFFFFFF     | 4 bytes     | <a href="#">MultiByteInt31-(4 Bytes)</a> |
| 0x0010000000 to 0x007FFFFFFF | 5 bytes     | <a href="#">MultiByteInt31-(5 Bytes)</a> |

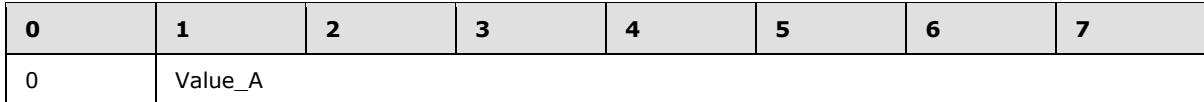
### 2.1.2.1 MultiByteInt31-(1 Byte)

The MultiByteInt31-(1 Byte) packet is used to store unsigned integer values in the range of 0x00 to 0x7F (decimal 0 to 127) inclusive.

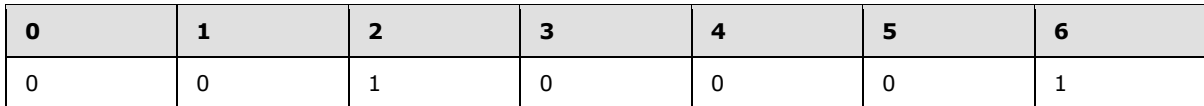
MultiByteInt31 (7 bits encoded in 1 byte)



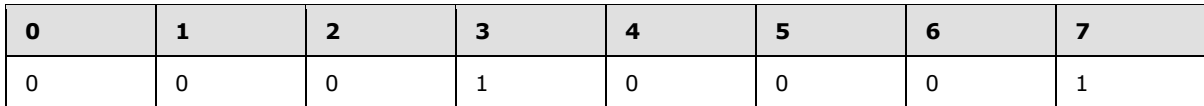
**Value\_A (7 bits):** Seven LSB of value



Example: decimal 17



Encodes as follows.



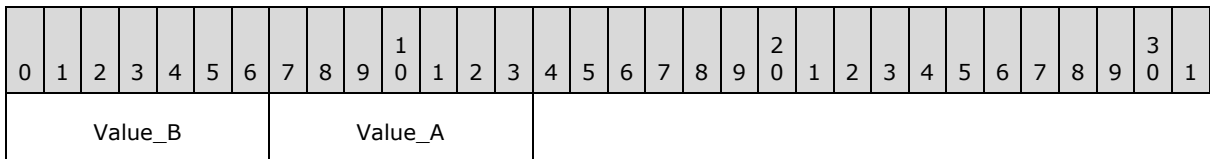
Thus, the decimal value 17 is encoded as 1 byte, as in the following example.

0x11

### 2.1.2.2 MultiByteInt31-(2 Bytes)

The MultiByteInt31-(2 Bytes) packet is used to store unsigned integers in the range of 0x0080 to 0x3FFF (decimal 128 to 16383) inclusive.

MultiByteInt31 (14 bits encoded in 2 bytes)



**Value\_B (7 bits):** Second seven LSB of value

**Value\_A (7 bits):** First seven LSB of value

| 0 | 1       | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9       | 10 | 11 | 12 | 13 | 14 | 15 |  |
|---|---------|---|---|---|---|---|---|---|---------|----|----|----|----|----|----|--|
| 0 | Value_B |   |   |   |   |   |   | 1 | Value_A |    |    |    |    |    |    |  |

Example: decimal 145

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0  | 0  | 0  | 1  |

Encodes as follows.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0  | 1  | 0  | 0  | 0  | 1  |

Thus, the decimal value 145 is encoded as 2 bytes, as in the following example.

0x91 0x01

Example: decimal 5521

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0  | 0  | 0  | 1  |

Encodes as follows.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0  | 1  | 0  | 0  | 0  | 1  |

Thus, the decimal value 5521 is encoded in 2 bytes, as in the following example.

0x91 0x2B

### 2.1.2.3 MultiByteInt31-(3 Bytes)

The MultiByteInt31-(3 Bytes) packet is used to store unsigned integers in the range of 0x004000 to 0x1FFFFFF (decimal 16384 to 2097151) inclusive.

MultiByteInt31 (21 bits encoded in 3 bytes)

|         |   |   |   |   |   |   |         |   |   |    |    |    |    |         |    |    |    |    |    |    |
|---------|---|---|---|---|---|---|---------|---|---|----|----|----|----|---------|----|----|----|----|----|----|
| 0       | 1 | 2 | 3 | 4 | 5 | 6 | 7       | 8 | 9 | 10 | 11 | 12 | 13 | 14      | 15 | 16 | 17 | 18 | 19 | 20 |
| Value_C |   |   |   |   |   |   | Value_B |   |   |    |    |    |    | Value_A |    |    |    |    |    |    |

**Value\_C (7 bits):** Third 7 LSB of value

**Value\_B (7 bits):** Second 7 LSB of value

**Value\_A (7 bits):** First 7 LSB of value

|   |         |   |   |   |   |   |   |   |         |    |    |    |    |    |    |    |         |    |    |    |    |    |    |
|---|---------|---|---|---|---|---|---|---|---------|----|----|----|----|----|----|----|---------|----|----|----|----|----|----|
| 0 | 1       | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9       | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17      | 18 | 19 | 20 | 21 | 22 | 23 |
| 0 | Value_C |   |   |   |   |   |   | 1 | Value_B |    |    |    |    |    |    | 1  | Value_A |    |    |    |    |    |    |

Example: decimal 16384

|   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Encodes as follows.

|   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Thus, the decimal value 16384 is encoded in 3 bytes, as in the following example.

0x80 0x80 0x01

### 2.1.2.4 MultiByteInt31-(4 Bytes)

The MultiByteInt31-(4 Bytes) packet is used to store unsigned integers in the range of 0x00200000 to 0xFFFFFFFF (decimal 2097152 to 268435455) inclusive.

MultiByteInt31 (28 bits encoded in 4 bytes)

|         |   |   |   |   |   |   |         |   |   |    |    |    |    |         |    |    |    |    |    |    |         |    |    |    |    |    |    |    |    |    |    |
|---------|---|---|---|---|---|---|---------|---|---|----|----|----|----|---------|----|----|----|----|----|----|---------|----|----|----|----|----|----|----|----|----|----|
| 0       | 1 | 2 | 3 | 4 | 5 | 6 | 7       | 8 | 9 | 10 | 11 | 12 | 13 | 14      | 15 | 16 | 17 | 18 | 19 | 20 | 21      | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Value D |   |   |   |   |   |   | Value C |   |   |    |    |    |    | Value B |    |    |    |    |    |    | Value A |    |    |    |    |    |    |    |    |    |    |

**Value D (7 bits):** Fourth 7 LSB of value

**Value C (7 bits):** Third 7 LSB of value

**Value B (7 bits):** Second 7 LSB of value

**Value A (7 bits):** First 7 LSB of value

|   |         |   |   |   |   |   |   |   |         |    |    |    |    |    |    |    |         |    |    |    |    |    |    |    |         |    |    |    |    |    |    |
|---|---------|---|---|---|---|---|---|---|---------|----|----|----|----|----|----|----|---------|----|----|----|----|----|----|----|---------|----|----|----|----|----|----|
| 0 | 1       | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9       | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17      | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25      | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | Value_D |   |   |   |   |   |   | 1 | Value_C |    |    |    |    |    |    | 1  | Value_B |    |    |    |    |    |    | 1  | Value_A |    |    |    |    |    |    |



Example: decimal 268435456

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Encodes as follows.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Thus, the decimal value 268435456 is encoded in 4 bytes, as in the following example.

0x80 0x80 0x80 0x01

### 2.1.2.5 MultiByteInt31-(5 Bytes)

The MultiByteInt31-(5 Bytes) packet is used to store unsigned integers in the range of 0x010000000 to 0x07FFFFFFF (decimal 268435456 to 2147483647) inclusive.

MultiByteInt31 (31 bits encoded in 5 bytes)

|         |   |   |         |   |   |   |   |   |   |         |   |   |   |   |   |   |         |   |   |   |   |   |   |         |   |   |   |   |   |   |   |   |
|---------|---|---|---------|---|---|---|---|---|---|---------|---|---|---|---|---|---|---------|---|---|---|---|---|---|---------|---|---|---|---|---|---|---|---|
| 0       | 1 | 2 | 3       | 4 | 5 | 6 | 7 | 8 | 9 | 1       | 2 | 3 | 4 | 5 | 6 | 7 | 8       | 9 | 2 | 0 | 1 | 2 | 3 | 4       | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Value_E |   |   | Value_D |   |   |   |   |   |   | Value_C |   |   |   |   |   |   | Value_B |   |   |   |   |   |   | Value_A |   |   |   |   |   |   |   |   |

**Value\_E (3 bits):** First 3 MSB of value

**Value\_D (7 bits):** Fourth 7 LSB of value

**Value\_C (7 bits):** Third 7 LSB of value

**Value\_B (7 bits):** Second 7 LSB of value

**Value\_A (7 bits):** First 7 LSB of value

Example: decimal 268435456

|   |   |   |   |         |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |         |   |   |   |   |
|---|---|---|---|---------|---|---|---|---|---|---|---|---------|---|---|---|---|---|---|---|---------|---|---|---|---|---|---|---|---------|---|---|---|---|---|---|---|---------|---|---|---|---|
| 0 | 1 | 2 | 3 | 4       | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3       | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8       | 9 | 3 | 0 | 1 | 2 | 3 | 4 | 5       | 6 | 7 | 8 | 9 |
| 0 | 0 |   |   | Value_E |   |   |   |   |   |   | 1 | Value_D |   |   |   |   |   |   | 1 | Value_C |   |   |   |   |   |   | 1 | Value_B |   |   |   |   |   |   | 1 | Value_A |   |   |   |   |

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |   |   |   |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Encodes As:

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |   |   |   |   |

Thus, the decimal value 268435456 is encoded in 5 bytes, as in the following example.

0x80 0x80 0x80 0x80 0x01

### 2.1.3 String

The String structure describes a set of characters encoded in UTF-8, as specified in [\[RFC2279\]](#).

|                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0                 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Length (variable) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...               |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Bytes (variable)  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...               |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Length (variable):** This is the length in bytes of the string when encoded in UTF-8, as specified in [\[RFC2279\]](#), and MUST be encoded using **MultiByteInt31**. For more information on MultiByteInt31 see section [2.1.2](#).

**Bytes (variable):** These are the bytes that constitute the string and MUST be encoded in UTF-8, as specified in [\[RFC2279\]](#).

For example, the string "abc" is encoded as 4 bytes.

0x03 0x61 0x62 0x63

This specification places no restrictions on the set of characters that may be encoded here.

### 2.1.4 DictionaryString

The DictionaryString structure describes a reference to a set of characters.

|                  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Value (variable) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...              |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Value (variable):** An integer value encoded using MultiByteInt31. For more information on MultiByteInt31 see section [2.1.2](#). The string that the integer refers to is determined by the producer and consumer of the document.

This specification places no restrictions on the set of characters that may be referenced.

## 2.2 Records

This section describes the format of each of the records noted earlier, and the characters they represent. The character representations of records are case sensitive and MUST use the exact casing depicted.

Records can largely be grouped into four categories:

- [Element Records](#)
- [Attribute Records](#)
- [Text Records](#)
- [Miscellaneous Records](#)

For reference, the record type is shown in hex following each record.

### 2.2.1 Element Records

This section describes the different kinds of element records. An element record is any record with a record type. See the following tables from 0x40 to 0x77 inclusive. Element records represent different kinds of elements in the XML document.

#### 2.2.1.1 ShortElement Record (0x40)

This structure represents an element without a prefix.

|                       |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-----------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Name (variable)       |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Attributes (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Name (variable):** The name of the element encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Attributes (variable):** Zero or more attribute records.

For example, if name refers to the string "element" and attributes refers to { xmlns="http://tempuri.org" }, this record is interpreted as the following characters.

```
<element_xmlns="http://tempuri.org">
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.1.2 Element Record (0x41)

This structure represents an element with a prefix.

| 0                     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-----------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Prefix (variable)     |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Name (variable)       |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Attributes (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Prefix (variable):** The prefix of the element encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Name (variable):** The name of the element encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Attributes (variable):** Zero or more attribute records.

For example, if prefix refers to the string "prefix", name refers to the string "element", and attributes refers to { xmlns:prefix="http://tempuri.org" }, this record is interpreted as the following characters.

```
<prefix:element xmlns:prefix="http://tempuri.org">
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.1.3 ShortDictionaryElement Record (0x42)

This structure represents an element without a prefix.

| 0                     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-----------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Name (variable)       |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Attributes (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Name (variable):** The name of the element encoded using DictionaryString. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Attributes (variable):** Zero or more attribute records.

For example, if name refers to the String "element" and attributes refers to { xmlns="http://tempuri.org" }, this record is interpreted as the following characters.

```
<element_xmlns="http://tempuri.org">
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

#### 2.2.1.4 DictionaryElement Record (0x43)

This structure represents an element with a prefix.

| 0                     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-----------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Prefix (variable)     |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Name (variable)       |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Attributes (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Prefix (variable):** The prefix of the element encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Name (variable):** The name of the element encoded using Dictionary. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Attributes (variable):** Zero or more attribute records.

For example, if prefix refers to the string "prefix", name refers to the string "element", and attributes refers to { xmlns:prefix="http://tempuri.org" }, this record is interpreted as the following characters.

```
<prefix:element_xmlns:prefix="http://tempuri.org">
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

#### 2.2.1.5 PrefixDictionaryElement[A-Z] Record (0x44-0x5D)

This structure represents an element with a single lowercase letter prefix.

|                       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-----------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0                     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Name (variable)       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Attributes (variable) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Name (variable):** The name of the element encoded using DictionaryString. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Attributes (variable):** Zero or more attribute records.

The prefix for this attribute is determined by the record type.

For example, if the record type is PrefixDictionaryElementB, name refers to the string "element", and attributes refers to { xmlns:b="http://tempuri.org" }, this record is interpreted as the following characters.

```
<b:element_xmlns:b="http://tempuri.org">
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.1.6 PrefixElement[A-Z] Record (0x5E-0x77)

This structure represents an element with a single lowercase letter prefix.

|                       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-----------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0                     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Name (variable)       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Attributes (variable) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Name (variable):** The name of the element encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Attributes (variable):** Zero or more attribute records.

The prefix for this attribute is determined by the record type.

For example, if the record type is PrefixElementB, name refers to the string "element", and attributes refers to { xmlns:b="http://tempuri.org" }, this record is interpreted as the following characters.

<b:element\_xmlns:b="http://tempuri.org">

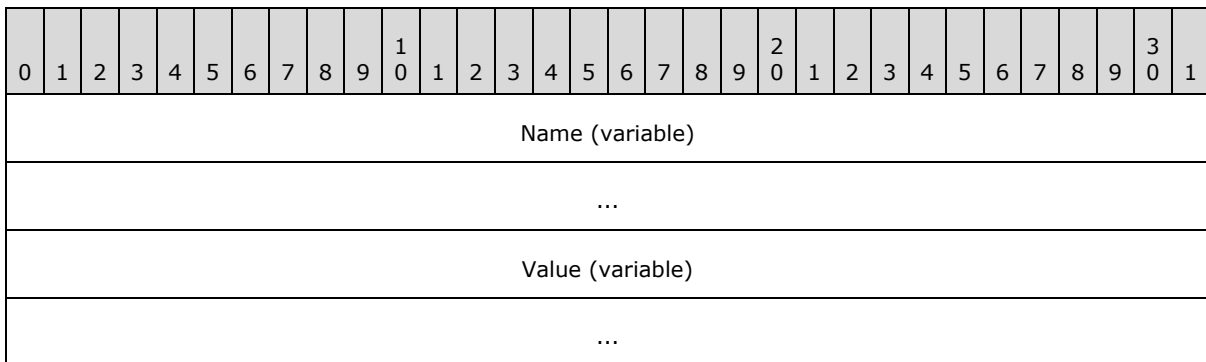
Note that the underscore is intended to represent a single ASCII white-space character (32).

## 2.2.2 Attribute Records

This section describes the different kinds of attribute records. An attribute record is any record with a record type (see Table 1) from 0x04 to 0x3F inclusive. An attribute record MUST follow another attribute record or an element record. Attribute records represent different kinds of attributes in the XML document.

### 2.2.2.1 ShortAttribute Record (0x04)

This structure represents an attribute without a prefix.



**Name (variable):** The name of the attribute encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using a text record.

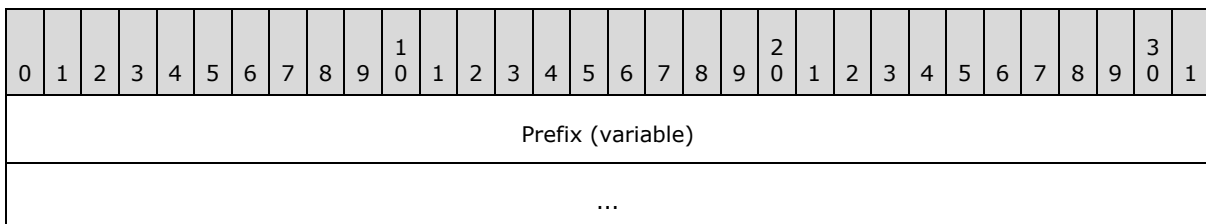
For example, if name refers to the string "attr" and value refers to the text "value", this record is interpreted as the following characters.

```
_attr="value"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.2.2 Attribute Record (0x05)

This structure represents an attribute with a prefix.



|                  |
|------------------|
| Name (variable)  |
| ...              |
| Value (variable) |
| ...              |

**Prefix (variable):** The prefix of the attribute encoded using String. The length of this String MUST be nonzero. The prefix MUST NOT be "xmlns".

**Name (variable):** The name of the attribute encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using a single text record ([Text Records](#)).

For example, if prefix refers to the string "prefix", and the name refers to the string "attr", and value refers to the text "value", this record is interpreted as the following characters.

```
_prefix:attr="value"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.2.3 ShortDictionaryAttribute Record (0x06)

This structure represents an attribute without a prefix.

|                  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Name (variable)  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...              |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Value (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...              |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Name (variable):** The name of the attribute encoded using DictionaryString. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using a text record.

For example if name refers to the string "attr" and value refers to the text "value", this record is interpreted as the following attribute.

```
_attr="value"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).



### 2.2.2.4 DictionaryAttribute Record (0x07)

This structure represents an attribute with a prefix.

|                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Prefix (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...               |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Name (variable)   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...               |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Value (variable)  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...               |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Prefix (variable):** The prefix of the attribute encoded using String. The length of this String MUST be nonzero. The prefix MUST NOT be "xmlns".

**Name (variable):** The name of the attribute encoded using DictionaryString. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using a text record.

For example, if prefix refers to the string "prefix", name refers to the string "attr", and value refers to the text "value", this record is interpreted as the following characters.

```
_prefix:attr="value"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.2.5 ShortXmlnsAttribute Record (0x08)

This structure represents an xmlns attribute without a prefix.

|                  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Value (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...              |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Value (variable):** The value of the attribute encoded using String.

For example, if value refers to the string "http://tempuri.org", this record is interpreted as the following characters.

```
_xmlns="http://tempuri.org"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.2.6 XmlnsAttribute Record (0x09)

This structure represents an xmlns attribute with a prefix.

|                   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Prefix (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...               |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Value (variable)  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...               |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Prefix (variable):** The prefix of the attribute encoded using String. The length of this String MUST be nonzero. The prefix MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using String.

For example, if prefix refers to the string "ENV" and value refers to the string "http://tempuri.org", this record is interpreted as the following characters.

```
_xmlns:ENV="http://tempuri.org"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.2.7 ShortDictionaryXmlnsAttribute Record (0x0A)

This structure represents an xmlns attribute without a prefix.

|                  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Value (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...              |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Value (variable):** The value of the attribute encoded using DictionaryString.

For example, if value refers to the text "value", this record is interpreted as the following characters.

```
_xmlns="value"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.2.8 DictionaryXmIsAttribute Record (0x0B)

This structure represents an xmlns attribute with a prefix.

|                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0                 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Prefix (variable) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...               |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Value (variable)  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...               |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Prefix (variable):** The prefix of the attribute encoded using String. The length of this String MUST be nonzero. The prefix MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using DictionaryString.

For example, if prefix refers to the string "ENV" and value refers to the string "http://tempuri.org", this record is interpreted as the following characters.

```
_xmlns:ENV="http://tempuri.org"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.2.9 PrefixDictionaryAttribute[A-Z] Records (0x0C-0x25)

This structure represents an attribute with a single lowercase letter prefix.

|                  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| Name (variable)  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...              |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Value (variable) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...              |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Name (variable):** The name of the attribute encoded using DictionaryString. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using a text record.

### 2.2.2.10 PrefixAttribute[A-Z] Records (0x26-0x3F)

This structure represents an attribute with a single lowercase letter prefix.

|                  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| Name (variable)  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...              |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Value (variable) |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ...              |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Name (variable):** The name of the attribute encoded using String. The length of this String MUST be nonzero. The name MUST NOT be "xmlns".

**Value (variable):** The value of the attribute encoded using text record.

The prefix for this attribute is determined by the record type.

For example, if the record type is PrefixAttributeX, name refers to the string "attr", and value refers to the text "value", this record is interpreted as the following characters.

```
_x:attr="value"
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.3 Text Records

This section describes the different kinds of text records. A text record is any record with a record type (see Table 1) from 0x80 to 0xBD inclusive. Text records are used to represent the attribute or element content of the XML document.

#### 2.2.3.1 ZeroText Record (0x80)

This structure represents attribute or element content and MUST be interpreted as representing the following characters.

```
0
```

There are no additional fields for this record.

#### 2.2.3.2 OneText Record (0x82)

This structure represents attribute or element content and MUST be interpreted as representing the following characters.

```
1
```

There are no additional fields for this record.

### 2.2.3.3 FalseText Record (0x84)

This structure represents attribute or element content and MUST be interpreted as representing the following characters.

```
false
```

There are no additional fields for this record.

### 2.2.3.4 TrueText Record (0x86)

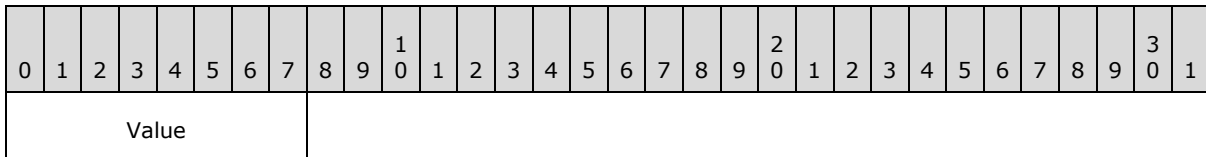
This structure represents attribute or element content and MUST be interpreted as representing the following characters.

```
true
```

There are no additional fields for this record.

### 2.2.3.5 Int8Text Record (0x88)

This structure represents attribute or element content.



**Value (1 byte):** The signed 8-bit integer value.

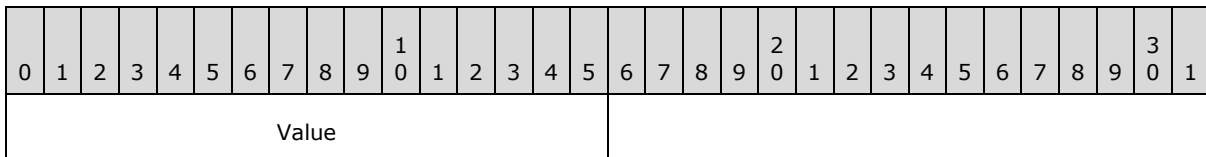
This structure MUST be interpreted as representing the characters formed by converting the value to base 10. The characters MUST be preceded by a minus sign "-" if the value is negative. There MUST NOT be any leading zeroes or decimal point.

For example, if value is 0x80, this is interpreted as the following characters.

```
-128
```

### 2.2.3.6 Int16Text Record (0x8A)

This structure represents attribute or element content.



**Value (2 bytes):** The signed 16-bit integer value.

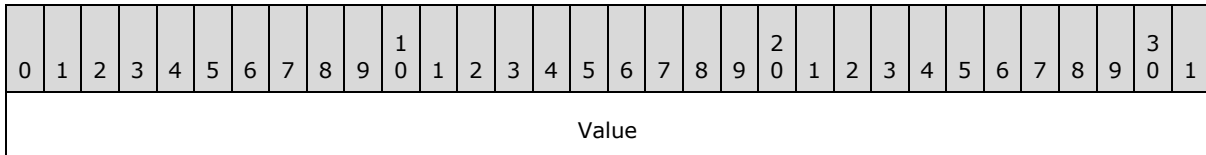
This structure MUST be interpreted as representing the characters formed by converting the value to base 10. The characters MUST be preceded by a minus sign "-" if the value is negative. There MUST NOT be any leading zeroes or decimal point.

For example, if value is 0x8000, this is interpreted as the following characters.

-32768

### 2.2.3.7 Int32Text Record (0x8C)

This structure represents attribute or element content.



**Value (4 bytes):** The signed 32-bit integer value.

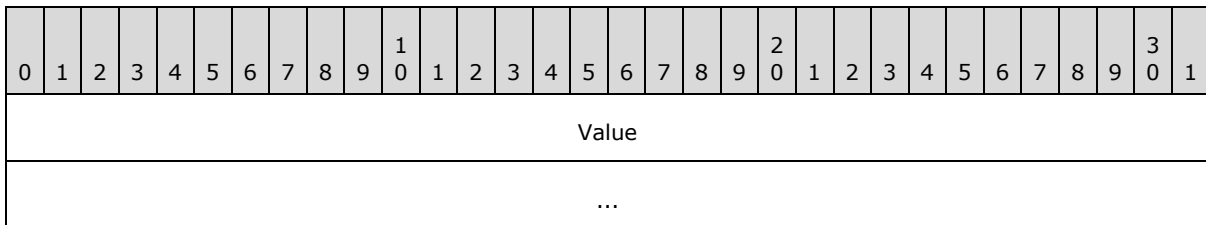
This structure MUST be interpreted as representing the characters formed by converting the value to base 10. The characters MUST be preceded by a minus sign "-" if the value is negative. There MUST NOT be any leading zeroes or decimal point.

For example, if value is 0x80000000, this is interpreted as the following characters.

-2147483648

### 2.2.3.8 Int64Text Record (0x8E)

This structure represents attribute or element content.



**Value (8 bytes):** The signed 64-bit integer value.

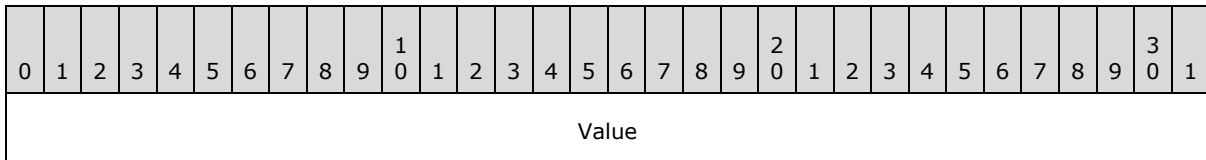
This structure MUST be interpreted as representing the characters formed by converting the value to base 10. The characters MUST be preceded by a minus sign "-" if the value is negative. There MUST NOT be any leading zeroes or decimal point.

For example, if value is 0x8000000000000000, this is interpreted as the following characters.

-9223372036854775808

### 2.2.3.9 FloatText Record (0x90)

This structure represents attribute or element content.



**Value (4 bytes):** The 32-bit single precision floating point value as described in [\[IEEE754\]](#).

This structure MUST be interpreted as representing the characters formed by converting the value to base 10. The period "." MUST be used as the decimal point only if a fractional component exists. The least number of digits that exactly reproduces the IEEE representation MUST be used. There MUST NOT be any unnecessary leading or trailing zeroes, except when a decimal point is the first character, in which case a single zero "0" MUST precede the decimal point. Exponential notation MUST be used when the position of the decimal point is outside the range of significant digits. When exponential notation is used, the character "E" MUST be used, and MUST be followed by a plus sign "+" or minus sign "-", and MUST be followed by the magnitude of the exponent.

Furthermore, special values have special characters that MUST be used.

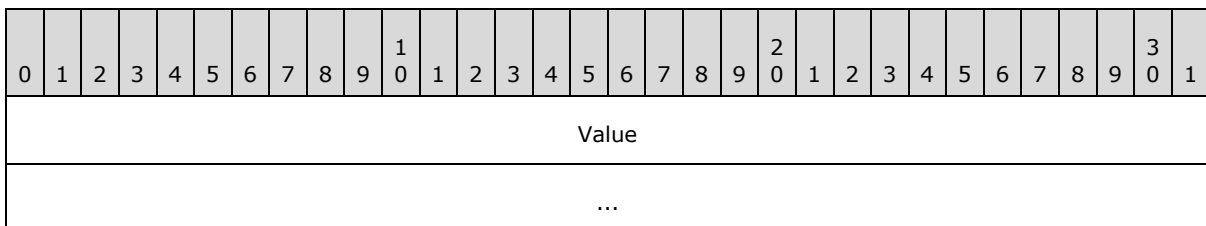
| Value             | Characters |
|-------------------|------------|
| Infinity          | INF        |
| Negative infinity | -INF       |
| Nan               | NaN        |
| Negative zero     | -0         |

For example, if value is 0x3F8CCCCD, this is interpreted as the following characters.

1.1

### 2.2.3.10 DoubleText Record (0x92)

This structure represents attribute or element content.



**Value (8 bytes):** The 64-bit single precision floating point value as specified in [\[IEEE754\]](#).

This structure MUST be interpreted as representing the characters formed by converting the value to base 10. The period "." MUST be used as the decimal point only if a fractional component exists. The least number of digits that exactly reproduces the IEEE representation MUST be used. There MUST

NOT be any unnecessary leading or trailing zeroes, except when a decimal point is the first character, in which case a single zero "0" MUST precede the decimal point. Exponential notation MUST be used when the position of the decimal point is outside the range of significant digits. When exponential notation is used, the character "E" MUST be used, and MUST be followed by a plus sign "+" or minus sign "-", and MUST be followed by the magnitude of the exponent.

Furthermore, special values have special characters that MUST be used.

| Value             | Characters |
|-------------------|------------|
| Infinity          | INF        |
| Negative infinity | -INF       |
| Nan               | NaN        |
| Negative zero     | -0         |

For example, if value is 0x4005BF0A8B145774, this is interpreted as the following characters.

2.7182818284590451

### 2.2.3.11 DecimalText Record (0x94)

This structure represents attribute or element content.

| 0     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Value |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Value (16 bytes):** The decimal value encoded in 16-bytes as specified in [\[MS-OAUT\]](#) section 2.2.26. See also [\[IEEE854\]](#).

This structure MUST be interpreted as representing the characters formed by converting the value to base 10. The period "." MUST be used as the decimal point only if a fractional component exists. The least number of digits that exactly reproduces the IEEE representation MUST be used. There MUST NOT be any unnecessary leading or trailing zeros, except when a decimal point is the first character, in which case a single zero "0" MUST precede the decimal point.

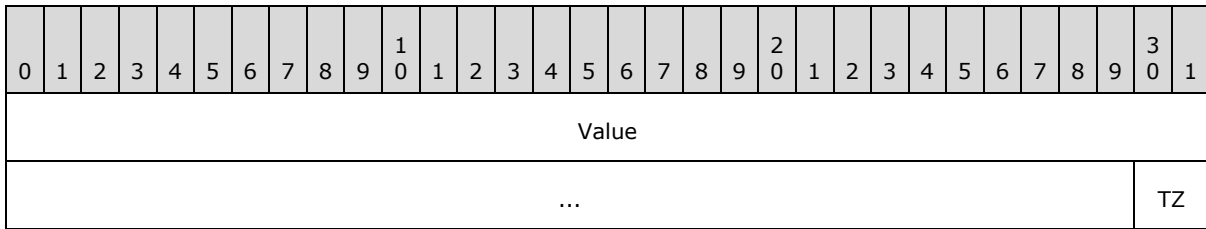
For example, if value is 0x0000000004F2D800000000000060000, this is interpreted as the following characters.

5.123456



### 2.2.3.12 DateTimeText Record (0x96)

This structure represents attribute or element content.



**Value (62 bits):** The 62-bit unsigned integer value that specifies the number of 100 nanoseconds that had elapsed since 12:00:00, January 1, 0001. The value can represent time instants in a granularity of 100 nanoseconds until 23:59:59.9999999, December 31, 9999. The value MUST be less than the decimal value 3155378976000000000.

**TZ (2 bits):** A two-bit unsigned integer that contains TimeZone information. This MUST be 0, 1, or 2.

This structure MUST be interpreted as representing the characters formed by converting the value to a date.

If the hour, minutes, seconds, and fraction of second parts are zero, the date MUST be interpreted as the following characters.

`yyyy-MM-dd`

Otherwise, if the fraction of a second part is zero, the date MUST be interpreted as the following characters.

`yyyy-MM-ddTHH:mm:ss`

Otherwise, the date MUST be interpreted as the following characters.

`yyyy-MM-ddTHH:mm:ss.ffffff`

where:

- yyyy is the four-digit representation of the year.
- MM is the two-digit representation of the month starting at "01".
- dd is the two-digit representation of the day of the month starting at "01".
- HH is the two-digit representation of the hour of the day starting at "00".
- mm is the two-digit representation of the minute of the hour starting at "00".
- ss is the two-digit representation of the second of the minute starting at "00".

- fffffff is up to seven digits representing the fraction of the second. There MUST be no trailing zeros.

All other characters are included as shown.

If TZ is one, then the time is in **UTC (Coordinated Universal Time)**, and the date MUST be interpreted as having a trailing character "Z".

If TZ is two, then the time is a local time, and the date MUST be interpreted as having additional characters that indicate the UTC offset. The UTC offset MUST be the time zone offset in which the document is being decoded.

If the UTC offset is positive, the date MUST be interpreted as having the following additional characters.

+HH:mm

If the UTC offset is negative, the date MUST be interpreted as having the following additional characters.

-HH:mm

where:

- HH is the two-digit representation of the absolute value of the hour UTC offset starting at "00".
- mm is the two-digit representation of the absolute value of the minute UTC offset starting at "00".

All other characters are included as shown.

If TZ is zero, the time is not specified as either UTC or a local time and nothing further is added.

The interpreted format of a DateTimeText record is [\[ISO-8601\]](#) compliant.

### 2.2.3.13 Chars8Text Record (0x98)

This structure represents attribute or element content.

|        |   |   |   |   |   |   |   |   |   |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |
|--------|---|---|---|---|---|---|---|---|---|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|
| 0      | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10               | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |  |  |  |  |  |
| Length |   |   |   |   |   |   |   |   |   | Bytes (variable) |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |
| ...    |   |   |   |   |   |   |   |   |   |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |

**Length (1 byte):** This is the length in bytes of the UTF-8 [\[RFC2279\]](#)-encoded string and is represented as [UINT8](#).

**Bytes (variable):** The string encoded as UTF-8 [\[RFC2279\]](#) bytes.

For example, if length is 3 and bytes = { 0x41, 0x42, 0x43 }, this record is interpreted as the following characters.

abc

UTF-8 [\[RFC2279\]](#)-encoded sequences MUST be fully formed. There MUST not be any partial UTF-8 [\[RFC2279\]](#) sequences within a record. UTF-8 [\[RFC2279\]](#) sequences that expand to a low surrogate character MUST be paired with a high surrogate character. (For more information on surrogate characters, see [\[UNICODE\]](#).)

### 2.2.3.13.1 Character Escaping

Characters MUST be interpreted as minimally escaped. This means that a character MUST be interpreted as escaped only if it is required to be escaped for the character to be legal at this point in the XML document. Characters considered illegal by XML MUST be considered escaped.

If a character must be interpreted as escaped and it is one of the characters in the first column of the following table, it MUST be interpreted as the characters in the second column.

| Character | Interpret as |
|-----------|--------------|
| "         | &quot;       |
| &         | &amp;        |
| <         | &lt;         |
| >         | &gt;         |
| '         | &apos;       |

Otherwise if a character does not fall within the legal character ranges defined in XML, the character MUST be interpreted as the following characters.

&#digits;

where digits is the value of the character expressed in base 10 characters. There MUST NOT be any unnecessary leading zeros in this representation.

For example, if length is 6, and bytes = { 0x22, 0x26, 0x3C, 0x3E, 0x27, 0x00 }, and this record is within an element, this record is interpreted as the following characters.

"&amp;&lt;&gt;'&#0;

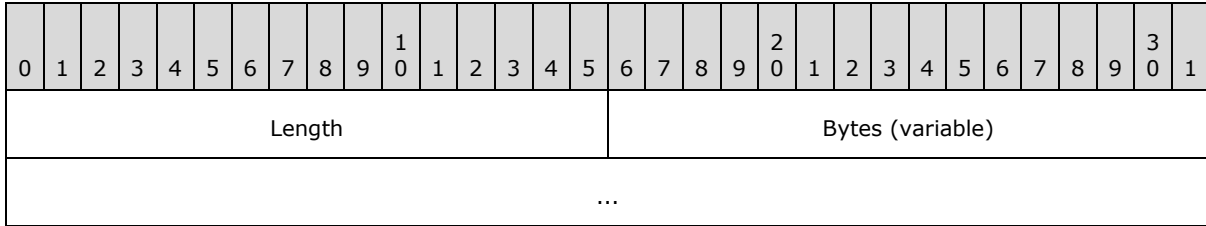
The ampersand (&), less than sign (<), and greater than sign (>) are required by XML to be escaped in element content; quotation marks (") and single quotation marks (') are not required to be escaped. The zero (0) is invalid in XML, but MUST be interpreted as appearing in its escaped form.

If the same record appeared as an attribute, this record is interpreted as the following characters.

&quot;&amp;&lt;>'&#0;

### 2.2.3.14 Chars16Text Record (0x9A)

This structure represents attribute or element content.



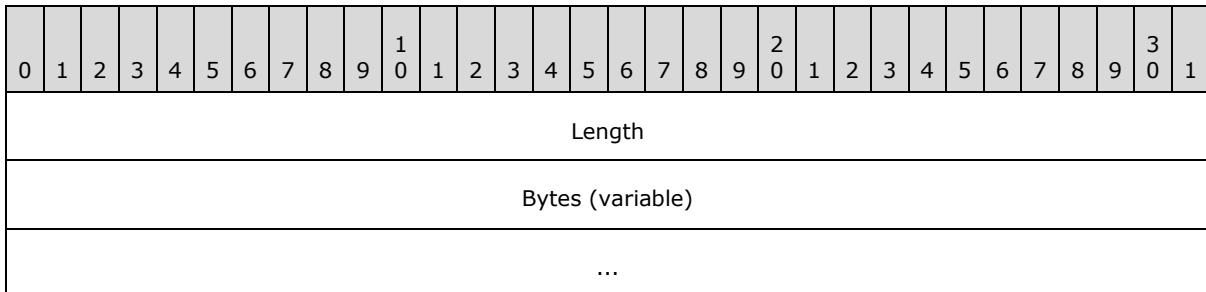
**Length (2 bytes):** This is the length in bytes of the UTF-8 [\[RFC2279\]](#)-encoded string and is represented as [UINT16](#).

**Bytes (variable):** The string encoded as UTF-8 [\[RFC2279\]](#) bytes.

See [Chars8Text Record](#) for examples.

### 2.2.3.15 Chars32Text Record (0x9C)

This structure represents attribute or element content.



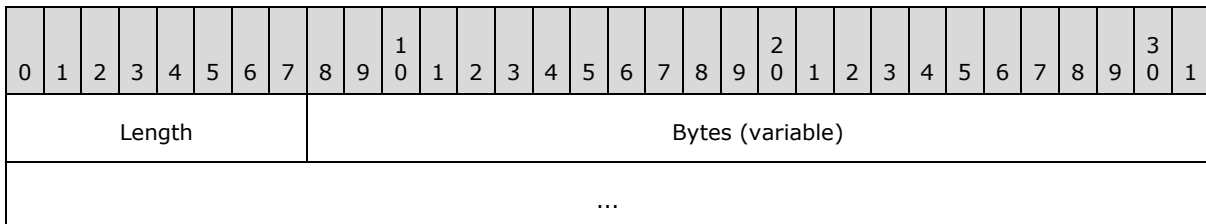
**Length (4 bytes):** This is the length in bytes of the string when encoded in UTF-8, as specified in [\[RFC2279\]](#), and is represented as [INT32](#). The value of Length MUST be positive.

**Bytes (variable):** The string encoded as UTF-8 [\[RFC2279\]](#) bytes.

See [Chars8Text Record](#) for examples.

### 2.2.3.16 Bytes8Text Record (0x9E)

This structure represents attribute or element content.



**Length (1 byte):** This is the length, in bytes, of the binary data and is represented as [UINT8](#).

**Bytes (variable):** The binary data.

This record **MUST** be interpreted as the characters obtained by encoding the bytes in **base64** as specified in [RFC3548](#).

For example, if length is 3 and bytes = { 0x01, 0x02, 0x03 }, this record is interpreted as the following characters.

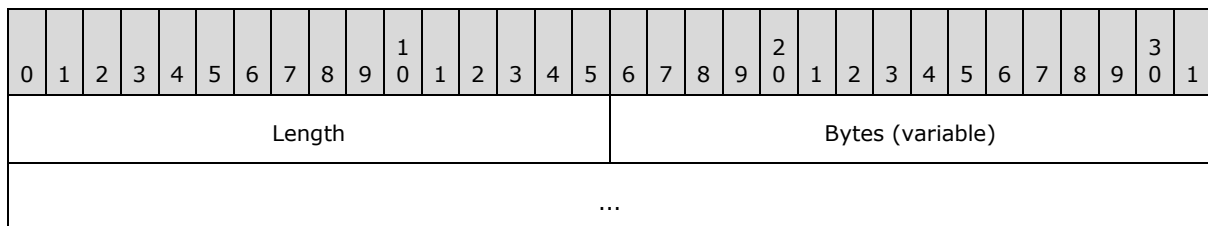
AQID

If length is 1 and bytes = { 0xFF }, this record **MUST** be interpreted as the following characters.

/w==

### 2.2.3.17 Bytes16Text Record (0xA0)

This structure represents attribute or element content.



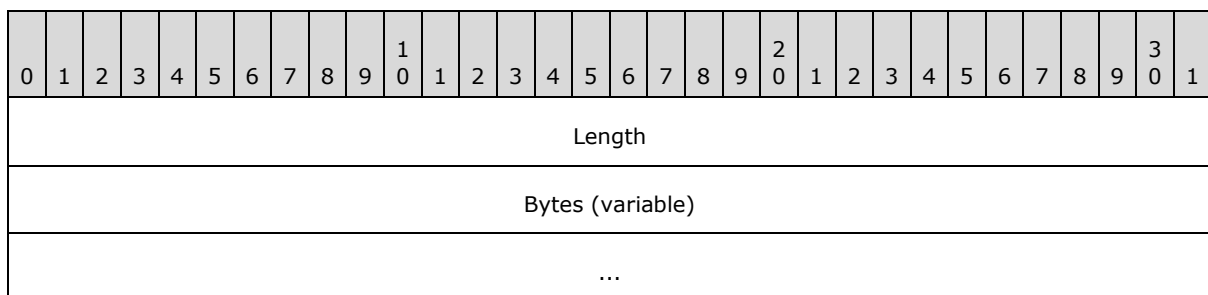
**Length (2 bytes):** This is the length in bytes of the binary data and is represented as [UINT16](#).

**Bytes (variable):** The binary data.

See [Bytes8Text Record](#) for examples.

### 2.2.3.18 Bytes32Text Record (0xA2)

This structure represents attribute or element content.



**Length (4 bytes):** This is the length in bytes of the binary data and is represented as [INT32](#).

The value of Length **MUST** be positive.

**Bytes (variable):** The binary data.

See [Bytes8Text Record](#) for examples.

### 2.2.3.19 StartListText / EndListText Records (0xA4, 0xA6)

This structure represents attribute or element content. These records identify the start and end of a list of text records separated by a single whitespace character. They have no additional fields. The records that they bracket MUST be text records and MUST NOT contain a StartListText or EndListText record. An EndListText record MUST have a corresponding StartListText record.

For example, this sequence of records

```
StartListText
TrueText
FalseText
ZeroText
OneText
EndListText
```

is interpreted as the following characters.

```
true_false_0_1
```

Note that the underscore is intended to represent a single ASCII white-space character (32).

### 2.2.3.20 EmptyText Record (0xA8)

This structure represents a zero-length string. It has no additional fields. It MUST be interpreted as no characters.

### 2.2.3.21 DictionaryText Record (0xAA)

This structure represents attribute or element content.

|                  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Value (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...              |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Value (variable):** The value of the string encoded using DictionaryString.

For example, if value refers to the string "hello", this record is interpreted as the following characters.

```
hello
```

See [Character Escaping](#) for notes on escaping of characters.

### 2.2.3.22 UniqueIdText Record (0xAC)

This structure represents attribute or element content.

|         |   |   |   |   |   |   |   |         |   |    |    |    |    |    |    |         |    |    |    |    |    |    |    |         |    |    |    |    |    |    |    |
|---------|---|---|---|---|---|---|---|---------|---|----|----|----|----|----|----|---------|----|----|----|----|----|----|----|---------|----|----|----|----|----|----|----|
| 0       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8       | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16      | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24      | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Data1   |   |   |   |   |   |   |   |         |   |    |    |    |    |    |    |         |    |    |    |    |    |    |    |         |    |    |    |    |    |    |    |
| Data2   |   |   |   |   |   |   |   |         |   |    |    |    |    |    |    | Data3   |    |    |    |    |    |    |    |         |    |    |    |    |    |    |    |
| Data4_1 |   |   |   |   |   |   |   | Data4_2 |   |    |    |    |    |    |    | Data4_3 |    |    |    |    |    |    |    | Data4_4 |    |    |    |    |    |    |    |
| Data4_5 |   |   |   |   |   |   |   | Data4_6 |   |    |    |    |    |    |    | Data4_7 |    |    |    |    |    |    |    | Data4_8 |    |    |    |    |    |    |    |

**Data1 (4 bytes):** The first 4 bytes of the **UUID**. For more information see [\[RFC4122\]](#).

**Data2 (2 bytes):** The 5th and 6th bytes of the UUID. For more information see [\[RFC4122\]](#).

**Data3 (2 bytes):** The 7th and 8th bytes of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_1 (1 byte):** The 9th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_2 (1 byte):** The 10th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_3 (1 byte):** The 11th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_4 (1 byte):** The 12th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_5 (1 byte):** The 13th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_6 (1 byte):** The 14th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_7 (1 byte):** The 15th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_8 (1 byte):** The 16th byte of the UUID. For more information see [\[RFC4122\]](#).

This record MUST be interpreted as the characters representing the UUID prefixed by the characters "urn:uuid:". The characters in the UUID MUST use lowercase. For example, if Data1 = 0x33221100, Data2 = 0x5544, Data3 = 0x7766, and Data4 = { 0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff }, this record is interpreted as the following characters.

```
urn:uuid:33221100-5544-7766-8899-aabbccddeeff
```

### 2.2.3.23 TimeSpanText Record (0xAE)

This structure represents attribute or element content.

|       |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Value |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

...

**Value (8 bytes):** A 64-bit signed integer value that specifies a duration in 100 nanosecond units. The values range from -10675199 days, 2 hours, 48 minutes, and 05.4775808 seconds to 10675199 days, 2 hours, 48 minutes, and 05.4775807 seconds.

This structure **MUST** be interpreted as representing the characters formed by converting the value as follows:

If the day part is non-zero and the fraction of a second part is zero, then the time **MUST** be interpreted as the following characters.

```
DDDDDDDD.HH:mm:ss
```

Otherwise, if the day part is non-zero and the fraction of a second part is non-zero, the time **MUST** be interpreted as the following characters.

```
DDDDDDDD.HH:mm:ss.fffffff
```

Otherwise, if the day part is zero and the fraction of a second part is zero, then the time **MUST** be interpreted as the following characters.

```
HH:mm:ss
```

Otherwise, the time **MUST** be interpreted as the following characters.

```
HH:mm:ss.fffffff
```

where:

- DDDDDDDD is up to eight digits representing the number of days.
- HH is the two-digit representation of the hour of the day starting at "00".
- mm is the two-digit representation of the minute of the hour starting at "00".
- ss is the two-digit representation of the second of the hour starting at "00".
- fffffff is up to seven digits representing the fraction of the second. There **MUST** be no trailing zeros.

All other characters are included as shown.

#### **2.2.3.24 UuidText Record (0xB0)**

This structure represents attribute or element content.



|         |   |   |   |   |   |   |   |         |   |    |    |    |    |    |    |         |    |    |    |    |    |    |    |         |    |    |    |    |    |    |    |
|---------|---|---|---|---|---|---|---|---------|---|----|----|----|----|----|----|---------|----|----|----|----|----|----|----|---------|----|----|----|----|----|----|----|
| 0       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8       | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16      | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24      | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Data1   |   |   |   |   |   |   |   |         |   |    |    |    |    |    |    |         |    |    |    |    |    |    |    |         |    |    |    |    |    |    |    |
| Data2   |   |   |   |   |   |   |   |         |   |    |    |    |    |    |    | Data3   |    |    |    |    |    |    |    |         |    |    |    |    |    |    |    |
| Data4_1 |   |   |   |   |   |   |   | Data4_2 |   |    |    |    |    |    |    | Data4_3 |    |    |    |    |    |    |    | Data4_4 |    |    |    |    |    |    |    |
| Data4_5 |   |   |   |   |   |   |   | Data4_6 |   |    |    |    |    |    |    | Data4_7 |    |    |    |    |    |    |    | Data4_8 |    |    |    |    |    |    |    |

**Data1 (4 bytes):** The first 4 bytes of the UUID. For more information see [\[RFC4122\]](#).

**Data2 (2 bytes):** The first 5th and 6th bytes of the UUID. For more information see [\[RFC4122\]](#).

**Data3 (2 bytes):** The first 7th and 8th bytes of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_1 (1 byte):** The 9th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_2 (1 byte):** The 10th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_3 (1 byte):** The 11th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_4 (1 byte):** The 12th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_5 (1 byte):** The 13th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_6 (1 byte):** The 14th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_7 (1 byte):** The 15th byte of the UUID. For more information see [\[RFC4122\]](#).

**Data4\_8 (1 byte):** The 16th byte of the UUID. For more information see [\[RFC4122\]](#).

This record MUST be interpreted as the characters representing the UUID. The characters in the UUID MUST use lowercase. For example, if Data1 = 0x33221100, Data2 = 0x5544, Data3 = 0x7766, and Data4 = { 0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff }, this record is interpreted as the following characters.

```
33221100-5544-7766-8899-aabbccdeeff
```

Note that this record differs from the UniqueIdText record only by the absence of the characters "urn:uuid:".

### 2.2.3.25 UInt64Text Record (0xB2)

This structure represents attribute or element content.

|       |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Value |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

...

**Value (8 bytes):** The unsigned 64-bit integer value.

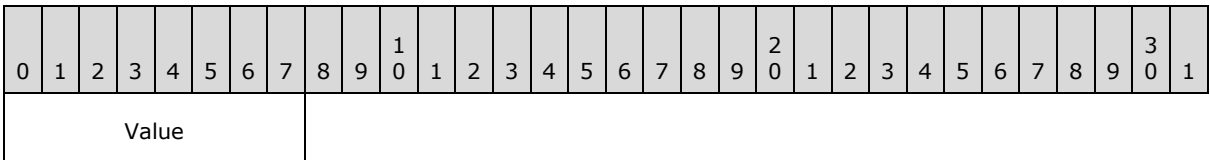
This structure MUST be interpreted as representing the characters formed by converting the value to base 10. There MUST NOT be any leading zeroes or decimal point.

For example, if value is 0xFFFFFFFFFFFFFFFF, this is interpreted as the following characters.

18446744073709551615

### 2.2.3.26 BoolText Record (0xB4)

This structure represents attribute or element content.



**Value (1 byte):** The Boolean value. This value MUST be 0 or 1.

If the value is 0, this record MUST be interpreted as the following characters.

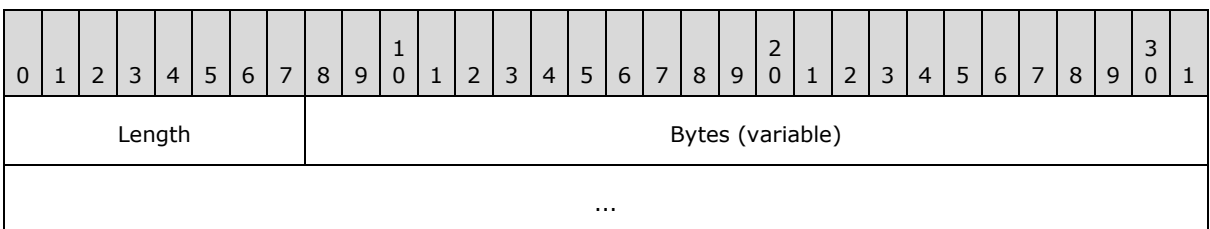
false

If value is 1, this record MUST be interpreted as the following characters.

true

### 2.2.3.27 UnicodeChars8Text Record (0xB6)

This structure represents attribute or element content.



**Length (1 byte):** The length in bytes of the UTF-16 [\[RFC2781\]](#)-encoded string.

**Bytes (variable):** The string encoded as UTF-16 [\[RFC2781\]](#) bytes.

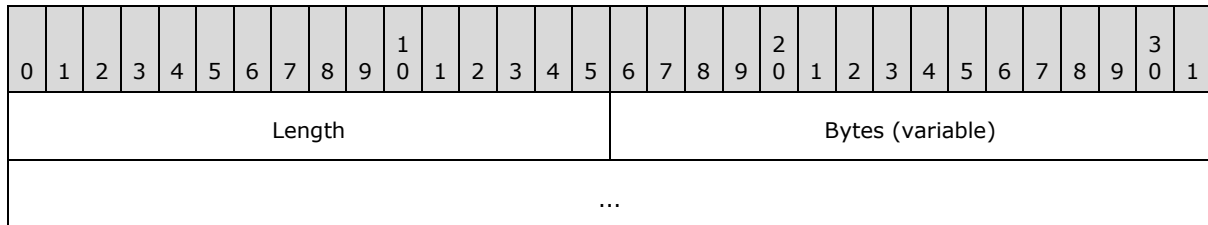
For example, if the length is 6 and bytes = { 0x41, 0x00, 0x42, 0x00, 0x43, 0x00 }, this record is interpreted as the following characters.

abc

See [Chars8Text Record](#) for notes on escaping of characters.

### 2.2.3.28 UnicodeChars16Text Record (0xB8)

This structure represents attribute or element content.



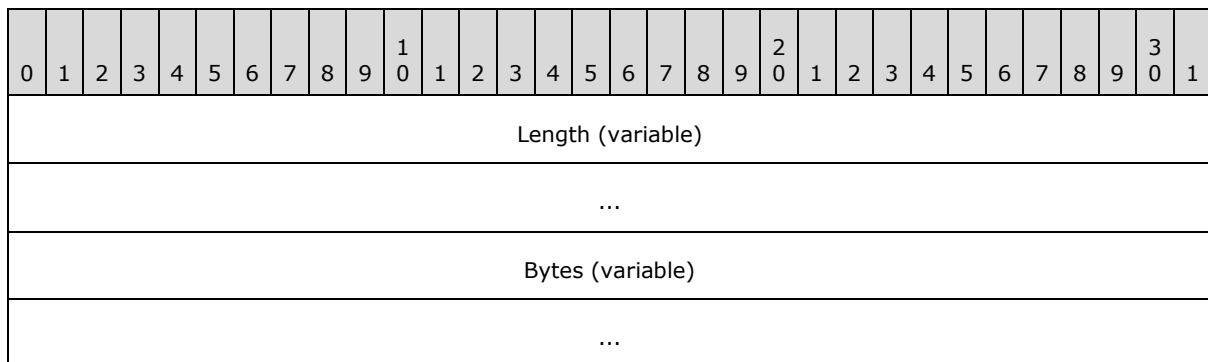
**Length (2 bytes):** The length in bytes of the UTF-16 [\[RFC2781\]](#) encoded string.

**Bytes (variable):** The string encoded as UTF-16 [\[RFC2781\]](#) bytes.

See [UnicodeChars8Text Record](#) for examples.

### 2.2.3.29 UnicodeChars32TextRecord(0xBA)

This structure represents attribute or element content.



**Length (variable):** This is the length in bytes of the string when encoded in UTF-16, as specified in [\[RFC2781\]](#), and MUST be encoded using MultiByteInt31.

**Bytes (variable):** The string encoded as UTF-16 [\[RFC2781\]](#) bytes.

See [UnicodeChars8Text Record](#) for examples.

### 2.2.3.30 QNameDictionaryTextRecord(0xBC)

This structure represents attribute or element content.

|        |   |   |   |   |   |   |   |   |   |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0      | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| Prefix |   |   |   |   |   |   |   |   |   | Name |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Prefix (1 byte):** A value from 0 to 25 inclusive, indicating a single lowercase prefix character.

**Name (3 bytes):** The name encoded as a DictionaryString.

For example, if prefix is 1 and name refers to the string "name", this record is interpreted as the following characters.

```
b:name
```

### 2.2.3.31 \*TextWithEndElement Records

These records are a simple optimization intended to reduce the size of the document.

XML of the form

```
<value>123</value>
```

can be represented by three records in the following format.

```
ShortElement(name="value")
Chars8Text(value="123")
EndElement
```

By marking the Text record to indicate that an EndElement follows, the number of records can be reduced.

```
ShortElement(name="value")
Chars8TextWithEndElement(value="123")
```

Any record with the name in the form \*TextWithEndElement MUST be interpreted as a sequence of two records: A \*Text record followed by an EndElement record.

For example, the Int32TextWithEndElement record is interpreted as an Int32TextRecord followed by an EndElement record and must behave identically.

These records MUST NOT be used inside Attribute records.

## 2.3 Miscellaneous Records

This section lists the few remaining records that are not element, attribute, or text records.

### 2.3.1 EndElement Record (0x01)

This structure represents an end element. There are no additional fields for this record beyond the record type.

This record **MUST** be interpreted as the end element of the most recent open element and there **MUST** exist such an element. For example, if the most recent element record corresponded to the following start element

```
<ENV:envelope>
```

then this record is interpreted as the following characters.

```
</ENV:envelope>
```

Additionally, this record **MUST** be interpreted as closing the most recently open element.

### 2.3.2 Comment Record (0x02)

This structure represents a comment.

|                  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Value (variable) |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ...              |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

**Value (variable):** The text of the comment encoded using String.

For example, if the value field represents the string "comment", this record is interpreted as the following characters.

```
<!--comment-->
```

### 2.3.3 Array Record (0x03)

This structure represents a series of repeating elements.

|                    |   |   |   |   |   |   |   |   |   |             |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |
|--------------------|---|---|---|---|---|---|---|---|---|-------------|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|
| 0                  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10          | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20                | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Element (variable) |   |   |   |   |   |   |   |   |   |             |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |
| ...                |   |   |   |   |   |   |   |   |   |             |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |
| End Element        |   |   |   |   |   |   |   |   |   | Record Type |    |    |    |    |    |    |    |    |    | Length (variable) |    |    |    |    |    |    |    |    |    |    |    |

|                 |
|-----------------|
| ...             |
| Data (variable) |
| ...             |

**Element (variable):** An Element record.

**End Element (1 byte):** An EndElement record.

**Record Type (1 byte):** The record type of the element content. This MUST be one of the values in the following table.

**Length (variable):** The number of elements, encoded with MultiByteInt31. This MUST not be zero.

**Data (variable):** The values for the elements, encoded according to RecordType.

The size of Data is the Length multiplied by the size of the RecordType according to the following table.

| RecordType | Record                     | DataSize |
|------------|----------------------------|----------|
| 0xB5       | BoolTextWithEndElement     | 1        |
| 0x8B       | Int16TextWithEndElement    | 2        |
| 0x8D       | Int32TextWithEndElement    | 4        |
| 0x8F       | Int64TextWithEndElement    | 8        |
| 0x91       | FloatTextWithEndElement    | 4        |
| 0x93       | DoubleTextWithEndElement   | 8        |
| 0x95       | DecimalTextWithEndElement  | 16       |
| 0x97       | DateTimeTextWithEndElement | 8        |
| 0xAF       | TimeSpanTextWithEndElement | 8        |
| 0XB1       | UuidTextWithEndElement     | 16       |

This record MUST be interpreted as the characters resulting from expanding this record into a series of records where the Element record is repeated for each value.

For example, if the Element and Attribute records expand to the following

```
<item xmlns='http://tempuri.org'>
```

and RecordType is Int32TextWithEndElement, and Length = 3, and Values = { 0x01, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00 }, this is interpreted as the following characters.

```
<item xmlns='http://tempuri.org'>1</item>  
<item xmlns='http://tempuri.org'>2</item>  
<item xmlns='http://tempuri.org'>3</item>
```

Since Length is 3 and the size of Int32TextWithEndElement is 4 according to the table, the size of Data is 12.. Values consist of 3 integer values, each encoded in 4 bytes as demanded by Int32TextWithEndElement. (As discussed in section [2.2.3.31](#), this MUST be interpreted as Int32Text followed by EndElement.)

Note that there is no carriage return or line feed included here, and the line break shown is for readability only.

### 3 Structure Examples

The following table provides an example of almost every record type and shows the characters that they MUST be interpreted as.

The record column shows each of the record types. The Type column shows the value of the record type, which is taken from Table 1. The Bytes column show a complete XML document encoded in this format that highlights the use of the record type. Because the record type is not always first, its location in the sequence of bytes is highlighted. The Chars column shows the same data from the Bytes column, formatted as characters. The final column, Characters represented, shows the XML that bytes MUST be interpreted as.

**Note** Records that refer to strings outside the document are shown as "strXXX" where XXX is the integer value.

| Record                       | Type (hex) | Bytes (hex)  | Chars                                 | Characters represented  |
|------------------------------|------------|--|---------------------------------------|---|
| EndElement                   | 01         | 40 03 64<br>6F 63 01   | @.doc.                                | <doc></doc>   |
| Comment                      | 02         | 02 07 63<br>6F 6D 6D<br>65 6E 74   | ..comment                             | <!--comment-->  |
| Array                        | 03         | 03 40 03<br>61 72 72<br>01 8B 03<br>33 33 88<br>88 DD DD   | .@.arr...<br>33..YY                   | <arr>13107</arr><br><arr>-30584</arr><br><arr>-8739</arr>       |
| ShortAttribute               | 04         | 40 03 64<br>6F 63 04<br>04 61 74<br>74 72 84<br>01   | @.doc..attr..                         | <doc attr="false"><br></doc>                                    |
| Attribute                    | 05         | 40 03 64<br>6F 63 09<br>03 70 72<br>65 0A 68<br>74 74 70<br>3A 2F 2F<br>61 62 63<br>05 03 70<br>72 65 04<br>61 74 74<br>72 84 01 | @.doc..pre.http://abc..pre.<br>attr.. | <doc xmlns:<br>pre="http://abc<br>" pre:attr="false"><br></doc> |
| ShortDictionary<br>Attribute | 06         | 40 03 64<br>6F 63 06<br>08 86 01   | @.doc....                             | <doc str8="true"><br></doc>                                     |
| DictionaryAttribute          | 07         | 40 03 64<br>6F 63 09<br>03 70 72<br>65 0A 68<br>74 74 70   | @.doc..pre.http://abc..<br>pre...     | <doc xmlns:pre=<br>"http://abc" p<br>re:str0="true"><br></doc>  |



| Record                        | Type (hex) | Bytes (hex)   | Chars                                 | Characters represented                               |
|-------------------------------|------------|---|---------------------------------------|--|
|                               |            | 3A 2F 2F<br>61 62 63<br>07 03 70<br>72 65 00<br>86 01   |                                       |  |
| ShortXmlnsAttribute           | 08         | 40 03 64<br>6F 63 08<br>0A 68 74<br>74 70 3A<br>2F 2F 61<br>62 63 01  | @.doc..http://abc.                    | <doc xmlns="http://abc"><br></doc>                   |
| XmlnsAttribute                | 09         | 40 03 64<br>6F 63 09<br>01 70 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 01                                     | @.doc..p.http://abc.                  | <doc xmlns:p="http://abc"><br></doc>                 |
| ShortDictionaryXmlnsAttribute | 0A         | 40 03 64<br>6F 63 0A<br>04 01   | @.doc...                              | <doc xmlns="str4"><br></doc>                         |
| DictionaryXmlnsAttribute      | 0B         | 40 03 64<br>6F 63 0B<br>01 70 04<br>01  | @.doc..p..                            | <doc xmlns:p="str4"><br></doc>                       |
| PrefixDictionaryAttributeF    | 11         | 40 03 64<br>6F 63 09<br>01 66 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 11 0B<br>98 05 68<br>65 6C 6C<br>6F 01 | @.doc..f.<br>http://abc<br>....hello. | <doc xmlns:f="http://abc" f:str11="hello"><br></doc> |
| PrefixDictionaryAttributeX    | 23         | 40 03 64<br>6F 63 09<br>01 78 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 23 15<br>98 05 77<br>6F 72 6C<br>64 01 | @.doc..x<br>.http://abc#<br>...world. | <doc xmlns:x="http://abc" x:str21="world"></doc>     |
| PrefixAttributeK              | 30         | 40 03 64<br>6F 63 09<br>01 6B 0A<br>68 74 74<br>70 3A 2F  | @.doc..<br>k.http://<br>abc0.attr..   | <doc xmlns:k="http://abc" k:attr="true"></doc>       |

| Record                    | Type (hex) | Bytes (hex)  | Chars                                  | Characters represented                           |
|---------------------------|------------|--|--|--|
|                           |            | 2F 61 62<br>63 30 04<br>61 74 74<br>72 86 01   |  |  |
| PrefixAttributeZ          | 3F         | 40 03 64<br>6F 63 09<br>01 7A 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 3F 03<br>61 62 63<br>98 03 78<br>79 7A 01 | @.doc..z<br>.http://abc?<br>.abc..xyz. | <doc xmlns:z="http://abc" z:abc="xyz"><br></doc> |
| ShortElement              | 40         | 40 03 64<br>6F 63 01   | @.doc.                                 | <doc></doc>                                      |
| Element                   | 41         | 41 03 70<br>72 65 03<br>64 6F 63<br>09 03 70<br>72 65 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 01                | A.pre.doc<br>..pre.http:<br>//abc.     | <pre:doc xmlns:pre="http://abc"><br></pre:doc>   |
| ShortDictionary Element   | 42         | 42 0E 01   | B..                                    | <str14></str14>                                  |
| DictionaryElement         | 43         | 43 03 70<br>72 65 0E<br>09 03 70<br>72 65 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 01                            | C.pre...pre.<br>http://abc.            | <pre:str14 xmlns:pre="http://abc"></pre:str14>   |
| PrefixDictionary ElementA | 44         | 44 0A 09<br>01 61 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 01  | D...a.http://<br>abc.                  | <a:str10 xmlns:a="http://abc"><br></a:str10>     |
| PrefixDictionary ElementS | 56         | 56 26 09<br>01 73 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 01  | V&..s.http:<br>//abc.                  | <s:str38 xmlns:s="http://abc"><br></s:str38>     |
| PrefixElementA            | 5E         | 5E 05 68<br>65 6C 6C   | ^.hello..a.                            | <a:hello xmlns:a="                               |

| Record                      | Type (hex) | Bytes (hex)   | Chars                          | Characters represented                                   |
|-----------------------------|------------|---|--------------------------------|--|
|                             |            | 6F 09 01<br>61 0A 68<br>74 74 70<br>3A 2F 2F<br>61 62 63<br>01  | http://abc.                    | "http://abc"><br></a:hello>                              |
| PrefixElementS              | 70         | 70 09 4D<br>79 4D 65<br>73 73 61<br>67 65 09<br>01 73 0A<br>68 74 74<br>70 3A 2F<br>2F 61 62<br>63 01 | p.MyMessage..<br>s.http://abc. | <s:MyMessage xmlns:s=<br>"http://abc"><br></s:MyMessage> |
| ZeroText                    | 80         | 40 03 64<br>6F 63 06<br>A0 03 80<br>01  | @.doc.ÿ...                     | <doc str416="0"><br></doc>                               |
| ZeroTextWithEnd<br>Element  | 81         | 40 03 61<br>62 63 81  | @.abc.                         | <abc>0</abc>   |
| OneText                     | 82         | 40 03 64<br>6F 63 06<br>00 82 01  | @.doc....                      | <doc str0="1"><br></doc>                                 |
| OneTextWithEnd<br>Element   | 83         | 40 03 61<br>62 63 83  | @.abc.                         | <abc>1</abc>   |
| FalseText                   | 84         | 40 03 64<br>6F 63 06<br>00 84 01  | @.doc....                      | <doc str0="false"><br></doc>                             |
| FalseTextWithEnd<br>Element | 85         | 40 03 61<br>62 63 85  | @.abc.                         | <abc>>false</abc>  |
| TrueText                    | 86         | 40 03 64<br>6F 63 06<br>00 86 01  | @.doc....                      | <doc str0="true"><br></doc>                              |
| TrueTextWithEnd<br>Element  | 87         | 40 03 61<br>62 63 87  | @.abc.                         | <abc>>true</abc>   |
| Int8Text                    | 88         | 40 03 64<br>6F 63 06<br>EC 01 88<br>DE 01   | @.doc.□.~.                     | <doc str236="-34"><br></doc>                             |
| Int8TextWithEnd<br>Element  | 89         | 42 9A 01<br>89 7F   | B....                          | <str154>127</str154>                                     |
| Int16Text                   | 8A         | 40 03 64<br>6F 63 06  | @.doc.□.....                   | <doc str236="-32768">                                    |

| Record                   | Type (hex) | Bytes (hex)  | Chars                               | Characters represented                  |
|--------------------------|------------|--|-------------------------------------|---|
|                          |            | EC 01 8A<br>00 80 01   |                                     | </doc>                                  |
| Int16TextWithEndElement  | 8B         | 42 9A 01<br>8B FF 7F   | B...~.                              | <str154>32767</str154>                  |
| Int32Text                | 8C         | 40 03 64<br>6F 63 06<br>EC 01 8C<br>15 CD 5B<br>07 01                | @.doc.□...I[.                       | <doc<br>str236="123456789"><br></doc>   |
| Int32TextWithEndElement  | 8D         | 42 9A 01<br>8D FF FF<br>FF 7F  | B...~.~.                            | <str154>2147483647<br></str154>         |
| Int64Text                | 8E         | 40 03 64<br>6F 63 06<br>EC 01 8E<br>00 00 00<br>80 00 00<br>00 00 01 | @.doc.□.....                        | <doc<br>str236="2147483648"><br></doc>  |
| Int64TextWithEndElement  | 8F         | 42 9A 01<br>8F 00 00<br>00 00 00<br>01 00 00                         | B.....                              | <str154>1099511627776<br></str154>      |
| FloatText                | 90         | 40 03 64<br>6F 63 04<br>01 61 90<br>CD CC 8C<br>3F 01                | @.doc..a.II.?                       | <doc a="1.1"></doc>                     |
| FloatTextWithEndElement  | 91         | 40 05 50<br>72 69 63<br>65 91 CD<br>CC 01 42                         | @.Price.II.B                        | <Price>32.45</Price>                    |
| DoubleText               | 92         | 40 03 64<br>6F 63 04<br>01 61 92<br>74 57 14<br>8B 0A BF<br>05 40 01 | @.doc..a.tW...".@.                  | <doc<br>a="2.71828182845905"><br></doc> |
| DoubleTextWithEndElement | 93         | 40 02 50<br>49 93 11<br>2D 44 54<br>FB 21 09<br>40                   | @.PI..-DT-!.@                       | <PI>3.14159265358979<br></PI>           |
| DecimalText              | 94         | 40 03 64<br>6F 63 04<br>03 69 6E<br>74 94 00<br>00 06 00             | @.doc..int.....<br>.....-N<br>..... | <doc int="5.123456"><br></doc>          |

| Record                     | Type (hex) | Bytes (hex)  | Chars                        | Characters represented   |
|----------------------------|------------|--|------------------------------|--|
|                            |            | 00 00 00<br>00 80 2D<br>4E 00 00<br>00 00 00<br>01   |                              |  |
| DecimalTextWithEndElement  | 95         | 40 08 4D<br>61 78 56<br>61 6C 75<br>65 95 00<br>00 00 00<br>FF FF FF<br>FF FF FF<br>FF FF FF<br>FF FF FF | @.MaxValue<br>.....<br>~~~~~ | <MaxValue><br>792281625142643375<br>93543950335<br></MaxValue> |
| DateTimeText               | 96         | 40 03 64<br>6F 63 06<br>6E 96 FF<br>3F 37 F4<br>75 28 CA<br>2B 01  | @.doc.n."??"<br>u(E+.        | <doc str110=<br>"9999-12-<br>31T23:59:59.9999999"><br></doc>   |
| DateTimeTextWithEndElement | 97         | 42 6C 97<br>00 40 8E<br>F9 5B 47<br>C8 08  | Bl..@.-[GE.                  | <str108><br>2006-05-17T00:00:00<br></str108>                   |
| Chars8Text                 | 98         | 40 03 64<br>6F 63 98<br>05 68 65<br>6C 6C 6F<br>01   | @.doc..hello.                | <doc>hello</doc>   |
| Chars8TextWithEndElement   | 99         | 40 01 61<br>99 05 68<br>65 6C 6C<br>6F   | @.a..hello                   | <a>hello</a>   |
| Chars16Text                | 9A         | 40 03 64<br>6F 63 9A<br>05 00 68<br>65 6C 6C<br>6F 01  | @.doc...hello.               | <doc>hello</doc>   |
| Chars16TextWithEndElement  | 9B         | 40 01 61<br>9B 05 00<br>68 65 6C<br>6C 6F  | @.a...hello                  | <a>hello</a>   |
| Chars32Text                | 9C         | 40 03 64<br>6F 63 9C<br>05 00 00<br>00 68 65<br>6C 6C 6F<br>01   | @.doc.....hello.             | <doc>hello</doc>   |

| Record                    | Type (hex) | Bytes (hex)  | Chars                    | Characters represented             |
|---------------------------|------------|--|--------------------------|------------------------------------|
| Chars32TextWithEndElement | 9D         | 40 01 61<br>9D 05 00<br>00 00 68<br>65 6C 6C<br>6F                               | @.a.....hello            | <a>hello</a>                       |
| Bytes8Text                | 9E         | 40 03 64<br>6F 63 9E<br>08 00 01<br>02 03 04<br>05 06 07<br>01                   | @.doc.....               | <doc>AAECAwQFBgc=<br></doc>        |
| Bytes8TextWithEndElement  | 9F         | 40 06 42<br>61 73 65<br>36 34 9F<br>08 00 01<br>02 03 04<br>05 06 07             | @.Base64<br>.....        | <Base64>AAECAwQFBgc=<br></Base64>  |
| Bytes16Text               | A0         | 40 03 64<br>6F 63 A0<br>08 00 00<br>01 02 03<br>04 05 06<br>07 01                | @.docÿ<br>.....          | <doc>AAECAwQFBgc=<br></doc>        |
| Bytes16TextWithEndElement | A1         | 40 06 42<br>61 73 65<br>36 34 A1<br>08 00 00<br>01 02 03<br>04 05 06<br>07       | @.Base64-<br>.....       | <Base64>AAECAwQFBgc=<br></Base64>  |
| Bytes32Text               | A2         | 40 03 64<br>6F 63 A2<br>08 00 00<br>00 00 01<br>02 03 04<br>05 06 07<br>01       | @.doc><br>.....          | <doc>AAECAwQFBgc=<br></doc>        |
| Bytes32TextWithEndElement | A3         | 40 06 42<br>61 73 65<br>36 34 A3<br>08 00 00<br>00 00 01<br>02 03 04<br>05 06 07 | @.Base64œ<br>.....       | <Base64>AAECAwQFBgc=<br></Base64>  |
| StartListText             | A4         | 40 03 64<br>6F 63 04<br>01 61 A4<br>88 7B 98<br>05 68 65<br>6C 6C 6F             | @.doc..a.{..<br>hello.Ÿ. | <doc a="123 hello true"><br></doc> |

| Record                       | Type (hex) | Bytes (hex)   | Chars                                     | Characters represented   |
|------------------------------|------------|---|---|--|
|                              |            | 86 A6 01  |   |  |
| EndListText                  | A6         | 40 03 64<br>6F 63 04<br>01 61 A4<br>88 7B 98<br>05 68 65<br>6C 6C 6F<br>86 A6 01          | @.doc..a.{..<br>hello.Ý.                  | <doc a="123 hello true"><br></doc>   |
| EmptyText                    | A8         | 40 03 64<br>6F 63 04<br>01 61 A8<br>01  | @.doc..a".                                | <doc a=""></doc>   |
| EmptyTextWithEndElement      | A9         | 40 03 64<br>6F 63 A9  | @.docc                                    | <doc></doc>  |
| DictionaryText               | AA         | 40 03 64<br>6F 63 04<br>02 6E 73<br>AA 38 01  | @.doc..ns!8.                              | <doc ns="str56"><br></doc>   |
| DictionaryTextWithEndElement | AB         | 40 04 54<br>79 70 65<br>AB C4 01  | @.Type@Ž.                                 | <Type>str196</Type>  |
| UniqueIdText                 | AC         | 40 03 64<br>6F 63 AC<br>00 11 22<br>33 44 55<br>66 77 88<br>99 AA BB<br>CC DD EE<br>FF 01 | @.doc <sup>a</sup> .."3DU<br>fw..!~IYCE~. | <doc>urn:uuid:<br>33221100-5544-<br>7766-8899-<br>aabbccddeeff<br></doc>     |
| UniqueIdTextWithEndElement   | AD         | 42 1A AD<br>00 11 22<br>33 44 55<br>66 77 88<br>99 AA BB<br>CC DD EE<br>FF                | B.-.."3DUfw..!~IYCE~                      | <str26>urn:uuid:<br>33221100-5544-<br>7766-8899-<br>aabbccddeeff<br></str26> |
| TimeSpanText                 | AE         | 40 03 64<br>6F 63 AE<br>00 C4 F5<br>32 FF FF<br>FF FF 01                                  | @.docr.Žo2~~~~.                           | <doc>-PT5M44S</doc>  |
| TimeSpanTextWithEndElement   | AF         | 42 94 07<br>AF 00 B0<br>8E F0 1B<br>00 00 00  | B.._.ø.d....                              | <str916>PT3H20M<br></str916>   |
| UuidText                     | B0         | 40 03 64<br>6F 63 B0  | @.docø....                                | <doc>03020100-   |

| Record                          | Type (hex) | Bytes (hex)  | Chars                  | Characters represented  |
|---------------------------------|------------|--|------------------------|---|
|                                 |            | 00 01 02<br>03 04 05<br>06 07 08<br>09 0A 0B<br>0C 0D 0E<br>0F 01                | .....                  | 0504-0706-0809-0a0b0c0d0e0f</doc>   |
| UuidTextWithEndElement          | B1         | 40 02 49<br>44 B1 00<br>01 02 03<br>04 05 06<br>07 08 09<br>0A 0B 0C<br>0D 0E 0F | @.IDñ.....<br>.....    | <ID>03020100-0504-0706-0809-0a0b0c0d0e0f</ID>   |
| UInt64Text                      | B2         | 40 03 64<br>6F 63 B2<br>FF FF FF<br>FF FF FF<br>FF FF 01                         | @.docý~~~~~.           | <doc>18446744073709551615</doc>   |
| UInt64TextWithEndElement        | B3         | 42 9A 01<br>B3 FE FF<br>FF FF FF<br>FF FF FF                                     | B..3_~~~~~             | <str154>18446744073709551614</str154>   |
| BoolText                        | B4         | 40 03 64<br>6F 63 B4<br>01 01  | @.doc'..               | <doc>>true</doc>  |
| BoolTextWithEndElement          | B5         | 03 40 03<br>61 72 72<br>01 B5 05<br>01 00 01<br>00 01                            | .@.arr.æ.....          | <arr>true</arr><br><arr>>false</arr><br><arr>true</arr><br><arr>>false</arr><br><arr>true</arr> |
| UnicodeChars8Text               | B6         | 40 03 64<br>6F 63 04<br>01 75 B6<br>06 75 00<br>6E 00 69<br>00 01                | @.doc..u.u.n.i..       | <doc u="uni"></doc>   |
| UnicodeChars8TextWithEndElement | B7         | 40 01 55<br>B7 06 75<br>00 6E 00<br>69 00  | @.Uú.u.n.i.            | <U>uni</U>  |
| UnicodeChars16Text              | B8         | 40 03 64<br>6F 63 04<br>03 75 31<br>36 B8 08<br>00 75 00<br>6E 00 69<br>00 32 00 | @.doc..u16,..u.n.i.2.. | <doc u16="uni2"></doc>  |



| Record                                | Type (hex) | Bytes (hex)   | Chars                | Characters represented            |
|---------------------------------------|------------|---|----------------------|-----------------------------------|
|                                       |            | 01  |                      |                                   |
| UnicodeChars16Text<br>WithEndElement  | B9         | 40 03 55<br>31 36 B9<br>08 00 75<br>00 6E 00<br>69 00 32<br>00                | @.U161..u.n.i.2.     | <U16>uni2</U16>                   |
| UnicodeChars32Text                    | BA         | 40 03 64<br>6F 63 04<br>03 75 33<br>32 BA 04<br>00 00 00<br>33 00 32<br>00 01 | @.doc..u32§....3.2.. | <doc u32="32"><br></doc>          |
| UnicodeChars32Text<br>WithEndElement  | BB         | 40 03 55<br>33 32 BB<br>04 00 00<br>00 33 00<br>32 00                         | @.U32¯....3.2.       | <U32>32</U32>                     |
| QNameDictionaryText                   | BC         | 40 03 64<br>6F 63 06<br>F0 06 BC<br>08 8E 07<br>01                            | @.doc.d.¬....        | <doc str880="i:<br>str910"></doc> |
| QNameDictionaryText<br>WithEndElement | BD         | 40 04 54<br>79 70 65<br>BD 12 90<br>07  | @.Type«...           | <Type>s:str912<br></Type>         |

As described in section 2, the document is represented by the concatenation of the characters represented by the records. There MUST NOT be any additional characters inserted.

For example, if the document consists of the records

```
ShortElement(name="element", attributes={})
Int32Text(value=1234)
FalseText
EndElement
```

then the characters represented by these records MUST be interpreted as the following.

```
<element>1234false</element>
```

## 4 Security Considerations

Ultimately, this format simply represents a textual XML document. Implementations that process XML documents represented by this format should guard against the same kinds of threats that occur when processing equivalent, textually encoded XML documents.

In many records, the length of the data precedes the data itself. Implementations should avoid allocating memory based solely on the length in order to guard against malformed or malicious records.

## 5 Appendix A: Product Behavior

This document specifies version-specific details in the Microsoft .NET Framework. The following versions of .NET Framework are available in the following released Windows product or as supplemental software, see [.NET Framework](#).

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft .NET Framework 3.0
- Microsoft .NET Framework 3.5
- Microsoft .NET Framework 4.0
- Microsoft .NET Framework 4.5

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

## 6 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

## 7 Index

### A

[Applicability](#) 8  
[ArrayRecord packet](#) 45  
[Attribute records](#) 23  
[AttributeRecord packet](#) 23

### B

[BoolTextRecord packet](#) 42  
[Bytes16TextRecord packet](#) 37  
[Bytes32TextRecord packet](#) 37  
[Bytes8TextRecord packet](#) 36

### C

[Change tracking](#) 60  
[Chars16TextRecord packet](#) 36  
[Chars32TextRecord packet](#) 36  
[Chars8TextRecord packet](#) 34  
[CommentRecord packet](#) 45  
[Common data types and fields](#) 10

### D

[Data types and fields - common](#) 10  
[DateTimeTextRecord packet](#) 33  
[DecimalTextRecord packet](#) 32  
[Definitions](#) 10  
Details  
    [common data types and fields](#) 10  
    [MultiByteInt31 structure](#) 13  
[DictionaryAttributeRecord packet](#) 25  
[DictionaryElementRecord packet](#) 21  
[DictionaryString packet](#) 18  
[DictionaryTextRecord packet](#) 38  
[DictionaryXmIsAttributeRecord packet](#) 27  
[DoubleTextRecord packet](#) 31

### E

[Element records](#) 19  
[Element\\_Record packet](#) 20  
[Example](#) 48

### F

[Fields - vendor-extensible](#) 9  
[FloatTextRecord packet](#) 31

### G

[Glossary](#) 6

### I

[Implementer - security considerations](#) 58  
[Informative references](#) 7  
[Int16TextRecord packet](#) 29

[Int32TextRecord packet](#) 30  
[Int64TextRecord packet](#) 30  
[Int8TextRecord packet](#) 29  
[Introduction](#) 6

### L

[Localization](#) 9

### M

[Miscellaneous records](#) 44  
[MultiByteInt31 structure](#) 13  
[MultiByteInt31\\_1\\_Byte packet](#) 14  
[MultiByteInt31\\_2\\_Byte packet](#) 14  
[MultiByteInt31\\_3\\_Bytes packet](#) 15  
[MultiByteInt31\\_4\\_Bytes packet](#) 16  
[MultiByteInt31\\_5\\_Bytes packet](#) 17

### N

[Normative references](#) 7

### O

[Overview \(synopsis\)](#) 8

### P

[PrefixAttributeRecords packet](#) 27  
[PrefixDictionaryAttributeRecords packet](#) 27  
[PrefixDictionaryElement\\_A\\_Z\\_Record packet](#) 21  
[PrefixElement\\_A\\_Z\\_Record packet](#) 22  
[Product behavior](#) 59

### Q

[QNameDictionaryTextRecord packet](#) 43

### R

[Record packet](#) 10  
Records  
    [attribute](#) 23  
    [element](#) 19  
    [miscellaneous](#) 44  
    [overview](#) 19  
    [text](#) 28  
References  
    [informative](#) 7  
    [normative](#) 7  
[Relationship to protocols and other structures](#) 8

### S

[Security - implementer considerations](#) 58  
[ShortAttributeRecord packet](#) 23  
[ShortDictionaryAttributeRecord packet](#) 24  
[ShortDictionaryElementRecord packet](#) 20

[ShortDictionaryXmlnsAttributeRecord\\_packet](#) 26  
[ShortElementRecord\\_packet](#) 19  
[ShortXmlnsAttributeRecord\\_packet](#) 25  
[String\\_packet](#) 18  
Structures  
    [MultiByteInt31](#) 13  
    [overview](#) 10

## T

[Text\\_records](#) 28  
[TimeSpanTextRecord\\_packet](#) 39  
[Tracking\\_changes](#) 60

## U

[UInt64TextRecord\\_packet](#) 41  
[UnicodeChars16TextRecord\\_packet](#) 43  
[UnicodeChars32TextRecord\\_packet](#) 43  
[UnicodeChars8TextRecord\\_packet](#) 42  
[UniqueIdTextRecord\\_packet](#) 39  
[UuidTextRecord\\_packet](#) 40

## V

[Vendor-extensible\\_fields](#) 9  
[Versioning](#) 9

## X

[XmlnsAttributeRecord\\_packet](#) 26