**[MC-NMF]:**

**.NET Message Framing Protocol**

Intellectual Property Rights Notice for Open Specifications Documentation

* **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
* **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
* **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
* **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](https://go.microsoft.com/fwlink/?LinkId=214445) or the [Microsoft Community Promise](https://go.microsoft.com/fwlink/?LinkId=214448). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
* **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](https://msdn.microsoft.com/en-us/openspecifications/dn750984).
* **Trademarks**. The names of companies and products contained in this documentation might 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](https://www.microsoft.com/trademarks).
* **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

**Tools**. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

**Support.** For questions and support, please contact dochelp@microsoft.com.

**Revision Summary**

| Date | Revision History | Revision Class | Comments |
| --- | --- | --- | --- |
| 8/10/2007 | 0.1 | Major | Initial Availability |
| 9/28/2007 | 0.2 | Minor | Clarified the meaning of the technical content. |
| 10/23/2007 | 0.3 | Minor | Clarified the meaning of the technical content. |
| 11/30/2007 | 0.3.1 | Editorial | Changed language and formatting in the technical content. |
| 1/25/2008 | 0.3.2 | Editorial | Changed language and formatting in the technical content. |
| 3/14/2008 | 0.3.3 | Editorial | Changed language and formatting in the technical content. |
| 5/16/2008 | 0.3.4 | Editorial | Changed language and formatting in the technical content. |
| 6/20/2008 | 0.4 | Minor | Clarified the meaning of the technical content. |
| 7/25/2008 | 0.4.1 | Editorial | Changed language and formatting in the technical content. |
| 8/29/2008 | 0.4.2 | Editorial | Changed language and formatting in the technical content. |
| 10/24/2008 | 0.4.3 | Editorial | Changed language and formatting in the technical content. |
| 12/5/2008 | 1.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 1.0.1 | Editorial | Changed language and formatting in the technical content. |
| 2/27/2009 | 1.0.2 | Editorial | Changed language and formatting in the technical content. |
| 4/10/2009 | 1.0.3 | Editorial | Changed language and formatting in the technical content. |
| 5/22/2009 | 1.1 | Minor | Clarified the meaning of the technical content. |
| 7/2/2009 | 1.1.1 | Editorial | Changed language and formatting in the technical content. |
| 8/14/2009 | 1.1.2 | Editorial | Changed language and formatting in the technical content. |
| 9/25/2009 | 1.2 | Minor | Clarified the meaning of the technical content. |
| 11/6/2009 | 1.2.1 | Editorial | Changed language and formatting in the technical content. |
| 12/18/2009 | 1.2.2 | Editorial | Changed language and formatting in the technical content. |
| 1/29/2010 | 1.3 | Minor | Clarified the meaning of the technical content. |
| 3/12/2010 | 1.4 | Minor | Clarified the meaning of the technical content. |
| 4/23/2010 | 1.5 | Minor | Clarified the meaning of the technical content. |
| 6/4/2010 | 1.6 | Minor | Clarified the meaning of the technical content. |
| 7/16/2010 | 2.0 | Major | Updated and revised the technical content. |
| 8/27/2010 | 2.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/8/2010 | 2.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/19/2010 | 2.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/7/2011 | 3.0 | Major | Updated and revised the technical content. |
| 2/11/2011 | 3.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/25/2011 | 3.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/6/2011 | 3.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/17/2011 | 4.0 | Major | Updated and revised the technical content. |
| 9/23/2011 | 4.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011 | 5.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 5.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/12/2012 | 5.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/25/2012 | 6.0 | Major | Updated and revised the technical content. |
| 1/31/2013 | 6.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/8/2013 | 6.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/14/2013 | 6.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 6.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 6.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 7.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 7.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 7.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/16/2017 | 8.0 | Major | Significantly changed the technical content. |
| 6/1/2017 | 8.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/16/2018 | 9.0 | Major | Significantly changed the technical content. |
| 3/13/2019 | 10.0 | Major | Significantly changed the technical content. |

Table of Contents

[1 Introduction 7](#_Toc2248971)

[1.1 Glossary 7](#_Toc2248972)

[1.2 References 7](#_Toc2248973)

[1.2.1 Normative References 8](#_Toc2248974)

[1.2.2 Informative References 8](#_Toc2248975)

[1.3 Overview 9](#_Toc2248976)

[1.3.1 Scenarios 10](#_Toc2248977)

[1.3.1.1 Multiple Bidirectional Message Exchange Scenario 11](#_Toc2248978)

[1.3.1.2 Large Message Exchange Scenario 11](#_Toc2248979)

[1.3.1.3 Offline Message Exchange Scenario 11](#_Toc2248980)

[1.3.2 Communication Modes 11](#_Toc2248981)

[1.3.2.1 Message Property Scope 11](#_Toc2248982)

[1.3.2.2 Protocol Receiver Mode 12](#_Toc2248983)

[1.3.2.3 Message Traffic Flow 12](#_Toc2248984)

[1.3.2.4 Message Chunking 12](#_Toc2248985)

[1.3.3 Protocol Upgrades 12](#_Toc2248986)

[1.4 Relationship to Other Protocols 12](#_Toc2248987)

[1.5 Prerequisites/Preconditions 12](#_Toc2248988)

[1.6 Applicability Statement 13](#_Toc2248989)

[1.7 Versioning and Capability Negotiation 13](#_Toc2248990)

[1.8 Vendor-Extensible Fields 13](#_Toc2248991)

[1.9 Standards Assignments 13](#_Toc2248992)

[2 Messages 14](#_Toc2248993)

[2.1 Transport 14](#_Toc2248994)

[2.2 Message Syntax 14](#_Toc2248995)

[2.2.1 Record Types 14](#_Toc2248996)

[2.2.2 Record Size Encoding 14](#_Toc2248997)

[2.2.3 Property Records 15](#_Toc2248998)

[2.2.3.1 Version Record 16](#_Toc2248999)

[2.2.3.2 Mode Record 16](#_Toc2249000)

[2.2.3.3 Via Record 17](#_Toc2249001)

[2.2.3.4 Envelope Encoding Record 17](#_Toc2249002)

[2.2.3.4.1 Known Encoding Record 17](#_Toc2249003)

[2.2.3.4.2 Extensible Encoding Record 18](#_Toc2249004)

[2.2.3.5 Upgrade Request Record 19](#_Toc2249005)

[2.2.3.6 Upgrade Response Record 20](#_Toc2249006)

[2.2.3.7 Preamble End Record 20](#_Toc2249007)

[2.2.3.8 Preamble Ack Record 20](#_Toc2249008)

[2.2.3.9 End Record 20](#_Toc2249009)

[2.2.4 Envelope Records 20](#_Toc2249010)

[2.2.4.1 Sized Envelope Record 21](#_Toc2249011)

[2.2.4.2 Data Chunk 21](#_Toc2249012)

[2.2.4.3 Unsized Envelope Record 21](#_Toc2249013)

[2.2.5 Fault Records 22](#_Toc2249014)

[2.2.6 Preamble Message 23](#_Toc2249015)

[3 Protocol Details 25](#_Toc2249016)

[3.1 Common Details 25](#_Toc2249017)

[3.1.1 Abstract Data Model 25](#_Toc2249018)

[3.1.1.1 Initiator-Receiver Interactions 25](#_Toc2249019)

[3.1.1.1.1 Singleton Unsized Mode 25](#_Toc2249020)

[3.1.1.1.2 Duplex Mode 26](#_Toc2249021)

[3.1.1.1.3 Simplex Mode 27](#_Toc2249022)

[3.1.1.1.4 Singleton Sized Mode 28](#_Toc2249023)

[3.1.1.1.5 Upgrades 28](#_Toc2249024)

[3.1.1.1.6 Faults 29](#_Toc2249025)

[3.1.1.2 Protocol Grammar 31](#_Toc2249026)

[3.1.2 Timers 33](#_Toc2249027)

[3.1.3 Initialization 33](#_Toc2249028)

[3.1.4 Higher-Layer Triggered Events 33](#_Toc2249029)

[3.1.4.1 Reading Variable-Sized Records 33](#_Toc2249030)

[3.1.4.2 Handling Receipt of an Unexpected Record Type 34](#_Toc2249031)

[3.1.4.3 Version Record 34](#_Toc2249032)

[3.1.4.4 Mode Record 34](#_Toc2249033)

[3.1.4.5 Via Record 34](#_Toc2249034)

[3.1.4.6 Encoding Record 35](#_Toc2249035)

[3.1.4.7 Upgrade Request Record 35](#_Toc2249036)

[3.1.4.8 Upgrade Response Record 35](#_Toc2249037)

[3.1.4.9 Preamble End Record 35](#_Toc2249038)

[3.1.4.10 Preamble Ack Record 36](#_Toc2249039)

[3.1.4.11 Sized Envelope Record 36](#_Toc2249040)

[3.1.4.12 Unsized Envelope Record 36](#_Toc2249041)

[3.1.4.13 End Record 36](#_Toc2249042)

[3.1.5 Message Processing Events and Sequencing Rules 36](#_Toc2249043)

[3.1.6 Timer Events 36](#_Toc2249044)

[3.1.7 Other Local Events 36](#_Toc2249045)

[3.1.7.1 Underlying Transport Session Is Closed 36](#_Toc2249046)

[3.2 Initiator Details 37](#_Toc2249047)

[3.2.1 Abstract Data Model 37](#_Toc2249048)

[3.2.2 Timers 37](#_Toc2249049)

[3.2.3 Initialization 37](#_Toc2249050)

[3.2.4 Higher-Layer Triggered Events 37](#_Toc2249051)

[3.2.4.1 Initialize Session 37](#_Toc2249052)

[3.2.4.2 Send Preamble 37](#_Toc2249053)

[3.2.4.3 Send Message 37](#_Toc2249054)

[3.2.4.3.1 Singleton Unsized Mode 37](#_Toc2249055)

[3.2.4.3.2 Duplex or Simplex Mode 38](#_Toc2249056)

[3.2.4.3.3 Singleton Sized Mode 38](#_Toc2249057)

[3.2.4.4 Receive Message 38](#_Toc2249058)

[3.2.4.5 Send End Record 38](#_Toc2249059)

[3.2.4.6 Session Close 38](#_Toc2249060)

[3.2.5 Message Processing Events and Sequencing Rules 38](#_Toc2249061)

[3.2.6 Timer Events 38](#_Toc2249062)

[3.2.7 Other Local Events 38](#_Toc2249063)

[3.3 Receiver Details 38](#_Toc2249064)

[3.3.1 Abstract Data Model 38](#_Toc2249065)

[3.3.2 Timers 38](#_Toc2249066)

[3.3.3 Initialization 38](#_Toc2249067)

[3.3.4 Higher-Layer Triggered Events 39](#_Toc2249068)

[3.3.4.1 Initialize Session 39](#_Toc2249069)

[3.3.4.2 Receive Preamble 39](#_Toc2249070)

[3.3.4.3 Send Message 39](#_Toc2249071)

[3.3.4.4 Receive Message 39](#_Toc2249072)

[3.3.4.4.1 Singleton Unsized Mode 39](#_Toc2249073)

[3.3.4.4.2 Duplex or Simplex Mode 39](#_Toc2249074)

[3.3.4.4.3 Singleton Sized Mode 39](#_Toc2249075)

[3.3.4.5 Send End Record 40](#_Toc2249076)

[3.3.4.6 Session Close 40](#_Toc2249077)

[3.3.5 Message Processing Events and Sequencing Rules 40](#_Toc2249078)

[3.3.6 Timer Events 40](#_Toc2249079)

[3.3.7 Other Local Events 40](#_Toc2249080)

[4 Protocol Examples 41](#_Toc2249081)

[4.1 Duplex Mode 41](#_Toc2249082)

[4.1.1 Initiator Receiver: Preamble Message 42](#_Toc2249083)

[4.1.2 Initiator Receiver: Preamble End Message 42](#_Toc2249084)

[4.1.3 Receiver Initiator : Preamble Ack Message 43](#_Toc2249085)

[4.1.4 Initiator Receiver: Sized Envelope Message 43](#_Toc2249086)

[4.1.5 Receiver Initiator: Sized Envelope Message 44](#_Toc2249087)

[4.1.6 Initiator Receiver: End Message 44](#_Toc2249088)

[4.1.7 Receiver Initiator: End Message 44](#_Toc2249089)

[5 Security 45](#_Toc2249090)

[5.1 Security Considerations for Implementers 45](#_Toc2249091)

[5.2 Index of Security Parameters 45](#_Toc2249092)

[6 Appendix A: Product Behavior 46](#_Toc2249093)

[7 Change Tracking 50](#_Toc2249094)

[8 Index 51](#_Toc2249095)

# Introduction

This document specifies the .NET Message Framing Protocol, which defines a mechanism for framing messages. Although primarily used for framing SOAP messages, this protocol can also be used to frame messages that use non-SOAP envelope formats. The .NET Message Framing Protocol can run over any transport, including those that do not natively support message semantics, and can provide support for sending and receiving demarcated messages.

Familiarity with SOAP and XML technologies is required for a complete understanding of this document.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

## Glossary

This document uses the following terms:

**endpoint**: A node that sends or receives a [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b).

**envelope record**: A record that contains data, such as a SOAP message. For more information about envelope records, see [[SOAP1.1]](https://go.microsoft.com/fwlink/?LinkId=90520) and [[SOAP1.2-1/2007]](https://go.microsoft.com/fwlink/?LinkId=94664).

**Initiating Stream**: The protocol stream that flows from the initiator.

**initiator**: The node that initiates the connection over which a [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b) flows.

**little-endian**: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

**Property Record**: A record that contains a [**protocol stream property**](#gt_4ddc88f0-31bc-4e87-8e77-3388f9f3d012).

**protocol stream**: A continuous stream of records flowing in one direction.

**protocol stream property**: A protocol stream characteristic that can be set by a property record and that applies to subsequent records flowing with the [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b).

**receiver**: The node that is the receiver of the [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b).

**record**: A sequence of octets.

**Unicode**: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) standard [[UNICODE5.0.0/2007]](https://go.microsoft.com/fwlink/?LinkId=154659) provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119]](https://go.microsoft.com/fwlink/?LinkId=90317). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](https://go.microsoft.com/fwlink/?linkid=850906).

### 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. We will assist you in finding the relevant information.

[MC-NBFSE] Microsoft Corporation, "[.NET Binary Format: SOAP Extension](%5BMC-NBFSE%5D.pdf#Section_3caf1ffa63334c5f8aa7c06db4748c1b)".

[MC-NBFS] Microsoft Corporation, "[.NET Binary Format: SOAP Data Structure](%5BMC-NBFS%5D.pdf#Section_9567b35067884632aa34a87235e52459)".

[MS-DTYP] Microsoft Corporation, "[Windows Data Types](%5BMS-DTYP%5D.pdf#Section_cca2742956894a16b2b49325d93e4ba2)".

[MS-MQMQ] Microsoft Corporation, "[Message Queuing (MSMQ): Data Structures](%5BMS-MQMQ%5D.pdf#Section_b7cc2590a61745dfb6a31f31102b36fb)".

[RFC2045] Freed, N., and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996, [http://www.rfc-editor.org/rfc/rfc2045.txt](https://go.microsoft.com/fwlink/?LinkId=90307)

[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](https://go.microsoft.com/fwlink/?LinkId=90317)

[RFC2234] Crocker, D. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997, [http://www.ietf.org/rfc/rfc2234.txt](https://go.microsoft.com/fwlink/?LinkId=90323)

[RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998, [http://www.rfc-editor.org/rfc/rfc2279.txt](https://go.microsoft.com/fwlink/?LinkId=90331)

[RFC2396] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, [http://www.rfc-editor.org/rfc/rfc2396.txt](https://go.microsoft.com/fwlink/?LinkId=90339)

[RFC2781] Hoffman, P., and Yergeau, F., "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000, [http://www.rfc-editor.org/rfc/rfc2781.txt](https://go.microsoft.com/fwlink/?LinkId=90380)

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and Ingersoll, W., "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism", RFC 4178, October 2005, [https://www.rfc-editor.org/rfc/rfc4178.txt](https://go.microsoft.com/fwlink/?LinkId=90461)

[RFC5246] Dierks, T., and Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008, [http://www.ietf.org/rfc/rfc5246.txt](https://go.microsoft.com/fwlink/?LinkId=129803)

[RFC793] Postel, J., Ed., "Transmission Control Protocol: DARPA Internet Program Protocol Specification", RFC 793, September 1981, [http://www.rfc-editor.org/rfc/rfc793.txt](https://go.microsoft.com/fwlink/?LinkId=150872)

[SOAP-MTOM] Gudgin, M., Medelsohn, N., Nottingham, M., and Ruellan, H., "SOAP Message Transmission Optimization Mechanism", W3C Recommendation, 25 January 2005, [http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/](https://go.microsoft.com/fwlink/?LinkId=95126)

### Informative References

[MS-MQOD] Microsoft Corporation, "[Message Queuing Protocols Overview](%5BMS-MQOD%5D.pdf#Section_644be85f7a784bccb8a1389e4b24b2cc)".

[MS-NETOD] Microsoft Corporation, "[Microsoft .NET Framework Protocols Overview](%5BMS-NETOD%5D.pdf#Section_bcca8164da0843f2a983c34ed99171b0)".

[MSDN-BinaryMsgEncdngBindElmnt] Microsoft Corporation, "BinaryMessageEncodingBindingElement Class", [http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.binarymessageencodingbindingelement.aspx](https://go.microsoft.com/fwlink/?LinkId=216916)

[MSDN-NETMsmqBE] Microsoft Corporation, "MsmqTransportBindingElement Class", [http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.msmqtransp ortbindingelement.aspx](https://go.microsoft.com/fwlink/?LinkId=100289)

[MSDN-NETMsmq] Microsoft Corporation, "NetMsmqBinding Class", [http://msdn.microsoft.com/en-us/library/system.servicemodel.netmsmqbinding.aspx](https://go.microsoft.com/fwlink/?LinkId=100286)

[MSDN-NETNamedPipeBE] Microsoft Corporation, "NamedPipeTransportBindingElement Class", [http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.namedpipet ransportbindingelement.aspx](https://go.microsoft.com/fwlink/?LinkId=100288)

[MSDN-NETNamedPipe] Microsoft Corporation, "NetNamedPipeBinding Class", [http://msdn.microsoft.com/en-us/library/system.servicemodel.netnamedpipebinding.aspx](https://go.microsoft.com/fwlink/?LinkId=100285)

[MSDN-NETTcpBE] Microsoft Corporation, "TcpTransportBindingElement Class", [http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.tcptranspo rtbindingelement.aspx](https://go.microsoft.com/fwlink/?LinkId=100287)

[MSDN-NETTcp] Microsoft Corporation, "NetTcpBinding Class", [http://msdn.microsoft.com/en-us/library/system.servicemodel.nettcpbinding.aspx](https://go.microsoft.com/fwlink/?LinkId=100283)

[MSDN-WCF] Microsoft Corporation, "Windows Communication Foundation", [http://msdn.microsoft.com/en-us/library/ms735119.aspx](https://go.microsoft.com/fwlink/?LinkId=100290)

[MSDN-WSCHBIND] Microsoft Corporation, "WS\_CHANNEL\_BINDING enumeration", [http://msdn.microsoft.com/en-us/library/dd401780(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=182220)

[MSDN-WSSECBIND] Microsoft Corporation, "WS\_SECURITY\_BINDING structure", [http://msdn.microsoft.com/en-us/library/dd323380(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=182275)

[MSDN-WSTCPSSPI] Microsoft Corporation, "WS\_TCP\_SSPI\_TRANSPORT\_SECURITY\_BINDING structure", [http://msdn.microsoft.com/en-us/library/dd323466(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=182221)

## Overview

The .NET Message Framing Protocol defines a format for framing messages, including SOAP messages. Consider a scenario in which two SOAP nodes are interacting and exchanging SOAP messages. The transport used for communication might not inherently support the notion of messages. For example, if the underlying transport is TCP, it provides a byte stream, and the [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) needs to have additional parsing logic to be able to extract a SOAP message from this stream.

This protocol intends to meet the following requirements:

* Supports extensibility for different message-encoding formats.
* Provides delimiters for a message.
* Has capability to skip past a message that is not well formed. If the message frames are well formed but the embedded content is malformed, the protocol provides a means of skipping over all such message frames.
* Supports extensible upgrades of the underlying transport stream.

The basic idea is to first notify the recipient of the message properties (metadata), including what version of the framing protocol is being used, who the message is meant for, and what encoding algorithm is used to encode the message content; and then to send a number of message frames that conform to those properties. The recipient, based on the message properties, is able to extract the messages from the transport stream and deliver them to the appropriate [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee).

The message properties are typically controlled by the Protocol Configuration Object (PCO). The PCO determines the following aspects of a specified instance of the protocol:

* The transport to be used.
* The version of the .NET Message Framing Protocol being used.
* The mode of communication, which is explained in sections [1.3.2](#Section_0c44b067d8fb4852b8f09d5482179c67) and [2.2.3.2](#Section_8cd0b687a7fc45e5b328e00225836af3).
* The Via, which is a Uniform Resource Identifier (URI) that identifies the endpoint for which the messages are intended.
* The encoding format being used for the messages. The different encoding schemes are covered in section [2.2.3.4](#Section_8d29c0f1b83c4778b62555b03f373084).
* The chunk size. If the mode supports chunking, this determines the maximum size of a chunk.
* The implementation-defined maximum supported sizes for messages and [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) types. [<1>](#Appendix_A_1" \o "Product behavior note 1)

### Scenarios

This section describes scenarios that capture the various message exchange patterns between SOAP nodes. These scenarios help to define the communication modes that are covered in the next section and that the protocol needs to support.

The scenarios describe a sales organization that has several salespersons; some are in the head office and some offsite. They are interacting with the customers and preparing purchase orders that need to be sent to a central server as SOAP messages. The purchase orders can also be retrieved from the server, again as SOAP messages. The Asynchronous Message Relay is a mechanism that is used to queue up messages when the salesperson is offline and then relay the messages after connectivity is established. One such mechanism is Microsoft Message Queuing, as described in [[MS-MQOD]](%5BMS-MQOD%5D.pdf#Section_644be85f7a784bccb8a1389e4b24b2cc).



Figure 1: Asynchronous Message Relay

#### Multiple Bidirectional Message Exchange Scenario

In this scenario, two salespersons are working at the head office with several customers. Salesperson A is responsible for collecting the customer profiles, and salesperson B is responsible for collecting the customer requirements. The two pieces of information will need to be combined to create a purchase order. Also, the head office has a high volume of customers; so there will be frequent message exchanges between the two salespeople.

For this scenario, it makes sense for salesperson A to initiate a session where the message properties are sent out. Subsequently, the messages frames are sent from salesperson A or salesperson B, and the other salesperson can extract the message by using the message properties for that session. At the end of the conversation, either salesperson can terminate the session.

#### Large Message Exchange Scenario

In this scenario, a salesperson retrieves the entire customer inventory (in the form of a message) from the server at the start of the day.

Because this operation is typically performed only once each day, a session is not required, as was the case in the previous scenario. Instead, the protocol sends the message properties followed by the message frames, and the [**receiving end**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) applies the properties to extract the message.

In addition, because the inventory is large, the message content is broken up into multiple chunks. The receiving end can then stream the content one chunk at a time and does not have to process the entire message at one time.

#### Offline Message Exchange Scenario

In this scenario, salesperson C is visiting various customers and creating their purchase orders. However, the salesperson does not have access to the server and can upload these orders to the server only after he returns to his branch office. The order application uses some mechanism (for example, Microsoft Message Queuing) to store these messages locally, and the mechanism then relays the message to the server when the salesperson is again online.

This scenario differs from the scenario in section [1.3.1.2](#Section_0a8c5aee01f14b5b9632d30d338e6380) because the [**receiving end**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) of the protocol (that is, the relay) cannot actively participate in the protocol. This is a "store and forward" scenario in which the sending end of the protocol stores the message frame in an intermediate store, and later, the message frame is forwarded to, or retrieved by, the receiving end, which then extracts the message from the message frame.

Depending on the scenario characteristics, the message properties are sent on a per-message basis or sent once in advance of a number of messages. The latter case uses the same session semantic as before except that the session establishment involves participation from only one end.

### Communication Modes

Based on the preceding scenarios, the messages exchange between nodes can be classified along the following four criteria.

#### Message Property Scope

Message properties can be sent on a per message basis or sent once per session, which spans multiple messages. If many messages that have identical properties are being sent, the optimal workflow uses the per-session scope.

#### Protocol Receiver Mode

The receiving end can actively participate in the protocol, or it can be a passive relay entity. If the receiving end is active, it can negotiate certain capabilities, such as a protocol upgrade.

#### Message Traffic Flow

The logical flow of messages can be unidirectional, where only one end sends messages, or it can be bidirectional, where both ends send messages. For unidirectional messages, the [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) can acknowledge message receipt; however, the logical message flow is still in one direction.

#### Message Chunking

The entire message can be sent in one message frame, or it can be split across multiple chunks. Chunking is extremely useful when processing large messages.

Using these criteria, four communication modes are specified for the protocol to operate in. These modes determine the pattern of messages exchanged between the nodes, and determine when the message properties are exchanged and how the message frames are created.

| Mode name  | Message property scope  | Protocol receiver mode  | Traffic flow  | Message chunking  |
| --- | --- | --- | --- | --- |
| Singleton Unsized | Single | Active | Unidirectional | Yes |
| Duplex | Multiple | Active | Bidirectional | No |
| Simplex | Multiple  | Passive | Unidirectional | No |
| Singleton Sized | Single | Passive | Unidirectional | No |

### Protocol Upgrades

The .NET Message Framing Protocol provides the capability to upgrade the underlying [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b) to a complementary protocol, for example, to upgrade to Secure Sockets Layer (SSL)/Transport Layer Security (TLS). If the other end supports the complementary protocol and goes through with the upgrade, the subsequent byte stream (messages included) use the upgraded protocol.

The upgrade request is sent as part of message properties. Multiple upgrade negotiations can be performed. In addition, because this is a negotiation, it requires participation from both ends, and therefore, is available only when the communication mode is Singleton Unsized, or Duplex.

## Relationship to Other Protocols

This protocol is available for use over any network transport that needs to provide message send and receive semantics. Transports that fall in this category include TCP and named pipes.

## Prerequisites/Preconditions

The protocol assumes that a transport session has been established. The management of the transport session (that is, how and when it is established, management of idle sessions, and closure of the transport session) is not a responsibility of the protocol. The protocol only uses the transport session to send and receive octets.

For the Singleton Sized mode, which is described in section [1.3.2](#Section_0c44b067d8fb4852b8f09d5482179c67), the size of the message is not contained as part of the message frame. The protocol assumes that the underlying transport has a means to compute the size and relay it to the protocol.

## Applicability Statement

This protocol is applicable for implementation by a transport module that wants to provide message demarcation to higher-layer applications. Higher-layer applications can use this module to send and receive messages.

Applicable scenarios include the following:

* When the communicating nodes are connected (for example, employees in the head office) or when they are disconnected (for example, an employee working remotely).
* When the communicating nodes are exchanging large messages and message-level streaming is required to optimize the use of resources such as memory and processing.
* When the communicating nodes want to upgrade the underlying transport to a complementary protocol and exchange messages using the complementary protocol.
* When a receiving node wants to bypass embedded messages that are not well formed and process subsequent messages that are well-formed.

The protocol is not applicable for scenarios in which applications do not need message-level access or the native message format of the underlying transport is sufficient.

## Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

* **Protocol versions:** This document describes version 1.0 of the .NET Message Framing Protocol. The version information is part of the protocol exchange, as described in section [2.2.3.1](#Section_b8dfbcd6b65b495aa1f1e19b78897a3d).
* **Capability negotiation:** The .NET Message Framing Protocol does not support negotiation of the version, mode, upgrades, and message encoding. Instead, an implementation is configured with these, as described in section [3.1.3](#Section_6046b1b2de1544c684182d1a1a11ad67).

## Vendor-Extensible Fields

This protocol allows extensibility for the following fields:

* **Extensible encoding:** An implementation can opt for an extensible encoding. Vendors need to specify the encoding as specified in [[RFC2045]](https://go.microsoft.com/fwlink/?LinkId=90307) and covered in detail in section [2.2.3.4.2](#Section_ac28de9318424d6680741c2a34e24720).
* **Upgrades:** Vendors can define new protocol upgrades in addition to the ones specified in section [2.2.3.5](#Section_6ef3ede82ead42ceabf16d144271746f).
* **Faults:** An implementation can define new faults in addition to the ones specified in section [2.2.5](#Section_337e8351085448d2864b97520026c2c6). The fault is a URI, as defined in [[RFC2396]](https://go.microsoft.com/fwlink/?LinkId=90339) encoding using UTF-8 encoding as specified in [[RFC2279]](https://go.microsoft.com/fwlink/?LinkId=90331). Vendors define a URI namespace for their faults and that namespace is different from the http://schemas.microsoft.com/ws/2006/05/framing/faults/ namespace used by the faults in this protocol.

## Standards Assignments

None.

# Messages

This protocol references commonly used data types as defined in [[MS-DTYP]](%5BMS-DTYP%5D.pdf#Section_cca2742956894a16b2b49325d93e4ba2).

## Transport

This protocol is available for use over any network transport that needs to provide message send and receive semantics. Transports that fall in this category include TCP and named pipes.

## Message Syntax

### Record Types

This protocol involves the exchange of a number of [**records**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71). Records can be categorized as either [**Property Records**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) or [**Envelope Records**](#gt_db932f27-24c8-4fd8-a20f-b39334434637) based on their contents. The Property Records contain message properties. The Envelope Records contain the message payload.

These records and their structure are covered in detail in subsequent sections. Each record is prefixed with a record type, which is an octet, and MUST be set to one of the following specified values. Values of 0x0D-0xFF for this octet are reserved for future use.

| Value  | Record type  |
| --- | --- |
| 0x00 | [Version Record](#Section_b8dfbcd6b65b495aa1f1e19b78897a3d) |
| 0x01 | [Mode Record](#Section_8cd0b687a7fc45e5b328e00225836af3) |
| 0x02 | [Via Record](#Section_2d61448ab4d14cbd911cd5caff54b64f) |
| 0x03 | [Known Encoding Record](#Section_a7d444632c60482d88569a8ff5929c62) |
| 0x04 | [Extensible Encoding Record](#Section_ac28de9318424d6680741c2a34e24720) |
| 0x05 | [Unsized Envelope Record](#Section_5e46160dc9384882bf160477449e61e3) |
| 0x06 | [Sized Envelope Record](#Section_82159b784bbe4ca8a707eeb65d3c6173) |
| 0x07 | [End Record](#Section_9d691d25fd3a41ca9eb5241f47a914e5) |
| 0x08 | [Fault Record](#Section_337e8351085448d2864b97520026c2c6) |
| 0x09 | [Upgrade Request Record](#Section_6ef3ede82ead42ceabf16d144271746f) |
| 0x0A | [Upgrade Response Record](#Section_8f9da3a35345482cbcb7543972c1ee4a) |
| 0x0B | [Preamble Ack Record](#Section_3ec275c549e04becb7515cb572c3e7ab) |
| 0x0C | [Preamble End Record](#Section_2ac9c4efa6544b109cf8b5b8917c654e) |

### Record Size Encoding

For the variable-sized [**records**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) that are used by this protocol, the record needs to contain the size, in octets, of the content. An implementation SHOULD support record sizes as large as 0xffffffff octets (encoded size requires five octets).[<2>](#Appendix_A_2" \o "Product behavior note 2)

As represented in the following figure, the encoding algorithm takes the size of the record payload as input in [**little-endian**](#gt_079478cb-f4c5-4ce5-b72b-2144da5d2ce7) format and generates a stream of octets. The octets MUST be sent in the order in which they are generated.



Figure 2: The encoding algorithm

The following table lists the encoded sizes for the range of values of Size, which is computed as previously explained. The network ordering of octets is top-down. For example, if the size is in the range 0x80-0x3FFF, the network ordering of encoded size octets is (Size & 0x7F) | 0x80 followed by Size >> 0x07.

| Integer value (size)  | Encoding  |
| --- | --- |
| 0x00-0x7F | Size |
| 0x80-0x3FFF | (Size & 0x7F)| 0x80Size >> 0x07 |
| 0x4000-0x1FFFFF | (Size & 0x7F)| 0x80((Size >> 0x07) & 0x7F)| 0x80Size >> 0x0E |
| 0x200000-0x0FFFFFFF | (Size & 0x7F)| 0x80((Size >> 0x07) & 0x7F)| 0x80((Size >> 0x0E) & 0x7F)| 0x80Size >> 0x15 |
| 0x10000000-0x0FFFFFFFF | (Size & 0x7F)| 0x80((Size >> 0x07) & 0x7F)| 0x80((Size >> 0x0E) & 0x7F)| 0x80((Size >> 0x15) & 0x7F)| 0x80Size >> 0x1C |

In the preceding table, "&" refers to a bitwise "and" operation, "|" refers to a bitwise "or" operation, and ">>" refers to a right-shift operation.

### Property Records

The [**Property Records**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) contain metadata about the [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b). When Property Records are received, they set a [**protocol stream property**](#gt_4ddc88f0-31bc-4e87-8e77-3388f9f3d012) and affect the interpretation of the subsequent [**records**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) within the protocol stream.

#### Version Record

The Version Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) used to indicate which version of the .NET Message Framing Protocol is being used. The Version Record enables later versions of this specification to define additional record types and associated semantics.

The data portion of a Version Record is a pair of octets that indicate the major and minor version numbers. New sets of values for existing [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) types (for example, additional values of the Known Encoding Type Record) MUST be indicated by using a different minor version value. All other types of changes MUST be indicated with a different major version value.

The major and minor values of the Version Record denote the version of the framing format, not that of the payload envelope.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType | MajorVersion | MinorVersion |

**RecordType (1 byte):** This octet MUST be set to 0x00 to indicate that this record is a Version Record.

**MajorVersion (1 byte):** Specifies the major version of the .NET Message Framing Protocol. An implementation that conforms to this specification MUST set this field to 0x01. A value of 0x00 is not valid for this octet, and values of 0x02–0xff are reserved for future use.

**MinorVersion (1 byte):** Specifies the minor version of the .NET Message Framing Protocol. An implementation conforming to this specification MUST set this field to 0x00. The values 0x01 – 0xff for this octet are reserved for future use.[<3>](#Appendix_A_3" \o "Product behavior note 3)

#### Mode Record

The Mode Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that defines the communication mode for the session. The data portion of a Mode Record is a single octet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType | Mode |

**RecordType (1 byte):** This octet MUST be set to 0x01 to indicate that this is a Mode Record.

**Mode (1 byte):** The mode value MUST be set to one of the following values. A value of 0x00 is not valid for this octet, and values of 0x05–0xff are reserved for future use.

| Short Name | Meaning |
| --- | --- |
| Singleton-Unsized0x01 | The [**Initiating Stream**](#gt_0a414e88-de93-4303-8147-e177e7b48914) for a single one-way message or for a pair of messages in a request-reply manner between two nodes. |
| Duplex0x02 | The Initiating Stream for multiple bidirectional messages between two nodes. |
| Simplex0x03 | The Initiating Stream for multiple one-way messages from a single source. |
| Singleton-Sized0x04 | The Initiating Stream for a single one-way message from a single source. |

#### Via Record

The Via Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that defines the URI for which subsequent messages are bound. The data portion of a Via Record is of variable length.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType | ViaLength (variable) |
| ... |
| Via (variable) |
| ... |

**RecordType (1 byte):** This octet MUST be set to 0x02 to indicate that this is a Via Record.

**ViaLength (variable):** The value MUST be set to the size, in octets, of the Via, and encoded based on the scheme defined in section [2.2.2](#Section_070844defda143e389a0c9327f8f512e). The length MUST NOT be set to 0.

**Via (variable):** A URI (as defined in [[RFC2396]](https://go.microsoft.com/fwlink/?LinkId=90339) except that the "escaped" construct is never used). The URI MUST be encoded by using UTF-8, as specified in [[RFC2279]](https://go.microsoft.com/fwlink/?LinkId=90331).

#### Envelope Encoding Record

Envelope Encoding Records are the [**Property Records**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that define the encoding format that is used to encode the message envelope in subsequent [**Envelope Records**](#gt_db932f27-24c8-4fd8-a20f-b39334434637). Such [**records**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) come in two forms: [Known Encoding Records](#Section_a7d444632c60482d88569a8ff5929c62) and [Extensible Encoding Records](#Section_ac28de9318424d6680741c2a34e24720).

In messages, this record shows as variable-sized so that it can be either of the two forms. If the record uses Known Encoding, it is fixed-sized; otherwise, the record is variable-sized.

##### Known Encoding Record

The Known Encoding Record indicates a previously known encoding for the subsequent [**Envelope Records**](#gt_db932f27-24c8-4fd8-a20f-b39334434637). The data portion of this [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) is a single octet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType | Encoding |

**RecordType (1 byte):** This octet MUST be set to 0x03 to indicate that this is a Known Encoding Record.

**Encoding (1 byte):** This octet MUST be set to one of the following values. Values of 0x09–0xFF are reserved for future use.[<4>](#Appendix_A_4" \o "Product behavior note 4)

| SOAP Version 1.1 Value | Meaning |
| --- | --- |
| 0x00 | UTF-8, as specified in [[RFC2279]](https://go.microsoft.com/fwlink/?LinkId=90331). |
| 0x01 | UTF-16, as specified in [[RFC2781]](https://go.microsoft.com/fwlink/?LinkId=90380). |
| 0x02 | [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) [**little-endian**](#gt_079478cb-f4c5-4ce5-b72b-2144da5d2ce7). |

| SOAP Version 1.2 Value | Meaning |
| --- | --- |
| 0x03 | UTF-8. |
| 0x04 | UTF-16. |
| 0x05 | Unicode little-endian. |
| 0x06 | MTOM, as specified in [[SOAP-MTOM]](https://go.microsoft.com/fwlink/?LinkId=95126). |
| 0x07 | Binary, as specified in [[MC-NBFS]](%5BMC-NBFS%5D.pdf#Section_9567b35067884632aa34a87235e52459). |
| 0x08 | Binary with in-band dictionary, as specified in [[MC-NBFSE]](%5BMC-NBFSE%5D.pdf#Section_3caf1ffa63334c5f8aa7c06db4748c1b). |

##### Extensible Encoding Record

The Extensible Encoding Record indicates an ad hoc encoding for subsequent [**Envelope Records**](#gt_db932f27-24c8-4fd8-a20f-b39334434637). The [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) data in this case is a Multipurpose Internet Mail Extensions (MIME) content type, as specified in [[RFC2045]](https://go.microsoft.com/fwlink/?LinkId=90307), which is encoded by using UTF-8 encoding.[<5>](#Appendix_A_5" \o "Product behavior note 5)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| Record Type | Encoding Length (variable) |
| ... |
| Type (variable) |
| ... |
| Delimiter | Subtype (variable) |
| ... |
| Parameters (variable) |
| ... |

**Record Type (1 byte):** This octet MUST be set to 0x04 to indicate that this record is an Extensible Encoding Record.

**Encoding Length (variable):** The value MUST be set to the size, in octets, of the payload, and encoded based on the scheme that is specified in section [2.2.2](#Section_070844defda143e389a0c9327f8f512e). The length MUST NOT be set to 0.

**Type (variable):** This MUST be set to a type that is specified in [RFC2045] section 5.1.

**Delimiter (1 byte):** This MUST be set to the octet 0x2F (UTF-8 encoding for "/").

**Subtype (variable):** This MUST be set to a subtype that is specified in [RFC2045] section 5.1.

**Parameters (variable):** There can be one or more parameters in which the parameter structure is defined as follows.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| Parameter Delimiter | Parameter (variable) |
| ... |

**Parameter Delimiter (1 byte):** This MUST be set to the octet 0x3B (UTF-8 encoding for ";").

**Parameter (variable):** This MUST be set to a parameter as specified in [RFC2045] section 5.1.

#### Upgrade Request Record

The Upgrade Request Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that requests a protocol upgrade.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType | UpgradeProtocolLength (variable) |
| ... |
| UpgradeProtocol (variable) |
| ... |

**RecordType (1 byte):** This octet MUST be set to 0x09 to indicate that this is an Upgrade Request Record.

**UpgradeProtocolLength (variable):** This value MUST be set to the size, in octets, of the upgrade protocol name, encoded based on the scheme described in section [2.2.2](#Section_070844defda143e389a0c9327f8f512e). The length field MUST NOT be set to 0.

**UpgradeProtocol (variable):** The name of the protocol to upgrade to, encoded by using UTF-8. The following table identifies some known upgrade protocol names. An implementation SHOULD implement these upgrades and MAY define additional upgrade protocol definitions.[<6>](#Appendix_A_6" \o "Product behavior note 6)

| Protocol | Meaning |
| --- | --- |
| SSL/TLS"application/ssl-tls" | As defined in [[RFC5246]](https://go.microsoft.com/fwlink/?LinkId=129803). |
| Negotiate"application/negotiate" | As defined in [[RFC4178]](https://go.microsoft.com/fwlink/?LinkId=90461). |

#### Upgrade Response Record

The Upgrade Response Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that is sent in response to an [Upgrade Request Record](#Section_6ef3ede82ead42ceabf16d144271746f) to indicate a willingness to upgrade the [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b). This [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) has no data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType |

**RecordType (1 byte):** This octet MUST be set to 0x0A to indicate that this is an Upgrade Response Record.

#### Preamble End Record

The Preamble End Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that is sent to indicate the end of message properties. [**Envelope Records**](#gt_db932f27-24c8-4fd8-a20f-b39334434637) follow this [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71). This record has no data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType |

**RecordType (1 byte):** This octet MUST be set to 0x0C to indicate that this is a Preamble End Record.

#### Preamble Ack Record

The Preamble Ack Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that is sent to indicate receipt of a [Preamble End Record](#Section_2ac9c4efa6544b109cf8b5b8917c654e) and to indicate that all message properties and stream upgrades have been successfully applied. The [**receiving end**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) is now ready to receive the [**Envelope Records**](#gt_db932f27-24c8-4fd8-a20f-b39334434637). This [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) has no data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType |

**RecordType (1 byte):** This octet MUST be set to 0x0B to indicate that this is a Preamble Ack Record.

#### End Record

The End Record is a [**Property Record**](#gt_45e80254-eb1c-498b-bd99-a6cc613cad79) that indicates that communication over a connection has ended. This [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) has no data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType |

**RecordType (1 byte):** This octet MUST be set to 0x07 to indicate that this is an End Record.

### Envelope Records

An Envelope Record contains a message payload. There are two possible [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) types, depending on the message transfer mode.

#### Sized Envelope Record

A Sized Envelope Record contains a message of the specified size.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| Record Type | Size (variable) |
| ... |
| Payload (variable) |
| ... |

**Record Type (1 byte):** This octet MUST be set to 0x06 to indicate that this is a Sized Envelope Record.

**Size (variable):** The value MUST be set to the size, in octets, of the payload and encoded based on the scheme described in section [2.2.2](#Section_070844defda143e389a0c9327f8f512e). The size MUST NOT be set to 0.

**Payload (variable):** The content of the message encoded using the encoding indicated by an [Envelope Encoding Record](#Section_8d29c0f1b83c4778b62555b03f373084).

#### Data Chunk

A Data Chunk packet is used to transmit a portion of a message payload.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| Size (variable) |
| ... |
| Payload (variable) |
| ... |

**Size (variable):** The value MUST be set to the size, in octets, of the encoded payload, based on the scheme described in section [2.2.2](#Section_070844defda143e389a0c9327f8f512e). The size MUST NOT be set to 0.

**Payload (variable):** The content of the chunk.

#### Unsized Envelope Record

An Unsized Envelope Record contains a message that is encoded using the encoding indicated by an [Envelope Encoding Record](#Section_8d29c0f1b83c4778b62555b03f373084) that is broken into one or more data chunks of type Data Chunk (section [2.2.4.2](#Section_e72f9eb7eeaf48ce9c0750f7c5cd856b)). The end of this [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) is indicated by a single 0x00 octet in place of the start of the next data chunk.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType | DataChunk1 (variable) |
| ... |
| DataChunk2 (variable) |
| ... |
| Terminator |

**RecordType (1 byte):** This octet MUST be set to 0x05 to indicate that this is an Unsized Envelope Record.

**DataChunk1 (variable):** The first chunk of message data. This chunk MUST be present.

**DataChunk2 (variable):** Successive chunks of message data. Additional chunks MAY be present if the message is split across multiple chunks.

**Terminator (1 byte):** This field marks the end of chunks and MUST be set to 0x00.

### Fault Records

A Fault Record notifies the sender of an error encountered while processing a message frame. Generation of a Fault Record is informational only.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| RecordType | FaultSize (variable) |
| ... |
| Fault (variable) |
| ... |

**RecordType (1 byte):** This octet MUST be set to 0x08 to indicate that this is a Fault [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71).

**FaultSize (variable):** The value MUST be set to the size, in octets, of the fault, and encoded based on the scheme that is described in section [2.2.2](#Section_070844defda143e389a0c9327f8f512e). The size MUST NOT be set to 0.

**Fault (variable):** A URI (as defined by [[RFC2396]](https://go.microsoft.com/fwlink/?LinkId=90339) except that the "escaped" construct is never used). The URI is encoded by using UTF-8. The following table defines a collection of faults. An implementation MAY support these fault values and MAY also define new ones.[<7>](#Appendix_A_7" \o "Product behavior note 7)

For convenience, in this description the URI is broken into a namespace and fault name. The namespace for faults in the following table is http://schemas.microsoft.com/ws/2006/05/framing/faults/. Any additional faults that are defined MUST NOT use this namespace.

An example of a fault, as returned in a Fault Record, is the following: http://schemas.microsoft.com/ws/2006/05/framing/faults/UnsupportedMode

| Fault name values | Meaning |
| --- | --- |
| "ConnectionDispatchFailed" | The [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) that is referenced by the [Via Record](#Section_2d61448ab4d14cbd911cd5caff54b64f) exists; however, the attempt to dispatch the message to the endpoint failed. |
| "ContentTypeInvalid" | The [Envelope Encoding Record](#Section_8d29c0f1b83c4778b62555b03f373084) that was sent is not supported by the endpoint. |
| "ContentTypeTooLong" | The [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) is enforcing a maximum content-type size, and the Envelope Encoding Record exceeded that quota. |
| "EndpointAccessDenied" | The endpoint that is referenced by the Via Record cannot be accessed. |
| "EndpointNotFound" | The endpoint that is referenced by the Via Record cannot be found. |
| "EndpointPaused" | The endpoint that is referenced by the Via Record exists; however, the endpoint is currently paused. |
| "EndpointUnavailable" | The endpoint that is referenced by the Via Record exists; however, the endpoint is currently unavailable. |
| "InvalidRecordSequence" | The record sequence does not conform to the grammar that is outlined in section [3.1.1.2](#Section_f651a506e0904abaa2cbfceeb5ff21a5). |
| "MaxMessageSizeExceededFault" | The receiver is enforcing a maximum message size, and the incoming message has exceeded that quota. |
| "ServerTooBusy" | The endpoint does not have sufficient resources to process the connection. |
| "ServiceActivationFailed" | The endpoint is in a process that cannot be activated. |
| "UnsupportedMode" | The [Mode Record](#Section_8cd0b687a7fc45e5b328e00225836af3) value is not supported by the destination. |
| "UnsupportedVersion" | The [Version Record](#Section_b8dfbcd6b65b495aa1f1e19b78897a3d) value is not supported by the destination. |
| "UpgradeInvalid" | The requested upgrade is not supported by the remote endpoint. |
| "ViaTooLong" | The receiver is enforcing a maximum Via size, and the Via Record exceeded that quota. |

### Preamble Message

To aid description, a Preamble Message is defined for an initial [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) sequence. The Preamble Message can apply to multiple messages, depending on the mode specified.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 |
| VersionRecord | ModeRecord |
| ... | ViaRecord (variable) |
| ... |
| EnvelopeEncodingRecord (variable) |
| ... |

**VersionRecord (3 bytes):** This field MUST be formatted as specified in section [2.2.3.1](#Section_b8dfbcd6b65b495aa1f1e19b78897a3d).

**ModeRecord** **(2 bytes):** This field MUST be formatted as specified in section [2.2.3.2](#Section_8cd0b687a7fc45e5b328e00225836af3).

**ViaRecord (variable):** This field MUST be formatted as specified in section [2.2.3.3](#Section_2d61448ab4d14cbd911cd5caff54b64f).

**EnvelopeEncodingRecord (variable):** This field MUST be formatted as specified in section [2.2.3.4](#Section_8d29c0f1b83c4778b62555b03f373084).

# Protocol Details

A node that is a participant in this protocol can behave in one of two roles:

* [**Initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab)
* [**Receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37)

An initiator initiates the protocol by sending a preamble message to the receiver. The initiator and receiver nodes then send and receive messages using the [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b) that connects the two [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee).

## Common Details

### Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the participants behave. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with what is described in this document.

The participant maintains the following state for each session:

* Protocol Configuration Object (PCO) - Determines the specific transport, protocol version, mode, [Via](#Section_2d61448ab4d14cbd911cd5caff54b64f), and message-encoding scheme to be used for this session.
* Send Allowed - A Boolean value that can be set to TRUE or FALSE to indicate whether messages can be sent on this session.
* Receive Allowed - A Boolean value that can be set to TRUE or FALSE to indicate whether messages can be received on this session.

#### Initiator-Receiver Interactions

This section describes some typical interactions between an [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) and [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37).

##### Singleton Unsized Mode



Figure 3: Singleton Unsized mode

##### Duplex Mode



Figure 4: Duplex mode

In the case illustrated, the [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) sends the [End Record](#Section_9d691d25fd3a41ca9eb5241f47a914e5) first. The protocol allows either participant to send the End Record first. After a participant sends the End Record, the participant MUST continue to receive messages until the session is closed.

##### Simplex Mode



Figure 5: Simplex mode

##### Singleton Sized Mode



Figure 6: Singleton Sized mode

##### Upgrades



Figure 7: Upgrades

This figure illustrates a stream upgrade that uses the [Singleton Unsized mode](#Section_de14e6f221bf43f69301bb9e4b690931). The figure would look very similar if the stream upgrade used the [Duplex mode](#Section_81875ae72d1649baa260ffd996e4c3d6).

After the protocol upgrade, subsequent protocol exchanges occur over the upgraded transport stream until a fault occurs or an [End Record](#Section_9d691d25fd3a41ca9eb5241f47a914e5) is received. Although the protocol allows for multiple upgrades, the preceding exchange illustrates a single upgrade only.

##### Faults



Figure 8: Unsupported version



Figure 9: Upgrade invalid



Figure 10: Maximum message size exceeded

The preceding exchanges capture some of the scenarios where a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) can be generated.

#### Protocol Grammar

This section uses the Augmented Backus-Naur Form (ABNF) notation that is specified in [[RFC2234]](https://go.microsoft.com/fwlink/?LinkId=90323) to define the [**protocol stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b) grammar. ProtocolStream-a represents the stream of octets flowing from the [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) to the [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37), and ProtocolStream-b represents the stream of octets flowing from the receiver to the initiator.

1. ProtocolStream-a = 1\*(SingletonUnsizedStream-a / DuplexStream-a / SimplexStream-a / SingletonSizedStream-a)
2. ProtocolStream-b = 1\*(SingletonUnsizedStream-b / DuplexStream-b)
3. SingletonUnsizedStream-a = VersionRecord ModeRecordType SingletonUnsizedMode ViaRecord EncodingRecord \*UpgradeRequest PreambleEndRecord UnsizedEnvelopeRecord EndRecord
4. DuplexStream-a = VersionRecord ModeRecordType DuplexMode ViaRecord EncodingRecord \*UpgradeRequest PreambleEndRecord \*SizedEnvelopeRecord EndRecord
5. SimplexStream-a = VersionRecord ModeRecordType SimplexMode ViaRecord EncodingRecord PreambleEndRecord \*SizedEnvelopeRecord EndRecord
6. SingletonSizedStream-a = VersionRecord ModeRecordType SingletonSizedMode ViaRecord EncodingRecord Octets
7. SingletonUnsizedStream-b = (\*UpgradeResponse FaultRecord) / (\*UpgradeResponse PreambleAckRecord \*1UnsizedEnvelopeRecord (FaultRecord / EndRecord))
8. DuplexStream-b = (\*UpgradeResponse FaultRecord) / (\*UpgradeResponse PreambleAckRecord \*SizedEnvelopeRecord (FaultRecord / EndRecord))
9. EncodingRecord = KnownEncodingRecord / ExtensibleEncodingRecord
10. UpgradeRequest = UpgradeRequestRecord Octets
11. UpgradeResponse = UpgradeResponseRecord Octets
12. VersionRecord = VersionRecordType MajorVersionNumber MinorVersionNumber
13. VersionRecordType = %x00
14. MajorVersionNumber = %x01
15. MinorVersionNumber = %x00
16. ModeRecordType = %x01
17. SingletonUnsizedMode = %x01
18. DuplexMode = %x02
19. SimplexMode = %x03
20. SingletonSizedMode = %x04
21. ViaRecord = ViaRecordType EncodedSize Utf8Octets
22. ViaRecordType = %x02
23. KnownEncodingRecord = KnownEncodingRecordType KnownEncodingType
24. KnownEncodingType = TextEncoding / BinaryEncoding / MtomEncoding
25. BinaryEncoding = BinarySessionlessEncoding / BinarySessionEncoding
26. TextEncoding = Soap11TextEncoding / Soap12TextEncoding
27. Soap11TextEncoding = Soap11Utf8Encoding / Soap11Utf16Encoding / Soap11UnicodeFFFEEncoding
28. Soap12TextEncoding = Soap12Utf8Encoding / Soap12Utf16Encoding / Soap12UnicodeFFFEEncoding
29. KnownEncodingRecordType = %x03
30. Soap11Utf8Encoding = %x00
31. Soap11Utf16Encoding = %x01
32. Soap11UnicodeFFFEEncoding = %x02
33. Soap12Utf8Encoding = %x03
34. Soap12Utf16Encoding = %x04
35. Soap12UnicodeFFFEEncoding = %x05
36. MtomEncoding = %x06
37. BinarySessionlessEncoding = %x07
38. BinarySessionEncoding = %x08
39. ExtensibleEncodingRecord = ExtensibleEncodingRecordType EncodedSize Utf8Octets
40. ExtensibleEncodingRecordType = %x04
41. UnsizedEnvelopeRecord = UnsizedEnvelopeRecordType 1\*(EncodedSize Octets) Terminator
42. UnsizedEnvelopeRecordType = %x05
43. Terminator = %x00
44. SizedEnvelopeRecord = SizedEnvelopeRecordType EncodedSize Octets
45. SizedEnvelopeRecordType = %x06
46. EndRecord = EndRecordType
47. EndRecordType = %x07
48. FaultRecord = FaultRecordType EncodedSize Utf8Octets
49. FaultRecordType = %x08
50. UpgradeRequestRecord = UpgradeRequestRecordType EncodedSize Utf8Octets
51. UpgradeRequestRecordType = %x09
52. UpgradeResponseRecord = UpgradeResponseRecordType
53. UpgradeResponseRecordType = %x0A
54. PreambleAckRecord = PreambleAckRecordType
55. PreambleAckRecordType = %x0B
56. PreambleEndRecord = PreambleEndRecordType
57. PreambleEndRecordType = %x0C
58. Utf8Octets = 1\*(Utf8Octet)
59. Utf8Octet = %x00-7F / %xC2-DF %x80-BF / %xE0-EF %x80-BF %x80-BF / %xF0-F4 %x80-BF %x80-BF %x80-BF
60. Octets = 1\*(%x00-FF)
61. EncodedSize = %x01-7F / %x80-FF %x01-7F / %x80-FF %x80-FF %x01-7F / %x80-FF %x80-FF %x80-FF %x01-7F / %x80-FF %x80-FF %x80-FF %x80-FF %x01-07

### Timers

None.

### Initialization

The **PCO** is made available to the participant as part of a higher-layer triggered event.

When the participant is initialized:

* The **Send Allowed** field MUST be set to FALSE.
* The **Receive Allowed** field MUST be set to FALSE.

### Higher-Layer Triggered Events

This section covers reading [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) types from the underlying transport. The higher-layer triggered events and related processing are role specific.

The following stipulations apply throughout the remaining sections.

Wherever it is mentioned that a session MUST be closed, it refers to the following actions being taken by the participant:

* Any session-related state MUST be discarded.
* The participant MUST notify the higher layer of the error.

Wherever it is mentioned that a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) MAY (or SHOULD) be sent, it refers to the following action being taken by the participant:

* If the mode is Singleton Unsized, or Duplex mode, a Fault Record MAY (or SHOULD) be sent, as described in section 2.2.5.

#### Reading Variable-Sized Records

When a variable-sized [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) is received, the participant MUST use the following algorithm to decode the size and read the payload. This section assumes that the record type has already been read.

The algorithm takes as input the MaxSize, that is, the maximum supported size for this record. If the encoded size is 0, a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) MAY[<8>](#Appendix_A_8" \o "Product behavior note 8) be sent to indicate that the size is 0 and the session MUST be closed. The decoded size is returned in [**little-endian**](#gt_079478cb-f4c5-4ce5-b72b-2144da5d2ce7) format.



Figure 11: Algorithm to decode the size and read the payload

#### Handling Receipt of an Unexpected Record Type

If the participant receives an unexpected [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) type, it MUST be handled as follows:

* If the record type is not [Fault Record](#Section_337e8351085448d2864b97520026c2c6), a Fault Record MAY be sent to indicate that an unexpected record type has been received.
* The session MUST be closed.

#### Version Record

* If the record type the participant read from the protocol stream is not [Version Record](#Section_b8dfbcd6b65b495aa1f1e19b78897a3d), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* The participant MUST read the next two octets, which contain the major and minor versions of the protocol being used.
* If the participant does not recognize the version, a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) MAY[<9>](#Appendix_A_9" \o "Product behavior note 9) be sent to indicate that an incorrect version was specified, and the session MUST be closed.

#### Mode Record

* If the record type the participant read from the protocol stream is not [Mode Record](#Section_8cd0b687a7fc45e5b328e00225836af3), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* The participant MUST read the next octet, which contains the mode.
* If the mode is incorrect for the session, a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) MAY[<10>](#Appendix_A_10" \o "Product behavior note 10) be sent to indicate that an incorrect mode has been specified, and the session MUST be closed.

#### Via Record

* If the record type the participant read from the protocol stream is not a [Via Record](#Section_2d61448ab4d14cbd911cd5caff54b64f), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* The participant MUST obtain the Via, as detailed in section [3.1.4.1](#Section_24e76a050dbd4aa49594678f12a01f3a). The participant SHOULD use a MaxViaSize.[<11>](#Appendix_A_11" \o "Product behavior note 11) If the Via is too long, a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) MAY[<12>](#Appendix_A_12" \o "Product behavior note 12) be sent, and the session MUST be closed.
* If the participant cannot locate an [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) that matches the Via, a Fault Record MAY[<13>](#Appendix_A_13" \o "Product behavior note 13) be sent, and the session MUST be closed.

#### Encoding Record

* If the record type the participant read from the protocol stream is not Known Encoding Record or Extensible Encoding Record, it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* If the record type is Known Encoding Record, the participant MUST read the next octet, which contains the message encoding scheme.
* If the record type is Extensible Encoding Record, the participant MUST obtain the encoding scheme, as detailed in section [3.1.4.1](#Section_24e76a050dbd4aa49594678f12a01f3a). The participant SHOULD use a MaxContentTypeSize.[<14>](#Appendix_A_14" \o "Product behavior note 14) If the content type is too long, a Fault Record MAY[<15>](#Appendix_A_15" \o "Product behavior note 15) be sent, and the session MUST be closed.
* If the encoding is not supported, a Fault Record MAY[<16>](#Appendix_A_16" \o "Product behavior note 16) be sent, and the session MUST be closed.

#### Upgrade Request Record

* If the record type the participant read from the protocol stream is not [Upgrade Request Record](#Section_6ef3ede82ead42ceabf16d144271746f), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* The participant MUST read the Upgrade Protocol, as detailed in section [3.1.4.1](#Section_24e76a050dbd4aa49594678f12a01f3a). The participant SHOULD use a MaxUpgradeProtocolSize.[<17>](#Appendix_A_17" \o "Product behavior note 17) If the upgrade name is too long, a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) MAY[<18>](#Appendix_A_18" \o "Product behavior note 18) be sent, and the session MUST be closed.
* If the upgrade is not supported, a Fault Record MAY[<19>](#Appendix_A_19" \o "Product behavior note 19) be sent, and the session MUST be closed.
* If the upgrade is supported, the participant MUST send an [Upgrade Response Record](#Section_8f9da3a35345482cbcb7543972c1ee4a), as described in section 2.2.3.6. The participant MUST invoke the upgrade handler identified by the upgrade protocol name in the Upgrade Request Record.

#### Upgrade Response Record

* If the record type the participant read from the protocol stream is not [Upgrade Response Record](#Section_8f9da3a35345482cbcb7543972c1ee4a), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* If the upgrade is supported, the participant MUST invoke the appropriate upgrade handler. How the upgrade handler achieves the upgrade is outside the scope of this document.

#### Preamble End Record

* If the record type the participant read from the protocol stream is not [Preamble End Record](#Section_2ac9c4efa6544b109cf8b5b8917c654e), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* In the case of Singleton Unsized and Duplex modes, the participant MUST send a [Preamble Ack Record](#Section_3ec275c549e04becb7515cb572c3e7ab) as described in section 2.2.3.8.

#### Preamble Ack Record

If the record type the participant read from the protocol stream is not [Preamble Ack Record](#Section_3ec275c549e04becb7515cb572c3e7ab), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).

#### Sized Envelope Record

* If the record type the participant read from the protocol stream is not [Sized Envelope Record](#Section_82159b784bbe4ca8a707eeb65d3c6173), it MUST be handled as follows:
	+ If the record type is [End Record](#Section_9d691d25fd3a41ca9eb5241f47a914e5), the participant MUST notify the higher layer of the receipt of End Record and set Receive Allowed to FALSE.
	+ If the record type is a [Fault Record](#Section_337e8351085448d2864b97520026c2c6), the session MUST be closed.
	+ Otherwise, it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* The participant MUST obtain the message as detailed in section [3.1.4.1](#Section_24e76a050dbd4aa49594678f12a01f3a). The participant SHOULD use a MaxEnvelopeSize.[<20>](#Appendix_A_20" \o "Product behavior note 20)

If the message is too large, a Fault Record MAY[<21>](#Appendix_A_21" \o "Product behavior note 21) be sent, and the session MUST be closed.

#### Unsized Envelope Record

* If the record type the participant read from the protocol stream is not [Unsized Envelope Record](#Section_5e46160dc9384882bf160477449e61e3), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* The participant MUST then process the first chunk and any additional chunks, as described in section [3.1.4.1](#Section_24e76a050dbd4aa49594678f12a01f3a), until the Terminator marker (octet 0x00) is read. To achieve streaming, reading chunks SHOULD be correlated with consumption of chunks by the higher layer. The participant SHOULD use a MaxChunkSize.[<22>](#Appendix_A_22" \o "Product behavior note 22) If the chunk size is too large, a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) MAY[<23>](#Appendix_A_23" \o "Product behavior note 23) be sent, and the session MUST be closed.

#### End Record

* If the record type the participant read from the protocol stream is not [End Record](#Section_9d691d25fd3a41ca9eb5241f47a914e5), it MUST be handled as described in section [3.1.4.2](#Section_4f257622d8364777ba43472ec928416f).
* The participant MUST set Receive Allowed to FALSE.

### Message Processing Events and Sequencing Rules

This document assumes that the processing of received octets is deferred until initiated by a higher-layer triggered event or a required response in the protocol. All message processing events and sequencing rules are explained in the context of higher-layer triggered events.

### Timer Events

None.

### Other Local Events

#### Underlying Transport Session Is Closed

If at any point, the underlying network transport session is closed, the [**Protocol Stream**](#gt_894895dc-81f8-468c-b9e4-08b4c004ed9b) is closed. The participant MUST discard any session-related state.

## Initiator Details

### Abstract Data Model

The details are covered in section [3.1.1](#Section_ce2c4bdc84c94a5fbb131fcadd68bb9c).

### Timers

None.

### Initialization

The details are covered in section [3.1.3](#Section_6046b1b2de1544c684182d1a1a11ad67).

### Higher-Layer Triggered Events

The operation of the [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) is driven by the following higher-layer triggered events.

#### Initialize Session

A new session state MUST be created, and session properties initialized as described in section [3.1.3](#Section_6046b1b2de1544c684182d1a1a11ad67).

#### Send Preamble

* The [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST send the Preamble Message as described in section [2.2.6](#Section_f7b0315d48434376bf66265e5756cb5c).
* In the case of Simplex mode, the initiator MUST send the Preamble End record as described in section [2.2.3.7](#Section_2ac9c4efa6544b109cf8b5b8917c654e).
* In the case of Singleton Unsized, and Duplex modes, the initiator MUST perform the following additional steps:
	+ If an upgrade is required, send the Upgrade Request Record as described in section [2.2.3.5](#Section_6ef3ede82ead42ceabf16d144271746f).
	+ If an upgrade is sent, read the Upgrade Response Record as described in section [3.1.4.8](#Section_7b8d1f6983904455aad22abf24831eb8).
	+ Send the Preamble End Record as described in section 2.2.3.7.
	+ Read the Preamble Ack Record as described in section [3.1.4.10](#Section_a013f71801604f238654bb5880536ed7).
* The initiator MUST set Send Allowed to TRUE.
* If the mode is Duplex, the initiator MUST set Receive Allowed to TRUE.

#### Send Message

If Send Allowed is set to FALSE, an error MUST be propagated to the higher layer, and no further processing done. Otherwise, the [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST do the following based on the mode.

##### Singleton Unsized Mode

* The [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST send an Unsized Envelope Record containing the message as described in section [2.2.4.3](#Section_5e46160dc9384882bf160477449e61e3).
* The initiator MUST send an End Record as described in section [2.2.3.9](#Section_9d691d25fd3a41ca9eb5241f47a914e5).
* The initiator MUST set Send Allowed to FALSE.

##### Duplex or Simplex Mode

The [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST send a Sized Envelope Record containing the message as described in section [2.2.4.1](#Section_82159b784bbe4ca8a707eeb65d3c6173).

##### Singleton Sized Mode

The [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST send the message and set Send Allowed to FALSE.

#### Receive Message

If Receive Allowed is set to FALSE, an error MUST be propagated to the higher layer and no further processing done. Otherwise, the [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST read a Sized Envelope Record as described in section [3.1.4.11](#Section_f1805a21c1614364b82808121a5ee18d), and propagate the contained message to a higher layer.

#### Send End Record

If mode is not Duplex or Simplex, an error MUST be propagated to the higher layer and no further processing done. Otherwise, the [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST send an End Record as described in section [2.2.3.9](#Section_9d691d25fd3a41ca9eb5241f47a914e5). The initiator MUST set Send Allowed to FALSE.

#### Session Close

The [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) MUST discard any session-related state and no further processing done.

### Message Processing Events and Sequencing Rules

The details are covered in section [3.1.5](#Section_f7653f834f8846e2812327c8949ff24e).

### Timer Events

None.

### Other Local Events

The details are covered in section [3.1.7](#Section_4d3909e2544241009e48f327b944c580).

## Receiver Details

### Abstract Data Model

The details are covered in section [3.1.1](#Section_ce2c4bdc84c94a5fbb131fcadd68bb9c).

### Timers

None.

### Initialization

The details are covered in section [3.1.3](#Section_6046b1b2de1544c684182d1a1a11ad67).

### Higher-Layer Triggered Events

The operation of the receiver is driven by the following higher-layer triggered events.

#### Initialize Session

A new session state MUST be created and session properties initialized as described in section [3.1.3](#Section_6046b1b2de1544c684182d1a1a11ad67).

#### Receive Preamble

* The [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST read the Version Record, as described in section [3.1.4.3](#Section_41938831569840c0829173a2ee107a39).
* The receiver MUST read the Mode Record, as described in section [3.1.4.4](#Section_ee2314e797d64753bed766d2768097ff).
* The receiver MUST read the Via Record, as described in section [3.1.4.5](#Section_8ca81a2267c346ce8c77eab9fe9100a4).
* The receiver MUST read the Encoding Record, as described in section [3.1.4.6](#Section_c80201ae35af443abf23d49997cb481b).
* If the mode is Simplex, the receiver MUST read the Preamble End record as described in section [3.1.4.9](#Section_3a30f5e662e84a27ab614cbfed066419).
* If the mode is Singleton Unsized, or Duplex, the receiver MUST perform these additional steps:
	+ If an upgrade is required, read the Upgrade Request Record, as described in section [3.1.4.7](#Section_bbec3f6b22474a2591abf44d0437e0de).
	+ Read the Preamble End Record, as described in section 3.1.4.9.
* The receiver MUST set Receive Allowed to TRUE.
* If the mode is Duplex, the receiver MUST set Send Allowed to TRUE.

#### Send Message

If Send Allowed is set to FALSE, an error MUST be propagated to the higher layer and no further processing done. Otherwise, the [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST send a Sized Envelope Record containing the message as described in section [2.2.4.1](#Section_82159b784bbe4ca8a707eeb65d3c6173).

#### Receive Message

If Receive Allowed is set to FALSE, an error MUST be propagated to the higher layer and no further processing done. Otherwise, the [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST do the following based on the Mode.

##### Singleton Unsized Mode

* The [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST read an Unsized Envelope Record as described in section [3.1.4.12](#Section_03664f95c9d44101ba5de9fb5b329e6b), and propagate the contained message to a higher layer.
* The receiver MUST read an End Record as described in section [3.1.4.13](#Section_460346a9710544b287248cc64750d155).
* The receiver MUST set Receive Allowed to FALSE.

##### Duplex or Simplex Mode

The [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST read a Sized Envelope Record as described in section [3.1.4.11](#Section_f1805a21c1614364b82808121a5ee18d), and propagate the contained message to a higher layer.

##### Singleton Sized Mode

The [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST read the message and propagate it to a higher layer. The receiver MUST set Receive Allowed to FALSE.

#### Send End Record

If the mode is not Duplex, an error MUST be propagated to the higher layer and no further processing done. Otherwise, the [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST send an End Record as described in section [2.2.3.9](#Section_9d691d25fd3a41ca9eb5241f47a914e5). The receiver MUST set Send Allowed to FALSE.

#### Session Close

The [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37) MUST discard any session-related state and no further processing done.

### Message Processing Events and Sequencing Rules

The details are covered in section [3.1.5](#Section_f7653f834f8846e2812327c8949ff24e).

### Timer Events

None.

### Other Local Events

The details are covered in section [3.1.7](#Section_4d3909e2544241009e48f327b944c580).

# Protocol Examples

## Duplex Mode

The protocol exchange involving a Duplex Mode session is illustrated in this section. The [**initiator**](#gt_c66d2bda-f3e1-4d01-aa9e-8d790d3be5ab) first establishes a session with the [**receiver**](#gt_0ff4678e-e7a2-44ae-a29a-0a561d7d3d37). The initiator then sends a message, and the receiver replies. Finally, the session is closed. The Protocol Configuration Object for this session has been configured as follows:

* Transport - The specifics of network transport are excluded from this example. The following packet captured demonstrates only the .NET Message Framing Protocol and message payload.
* Version - This exchange happened over Major Version 1 and Minor Version 0 of this protocol.
* Mode - Duplex mode was used.
* Via - The receiver was identified by the URI net.tcp://SampleServer/SampleApp/.
* Encoding - Binary Session Encoding was used to encode the messages.

### Initiator Receiver: Preamble Message



Figure 12: Initiator Receiver: Preamble Message

### Initiator Receiver: Preamble End Message



Figure 13: Initiator Receiver: Preamble End Message

### Receiver Initiator : Preamble Ack Message



Figure 14: Receiver Initiator : Preamble Ack Message

### Initiator Receiver: Sized Envelope Message



Figure 15: Initiator Receiver: Sized Envelope Message

### Receiver Initiator: Sized Envelope Message



Figure 16: Receiver Initiator: Sized Envelope

### Initiator Receiver: End Message



Figure 17: Initiator Receiver: End Message

### Receiver Initiator: End Message



Figure 18: Receiver Initiator: End Message

# Security

## Security Considerations for Implementers

To minimize the risk of a denial-of-service (DOS) attack, it is recommended that an implementation of this protocol limit the size of variable-length [**records**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71), including Via, Extensible Encoding, Upgrade Protocol, Sized Envelope, and Unsized Envelope Record chunks. Note that Via, Extensible Encoding, and Upgrade Protocol records are exchanged before a stream upgrade can supply transport level security. Therefore, particular care needs to be taken to limit these records to a reasonable size if security is not available.

## Index of Security Parameters

None.

# Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

This document specifies version-specific details in the Microsoft .NET Framework. For information about which versions of .NET Framework are available in each released Windows product or as supplemental software, see [[MS-NETOD]](%5BMS-NETOD%5D.pdf#Section_bcca8164da0843f2a983c34ed99171b0) section 4.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

* Microsoft .NET Framework 3.0
* Microsoft .NET Framework 3.5
* Microsoft .NET Framework 4.0
* Microsoft .NET Framework 4.5
* Microsoft .NET Framework 4.6
* Microsoft .NET Framework 4.7
* Microsoft .NET Framework 4.8

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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.

[<1> Section 1.3](#Appendix_A_Target_1): The Windows implementation of this protocol is exercised through the use of the following Windows Communication Framework bindings [[MSDN-WCF]](https://go.microsoft.com/fwlink/?LinkId=100290).

1. NetTcpBinding [[MSDN-NETTcp]](https://go.microsoft.com/fwlink/?LinkId=100283) - If the TransferMode property on the binding is set to Buffered, the mode is set to Duplex. Otherwise, the mode is set to Singleton Unsized. If the Security.Transport.ClientCredentialType property on the binding is set to Certificate, the "SSL/TLS" upgrade protocol is used. Otherwise, if it is set to Windows, the "Negotiate" upgrade protocol is used.
2. NetNamedPipeBinding [[MSDN-NETNamedPipe]](https://go.microsoft.com/fwlink/?LinkId=100285) - If the TransferMode property on the binding is set to Buffered, the mode is set to Duplex. Otherwise, the mode is set to Singleton Unsized. If the Security.Mode property on the binding is set to Transport, the "Negotiate" upgrade protocol is used.
3. NetMsmqBinding [[MSDN-NETMsmq]](https://go.microsoft.com/fwlink/?LinkId=100286) - If a TransactionScope is being used, the mode is set to Simplex. Otherwise, the mode is set to Singleton Sized. If the Security.Transport.MsmqAuthenticationMode property on the binding is set to Certificate, the "SSL/TLS" upgrade protocol is used. Otherwise, if it is set to WindowsDomain, the "Negotiate" upgrade protocol is used.

The Windows implementation of this protocol is also exercised through a custom Windows Communication Framework binding that uses the TcpTransportBindingElement [[MSDN-NETTcpBE]](https://go.microsoft.com/fwlink/?LinkId=100287) or the NamedPipeTransportBindingElement [[MSDN-NETNamedPipeBE]](https://go.microsoft.com/fwlink/?LinkId=100288), or the MsmqTransportBindingElement [[MSDN-NETMsmqBE]](https://go.microsoft.com/fwlink/?LinkId=100289).

The Windows implementation of this protocol is also exercised through the use of the following Windows Web Services API channel binding [[MSDN-WSCHBIND]](https://go.microsoft.com/fwlink/?LinkId=182220):

1. WS\_TCP\_CHANNEL\_BINDING - If channel binding is set to WS\_TCP\_CHANNEL\_BINDING, the mode is always set to Duplex. If channel security binding [[MSDN-WSSECBIND]](https://go.microsoft.com/fwlink/?LinkId=182275) is set to WS\_TCP\_SSPI\_TRANSPORT\_SECURITY\_BINDING [[MSDN-WSTCPSSPI]](https://go.microsoft.com/fwlink/?LinkId=182221), the "Negotiate" upgrade protocol is used.

[<2> Section 2.2.2](#Appendix_A_Target_2): The Windows implementation of the protocol that is exercised by Windows Communication Foundation will not allow record sizes larger than 0x7fffffff octets.

[<3> Section 2.2.3.1](#Appendix_A_Target_3): The Windows implementation of this protocol that is exercised by Windows Communication Foundation does not validate the value of the minor version when the value of the major version is 0x01.

[<4> Section 2.2.3.4.1](#Appendix_A_Target_4): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API supports all the known encoding schemes.

[<5> Section 2.2.3.4.2](#Appendix_A_Target_5): The .NET Framework 4.5 and later implementations of this protocol that are exercised by Windows Communication Foundation use the Extensible Encoding Record to indicate the MIME content type for binary message encoding compression (see [[MSDN-BinaryMsgEncdngBindElmnt]](https://go.microsoft.com/fwlink/?LinkId=216916)).

[<6> Section 2.2.3.5](#Appendix_A_Target_6): The Windows implementation of this protocol that is exercised by Windows Communication Framework supports only the SSL/TLS and Negotiate upgrade protocols.

The Windows implementation of this protocol that is exercised by Windows Web Services API supports only the Negotiate upgrade protocol.

[<7> Section 2.2.5](#Appendix_A_Target_7): The Windows implementation of this protocol that is exercised by Windows Communication Framework supports the following set of faults: ContentTypeInvalid, ContentTypeTooLong, ConnectionDispatchFailed, EndpointNotFound, EndpointUnavailable, MaxMessageSizeExceededFault, ServerTooBusy, ServiceActivationFailed, UnsupportedMode, UnsupportedVersion, UpgradeInvalid, and ViaTooLong.

The Windows implementation of this protocol that is exercised by Windows Web Services API supports the following set of faults: ContentTypeInvalid, EndpointNotFound, MaxMessageSizeExceededFault, UnsupportedMode, and UpgradeInvalid.

[<8> Section 3.1.4.1](#Appendix_A_Target_8): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API does not send a [Fault Record](#Section_337e8351085448d2864b97520026c2c6) if the size of a variable-sized [**record**](#gt_dca3e776-890f-48c8-be62-094a5f2fcf71) is 0.

[<9> Section 3.1.4.3](#Appendix_A_Target_9): The Windows implementation of this protocol that is exercised by Windows Communication Framework sends a Fault Record (UnsupportedVersion) if an incorrect version is specified in the received [Version Record](#Section_41938831569840c0829173a2ee107a39).

The Windows implementation of this protocol that is exercised by Windows Web Services API does not send a Fault Record if an incorrect version is specified in the received Version Record.

[<10> Section 3.1.4.4](#Appendix_A_Target_10): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API sends a Fault Record (UnsupportedMode) if an incorrect mode is specified in the received [Mode Record](#Section_8cd0b687a7fc45e5b328e00225836af3).

[<11> Section 3.1.4.5](#Appendix_A_Target_11): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API defines a MaxViaSize of 2,048 bytes.

[<12> Section 3.1.4.5](#Appendix_A_Target_12): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API does not send a Fault Record if the size of Via in the received [Via Record](#Section_2d61448ab4d14cbd911cd5caff54b64f) exceeds MaxViaSize.

[<13> Section 3.1.4.5](#Appendix_A_Target_13): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API sends a Fault Record (EndpointNotFound) if the [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) cannot be located for the specified Via in the received Via Record.

The Windows implementation of this protocol that is exercised by Windows Communication Framework sends a Via with a scheme component that is equal to "net.tcp" if exercised with NetTcpBinding (see [MSDN-NETTcp]) or TcpTransportBindingElement (see [MSDN-NETTcpBE]); and a Via with a scheme component that is equal to "net.msmq" if exercised with NetMsmqBinding (see [MSDN-NETMsmq]) or MsmqTransportBindingElement (see [MSDN-NETMsmqBE]).

A Via that has a scheme equal to "net.tcp" or "net.msmq" uses the following constructions: the URI reference is absolute, the URI contains a hierarchical part, the hierarchical part contains a network path, the authority is a server, and the server does not include user information.

The Windows implementation of this protocol that is exercised by Windows Web Services API sends a Via with a scheme component equal to "net.tcp" if exercised with WS\_TCP\_CHANNEL\_BINDING [MSDN-WSCHBIND]. A Via with a scheme equal to "net.tcp" uses the following constructions: the URI reference is absolute, the URI contains a hierarchical part, the hierarchical part contains a network path, the authority is a server, and the server does not include user information.

 The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API supports attempting to locate an endpoint for a specified Via with a scheme component that is equal to "net.tcp" when the transport session (as described in section [1.5](#Section_6a1575de73974a94a2f499ed9c83999c)) that is carrying the protocol stream is a TCP connection (as defined in [[RFC793]](https://go.microsoft.com/fwlink/?LinkId=150872)) whose destination address is equal to the authority of the Via; however, an authority that does not designate a port is equivalent to an authority that uses port 808.

 The Windows implementation of this protocol that is exercised by Windows Communication Framework supports attempting to locate an endpoint for a specified Via with a scheme component equal to "net.msmq" when the initiator is Microsoft Message Queuing, as specified in [[MS-MQMQ]](%5BMS-MQMQ%5D.pdf#Section_b7cc2590a61745dfb6a31f31102b36fb), whose queue path name computer is equal to the authority of the Via and the remainder of whose queue path name is equal to the absolute path of the Via, except that the first path segment in the Via of a private queue is "private" rather than "private$".

[<14> Section 3.1.4.6](#Appendix_A_Target_14): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API defines a MaxContentTypeSize of 256 bytes.

[<15> Section 3.1.4.6](#Appendix_A_Target_15): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API does not send a Fault Record if the size of the extensible encoding in the received Extensible Encoding Record exceeds MaxContentTypeSize.

[<16> Section 3.1.4.6](#Appendix_A_Target_16): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API sends a Fault Record (ContentTypeInvalid) if an unsupported content type is specified in the received Encoding Record.

[<17> Section 3.1.4.7](#Appendix_A_Target_17): The Windows implementation of this protocol exercised by both Windows Communication Framework and Windows Web Services API defines a MaxUpgradeProtocolSize of 256 bytes.

[<18> Section 3.1.4.7](#Appendix_A_Target_18): The Windows implementation of this protocol exercised by both Windows Communication Framework and Windows Web Services API does not send a Fault Record if the size of an upgrade protocol name in the received [Upgrade Request Record](#Section_6ef3ede82ead42ceabf16d144271746f) exceeds MaxUpgradeProtocolSize.

[<19> Section 3.1.4.7](#Appendix_A_Target_19): The Windows implementation of this protocol exercised by both Windows Communication Framework and Windows Web Services API sends a Fault Record (UpgradeInvalid) if an unsupported upgrade protocol name is specified in an Upgrade Request Record.

[<20> Section 3.1.4.11](#Appendix_A_Target_20): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services uses a MaxEnvelopeSize as configured externally.

[<21> Section 3.1.4.11](#Appendix_A_Target_21): The Windows implementation of this protocol that is exercised by both Windows Communication Framework and Windows Web Services API sends a Fault Record (MaxMessageSizeExceededFault) if the size of the received [Sized Envelope Record](#Section_82159b784bbe4ca8a707eeb65d3c6173) exceeds MaxEnvelopeSize but is not greater than 0xffffffff. No Fault Record is sent if the size of the received Sized Envelope Record exceeds 0xffffffff.

[<22> Section 3.1.4.12](#Appendix_A_Target_22): The Windows implementation of this protocol that is exercised by Windows Communication Framework defines a MaxChunkSize of 0xfffffffa.

[<23> Section 3.1.4.12](#Appendix_A_Target_23): The Windows implementation of this protocol that is exercised by Windows Communication Framework does not send a Fault Record if the size of a chunk in the received [Unsized Envelope Record](#Section_5e46160dc9384882bf160477449e61e3) exceeds MaxChunkSize.

# Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

* A document revision that incorporates changes to interoperability requirements.
* A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

| Section | Description | Revision class |
| --- | --- | --- |
| [6](#Section_a9856ce3300146f4aac39e7b5b9d736c) Appendix A: Product Behavior | Updated the applicability list for this release of Microsoft .NET Framework. | Major |

# Index

A

Abstract data model

 initiator ([section 3.1.1](#section_ce2c4bdc84c94a5fbb131fcadd68bb9c) 25, [section 3.2.1](#section_935400a0b6fa42ef86bc7b5ec9c57822) 37)

 receiver ([section 3.1.1](#section_ce2c4bdc84c94a5fbb131fcadd68bb9c) 25, [section 3.3.1](#section_f34a63aaf24644b8afc391c3a64fb7e2) 38)

[Applicability](#section_43a90a1342da4375a898ea1ced6811d7) 13

C

[Capability negotiation](#section_d61ad41f8cdb42188e65a9f475ae2924) 13

[Change tracking](#section_32621a625b6e47dcadc46a33065cedf2) 50

[Closed transport session - underlying](#section_4b11dcb02fb84c9880c560bddf64423d) 36

Communication modes

 [message chunking](#section_a3afffbf1c7d47b095652c4ff332dd52) 12

 [message property scope](#section_af983c6c2a7840e7ba0e27a306ea7b34) 11

 [message traffic flow](#section_17cb952eb1f843649036277e68054598) 12

 [overview](#section_0c44b067d8fb4852b8f09d5482179c67) 11

 [protocol receiver mode](#section_6c5e2e37eec94e33a8084dd09ae30ea0) 12

D

Data model - abstract

 initiator ([section 3.1.1](#section_ce2c4bdc84c94a5fbb131fcadd68bb9c) 25, [section 3.2.1](#section_935400a0b6fa42ef86bc7b5ec9c57822) 37)

 receiver ([section 3.1.1](#section_ce2c4bdc84c94a5fbb131fcadd68bb9c) 25, [section 3.3.1](#section_f34a63aaf24644b8afc391c3a64fb7e2) 38)

[Data\_Chunk packet](#section_e72f9eb7eeaf48ce9c0750f7c5cd856b) 21

[Duplex Mode example](#section_6c5667dfdc89426785aa189bab4317af) 41

E

[Encoding Record type](#section_c80201ae35af443abf23d49997cb481b) 35

[End Record type](#section_460346a9710544b287248cc64750d155) 36

[End\_Record packet](#section_9d691d25fd3a41ca9eb5241f47a914e5) 20

[Envelope Encoding Record](#section_8d29c0f1b83c4778b62555b03f373084) 17

[Envelope records](#section_2505d41817e44975a5cb0648cb9eda0e) 20

[Envelope Records message](#section_2505d41817e44975a5cb0648cb9eda0e) 20

Examples

 [Duplex Mode](#section_6c5667dfdc89426785aa189bab4317af) 41

 Initiator Receiver

 [End Message](#section_52062b15c24440c789e9b35c0ec4e411) 44

 [Preamble End Message](#section_5debbd33d9594b659292e4c5080b7c1a) 42

 [Preamble Message](#section_99b0c4e1fba14e4ab6213a73aa4d48ce) 42

 [Sized Envelope Message](#section_952336d9aa944472a6aa49252c329c7a) 43

 Receiver Initiator

 [End Message](#section_ec5c2c1cc3e9434eb9d5f5bfd97204e5) 44

 [Preamble Ack Message](#section_cb2486669bb84a56abaa1246a5a99603) 43

 [Sized Envelope Message](#section_7b9e916019504949bc3c22cf6790437b) 44

[Extensible\_Encoding\_Record packet](#section_ac28de9318424d6680741c2a34e24720) 18

F

[Fault Records message](#section_337e8351085448d2864b97520026c2c6) 22

[Fault\_Records packet](#section_337e8351085448d2864b97520026c2c6) 22

[Fields - vendor-extensible](#section_f478f2218bea4efe8d2134ff5d102095) 13

G

[Glossary](#section_442a483cf1b44b31b1e655e40e587fe9) 7

[Grammar](#section_f651a506e0904abaa2cbfceeb5ff21a5) 31

H

[Handling receipt of an unexpected record type](#section_4f257622d8364777ba43472ec928416f) 34

Higher-layer triggered events

 initiator

 [end record - send](#section_f9f7a056c4284d4db06dd6c835ca9696) 38

 message

 [receive](#section_58cca2499fe14512ba944d69946487fc) 38

 [send](#section_f482a2f220a647fe88d153fd6a5d6ff5) 37

 overview ([section 3.1.4](#section_0d990b63a29845a58ad42d71393ee3da) 33, [section 3.2.4](#section_43d7618da0e247d399512b6eee2002da) 37)

 [preamble - send](#section_9e89f03bf8d84eb6921d1c5084c7e53f) 37

 session

 [closed](#section_bf953746343148a899d1a850b49908e7) 38

 [initialized](#section_bf92bb9618494300ab18ecc04594bc9c) 37

 receiver

 [end record - send](#section_43de9e53ae554d6c8d0116e0dc773b6e) 40

 message

 [receive](#section_ec0100c6678c40d78a21fd89a1d94c45) 39

 [send](#section_3b563c0167244c1089eb2eaa8746ddd3) 39

 overview ([section 3.1.4](#section_0d990b63a29845a58ad42d71393ee3da) 33, [section 3.3.4](#section_4ef4f14e123e4dfab36150570ffda008) 39)

 [preamble - receive](#section_a1514480fb82428aaf21e80fcd86c7d9) 39

 session

 [closed](#section_9245115d15e8449ba4130f854d60e886) 40

 [initialized](#section_2b4af5e3cb1c4f79a0ea0898da65ca99) 39

I

[Implementer - security considerations](#section_c0339adcc9254d39856ce1f2874beea6) 45

[Index of security parameters](#section_e8af333b129d45329c581f1a65461e1d) 45

[Informative references](#section_64de0839a22b47fd9ceb65ae3c9a250a) 8

Initialization

 initiator ([section 3.1.3](#section_6046b1b2de1544c684182d1a1a11ad67) 33, [section 3.2.3](#section_00da154b701b4b9ba2c38ed82b96607d) 37)

 receiver ([section 3.1.3](#section_6046b1b2de1544c684182d1a1a11ad67) 33, [section 3.3.3](#section_ecda6223ea134d01ba5573ba097122d3) 38)

Initiator

 abstract data model ([section 3.1.1](#section_ce2c4bdc84c94a5fbb131fcadd68bb9c) 25, [section 3.2.1](#section_935400a0b6fa42ef86bc7b5ec9c57822) 37)

 higher-layer triggered events

 [end record - send](#section_f9f7a056c4284d4db06dd6c835ca9696) 38

 message

 [receive](#section_58cca2499fe14512ba944d69946487fc) 38

 [send](#section_f482a2f220a647fe88d153fd6a5d6ff5) 37

 overview ([section 3.1.4](#section_0d990b63a29845a58ad42d71393ee3da) 33, [section 3.2.4](#section_43d7618da0e247d399512b6eee2002da) 37)

 [preamble - send](#section_9e89f03bf8d84eb6921d1c5084c7e53f) 37

 session

 [closed](#section_bf953746343148a899d1a850b49908e7) 38

 [initialized](#section_bf92bb9618494300ab18ecc04594bc9c) 37

 initialization ([section 3.1.3](#section_6046b1b2de1544c684182d1a1a11ad67) 33, [section 3.2.3](#section_00da154b701b4b9ba2c38ed82b96607d) 37)

 local events ([section 3.1.7](#section_4d3909e2544241009e48f327b944c580) 36, [section 3.2.7](#section_bfa2546c5f7f4b529ee016efecc3b022) 38)

 message processing ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.2.5](#section_0eea09a0943640fd83bf4a1664418118) 38)

 sequencing rules ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.2.5](#section_0eea09a0943640fd83bf4a1664418118) 38)

 timer events ([section 3.1.6](#section_dbba7834b8f14363bdf67d9e11318ae7) 36, [section 3.2.6](#section_94977fc6b0b844cb862879878c44970e) 38)

 timers ([section 3.1.2](#section_9241118d867944799ae4d54eec31a420) 33, [section 3.2.2](#section_6d54604303d54f97bb57453bf8654aa0) 37)

Initiator Receiver

 [End Message example](#section_52062b15c24440c789e9b35c0ec4e411) 44

 [Preamble End Message example](#section_5debbd33d9594b659292e4c5080b7c1a) 42

 [Preamble Message example](#section_99b0c4e1fba14e4ab6213a73aa4d48ce) 42

 [Sized Envelope Message example](#section_952336d9aa944472a6aa49252c329c7a) 43

[Initiator-receiver interactions](#section_e95350122e4349379a549b2b41fa2ab9) 25

[Interactions - initiator-receiver](#section_e95350122e4349379a549b2b41fa2ab9) 25

[Introduction](#section_5d99c898c07e4d9daa413b52a5af6f12) 7

K

[Known\_Encoding\_Record packet](#section_a7d444632c60482d88569a8ff5929c62) 17

L

[Large message exchange scenario](#section_0a8c5aee01f14b5b9632d30d338e6380) 11

Local events

 initiator ([section 3.1.7](#section_4d3909e2544241009e48f327b944c580) 36, [section 3.2.7](#section_bfa2546c5f7f4b529ee016efecc3b022) 38)

 receiver ([section 3.1.7](#section_4d3909e2544241009e48f327b944c580) 36, [section 3.3.7](#section_ffb8f42b0bd94a00b392eb4b1544252c) 40)

M

Message exchange scenario

 [large](#section_0a8c5aee01f14b5b9632d30d338e6380) 11

 [multiple bidirectional](#section_a9a960b3b75c4abea5bca56fa4dd0132) 11

 [offline](#section_c265a609f6e24460b67e44632e2f5ecd) 11

Message processing

 initiator ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.2.5](#section_0eea09a0943640fd83bf4a1664418118) 38)

 receiver ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.3.5](#section_9e9abe1c938a490b956af386963faa21) 40)

Messages

 [chunking](#section_a3afffbf1c7d47b095652c4ff332dd52) 12

 [Envelope Records](#section_2505d41817e44975a5cb0648cb9eda0e) 20

 [Fault Records](#section_337e8351085448d2864b97520026c2c6) 22

 [Preamble Message](#section_f7b0315d48434376bf66265e5756cb5c) 23

 [Property Records](#section_0dcfc34423204a5fab7c69ab7c53f5cb) 15

 [property scope](#section_af983c6c2a7840e7ba0e27a306ea7b34) 11

 [Record Size Encoding](#section_070844defda143e389a0c9327f8f512e) 14

 [Record Types](#section_9f8bc7166b034c13b68347d740719b4e) 14

 [traffic flow](#section_17cb952eb1f843649036277e68054598) 12

 [transport](#section_4c69aedd9857491bbe3202951d3c81f1) 14

[Mode Record type](#section_ee2314e797d64753bed766d2768097ff) 34

[Mode\_Record packet](#section_8cd0b687a7fc45e5b328e00225836af3) 16

[Multiple bidirectional message exchange scenario](#section_a9a960b3b75c4abea5bca56fa4dd0132) 11

N

[Normative references](#section_eed9ba8c29694de29e0dac36252da073) 8

O

[Offline message exchange scenario](#section_c265a609f6e24460b67e44632e2f5ecd) 11

[Overview (synopsis)](#section_9e0fe2c177424322bbaff6aaf93d324b) 9

P

[Parameter index - security](#section_e8af333b129d45329c581f1a65461e1d) 45

[Parameters - security index](#section_e8af333b129d45329c581f1a65461e1d) 45

[Preamble Ack Record type](#section_a013f71801604f238654bb5880536ed7) 36

[Preamble End Record type](#section_3a30f5e662e84a27ab614cbfed066419) 35

[Preamble Message message](#section_f7b0315d48434376bf66265e5756cb5c) 23

[Preamble\_Ack\_Record packet](#section_3ec275c549e04becb7515cb572c3e7ab) 20

[Preamble\_End\_Record packet](#section_2ac9c4efa6544b109cf8b5b8917c654e) 20

[Preamble\_Message packet](#section_f7b0315d48434376bf66265e5756cb5c) 23

[Preconditions](#section_6a1575de73974a94a2f499ed9c83999c) 12

[Prerequisites](#section_6a1575de73974a94a2f499ed9c83999c) 12

[Product behavior](#section_a9856ce3300146f4aac39e7b5b9d736c) 46

[Property records](#section_0dcfc34423204a5fab7c69ab7c53f5cb) 15

[Property Records message](#section_0dcfc34423204a5fab7c69ab7c53f5cb) 15

Protocol Details

 [overview](#section_b4ce1c56f6b64936995fb97fba8b40d8) 25

[Protocol receiver mode](#section_6c5e2e37eec94e33a8084dd09ae30ea0) 12

[Protocol upgrades](#section_9b2101ded2f74881ae0ddd4deb0d5dc5) 12

R

[Reading variable-sized records](#section_24e76a050dbd4aa49594678f12a01f3a) 33

[Receipt of an unexpected record type - handling](#section_4f257622d8364777ba43472ec928416f) 34

Receiver

 abstract data model ([section 3.1.1](#section_ce2c4bdc84c94a5fbb131fcadd68bb9c) 25, [section 3.3.1](#section_f34a63aaf24644b8afc391c3a64fb7e2) 38)

 higher-layer triggered events

 [end record - send](#section_43de9e53ae554d6c8d0116e0dc773b6e) 40

 message

 [receive](#section_ec0100c6678c40d78a21fd89a1d94c45) 39

 [send](#section_3b563c0167244c1089eb2eaa8746ddd3) 39

 overview ([section 3.1.4](#section_0d990b63a29845a58ad42d71393ee3da) 33, [section 3.3.4](#section_4ef4f14e123e4dfab36150570ffda008) 39)

 [preamble - receive](#section_a1514480fb82428aaf21e80fcd86c7d9) 39

 session

 [closed](#section_9245115d15e8449ba4130f854d60e886) 40

 [initialized](#section_2b4af5e3cb1c4f79a0ea0898da65ca99) 39

 initialization ([section 3.1.3](#section_6046b1b2de1544c684182d1a1a11ad67) 33, [section 3.3.3](#section_ecda6223ea134d01ba5573ba097122d3) 38)

 local events ([section 3.1.7](#section_4d3909e2544241009e48f327b944c580) 36, [section 3.3.7](#section_ffb8f42b0bd94a00b392eb4b1544252c) 40)

 message processing ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.3.5](#section_9e9abe1c938a490b956af386963faa21) 40)

 sequencing rules ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.3.5](#section_9e9abe1c938a490b956af386963faa21) 40)

 timer events ([section 3.1.6](#section_dbba7834b8f14363bdf67d9e11318ae7) 36, [section 3.3.6](#section_ac30031c2f614755a47deac3da8bc705) 40)

 timers ([section 3.1.2](#section_9241118d867944799ae4d54eec31a420) 33, [section 3.3.2](#section_c4c069d0a00b43acb917386403c71de5) 38)

Receiver Initiator

 [End Message example](#section_ec5c2c1cc3e9434eb9d5f5bfd97204e5) 44

 [Preamble Ack Message example](#section_cb2486669bb84a56abaa1246a5a99603) 43

 [Sized Envelope Message example](#section_7b9e916019504949bc3c22cf6790437b) 44

[Record Size Encoding message](#section_070844defda143e389a0c9327f8f512e) 14

[Record Types message](#section_9f8bc7166b034c13b68347d740719b4e) 14

Records

 [Encoding Record type](#section_c80201ae35af443abf23d49997cb481b) 35

 [End Record type](#section_460346a9710544b287248cc64750d155) 36

 [envelope](#section_2505d41817e44975a5cb0648cb9eda0e) 20

 [handling receipt of an unexpected type](#section_4f257622d8364777ba43472ec928416f) 34

 [Mode Record type](#section_ee2314e797d64753bed766d2768097ff) 34

 [Preamble Ack Record type](#section_a013f71801604f238654bb5880536ed7) 36

 [Preamble End Record type](#section_3a30f5e662e84a27ab614cbfed066419) 35

 [property](#section_0dcfc34423204a5fab7c69ab7c53f5cb) 15

 [reading variable-sized](#section_24e76a050dbd4aa49594678f12a01f3a) 33

 [size encoding](#section_070844defda143e389a0c9327f8f512e) 14

 [Sized Envelope Record type](#section_f1805a21c1614364b82808121a5ee18d) 36

 [types](#section_9f8bc7166b034c13b68347d740719b4e) 14

 [Unsized Envelope Record type](#section_03664f95c9d44101ba5de9fb5b329e6b) 36

 [Upgrade Request Record type](#section_bbec3f6b22474a2591abf44d0437e0de) 35

 [Upgrade Response Record type](#section_7b8d1f6983904455aad22abf24831eb8) 35

 [Version Record type](#section_41938831569840c0829173a2ee107a39) 34

 [Via Record type](#section_8ca81a2267c346ce8c77eab9fe9100a4) 34

[References](#section_cc0ea2bc473c457c806cda413babe6f3) 7

 [informative](#section_64de0839a22b47fd9ceb65ae3c9a250a) 8

 [normative](#section_eed9ba8c29694de29e0dac36252da073) 8

[Relationship to other protocols](#section_c113d95ba631471ebc591aae95cf7cce) 12

S

Scenarios

 [large message exchange](#section_0a8c5aee01f14b5b9632d30d338e6380) 11

 message exchange

 [large](#section_0a8c5aee01f14b5b9632d30d338e6380) 11

 [multiple bidirectional](#section_a9a960b3b75c4abea5bca56fa4dd0132) 11

 [offline](#section_c265a609f6e24460b67e44632e2f5ecd) 11

 [multiple bidirectional message exchange](#section_a9a960b3b75c4abea5bca56fa4dd0132) 11

 [offline message exchange](#section_c265a609f6e24460b67e44632e2f5ecd) 11

 [overview](#section_8b9ecf23e75e4e1f82138595bb0aecbc) 10

Security

 [implementer considerations](#section_c0339adcc9254d39856ce1f2874beea6) 45

 [parameter index](#section_e8af333b129d45329c581f1a65461e1d) 45

Sequencing rules

 initiator ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.2.5](#section_0eea09a0943640fd83bf4a1664418118) 38)

 receiver ([section 3.1.5](#section_f7653f834f8846e2812327c8949ff24e) 36, [section 3.3.5](#section_9e9abe1c938a490b956af386963faa21) 40)

[Sized Envelope Record type](#section_f1805a21c1614364b82808121a5ee18d) 36

[Sized\_Envelope\_Record packet](#section_82159b784bbe4ca8a707eeb65d3c6173) 21

[Standards assignments](#section_51b5eb53f4884b74b21d8a498f016b61) 13

T

Timer events

 initiator ([section 3.1.6](#section_dbba7834b8f14363bdf67d9e11318ae7) 36, [section 3.2.6](#section_94977fc6b0b844cb862879878c44970e) 38)

 receiver ([section 3.1.6](#section_dbba7834b8f14363bdf67d9e11318ae7) 36, [section 3.3.6](#section_ac30031c2f614755a47deac3da8bc705) 40)

Timers

 initiator ([section 3.1.2](#section_9241118d867944799ae4d54eec31a420) 33, [section 3.2.2](#section_6d54604303d54f97bb57453bf8654aa0) 37)

 receiver ([section 3.1.2](#section_9241118d867944799ae4d54eec31a420) 33, [section 3.3.2](#section_c4c069d0a00b43acb917386403c71de5) 38)

[Tracking changes](#section_32621a625b6e47dcadc46a33065cedf2) 50

[Transport](#section_4c69aedd9857491bbe3202951d3c81f1) 14

[Transport session - underlying - closed](#section_4b11dcb02fb84c9880c560bddf64423d) 36

Triggered events - higher-layer

 initiator

 [end record - send](#section_f9f7a056c4284d4db06dd6c835ca9696) 38

 message

 [receive](#section_58cca2499fe14512ba944d69946487fc) 38

 [send](#section_f482a2f220a647fe88d153fd6a5d6ff5) 37

 overview ([section 3.1.4](#section_0d990b63a29845a58ad42d71393ee3da) 33, [section 3.2.4](#section_43d7618da0e247d399512b6eee2002da) 37)

 [preamble - send](#section_9e89f03bf8d84eb6921d1c5084c7e53f) 37

 session

 [closed](#section_bf953746343148a899d1a850b49908e7) 38

 [initialized](#section_bf92bb9618494300ab18ecc04594bc9c) 37

 receiver

 [end record - send](#section_43de9e53ae554d6c8d0116e0dc773b6e) 40

 message

 [receive](#section_ec0100c6678c40d78a21fd89a1d94c45) 39

 [send](#section_3b563c0167244c1089eb2eaa8746ddd3) 39

 overview ([section 3.1.4](#section_0d990b63a29845a58ad42d71393ee3da) 33, [section 3.3.4](#section_4ef4f14e123e4dfab36150570ffda008) 39)

 [preamble - receive](#section_a1514480fb82428aaf21e80fcd86c7d9) 39

 session

 [closed](#section_9245115d15e8449ba4130f854d60e886) 40

 [initialized](#section_2b4af5e3cb1c4f79a0ea0898da65ca99) 39

U

[Underlying transport session is closed](#section_4b11dcb02fb84c9880c560bddf64423d) 36

[Unexpected record type - handling receipt](#section_4f257622d8364777ba43472ec928416f) 34

[Unsized Envelope Record type](#section_03664f95c9d44101ba5de9fb5b329e6b) 36

[Unsized\_Envelope\_Record packet](#section_5e46160dc9384882bf160477449e61e3) 21

[Upgrade Request Record type](#section_bbec3f6b22474a2591abf44d0437e0de) 35

[Upgrade Response Record type](#section_7b8d1f6983904455aad22abf24831eb8) 35

[Upgrade\_Request\_Record packet](#section_6ef3ede82ead42ceabf16d144271746f) 19

[Upgrade\_Response\_Record packet](#section_8f9da3a35345482cbcb7543972c1ee4a) 20

[Upgrades](#section_9b2101ded2f74881ae0ddd4deb0d5dc5) 12

V

[Variable-sized records - reading](#section_24e76a050dbd4aa49594678f12a01f3a) 33

[Vendor-extensible fields](#section_f478f2218bea4efe8d2134ff5d102095) 13

[Version Record type](#section_41938831569840c0829173a2ee107a39) 34

[Version\_Record packet](#section_b8dfbcd6b65b495aa1f1e19b78897a3d) 16

[Versioning](#section_d61ad41f8cdb42188e65a9f475ae2924) 13

[Via Record type](#section_8ca81a2267c346ce8c77eab9fe9100a4) 34

[Via\_Record packet](#section_2d61448ab4d14cbd911cd5caff54b64f) 17