BS ISO 15000-1:2021

BSI Standards Publication

Electronic business eXtensible Markup Language (ebXML)

Part 1: Messaging service core specification

Documents
Document Sharing Hub

Full document details: BS ISO 15000-1:2021

National foreword

This British Standard is the UK implementation of ISO 15000-1:2021.

The UK participation in its preparation was entrusted to Technical Committee IST/47/-/3, eBusiness.

A list of organizations represented on this committee can be obtained on request to its committee manager.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2021
Published by BSI Standards Limited 2021

ISBN 978 0 539 13087 4

ICS 35.040.50

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 28 February 2021.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO 15000-1:2021

INTERNATIONAL STANDARD

ISO 15000-1

First edition
2021-02

Electronic business eXtensible Markup Language (ebXML) —

Part 1: Messaging service core specification

Reference number
ISO 15000-1:2021(E)

Documents
Document Sharing Hub
© ISO 2021

Full document details: BS ISO 15000-1:2021

COPYRIGHT PROTECTED DOCUMENT

© ISO 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO 15000-1:2021

Contents

Foreword ... vii
Introduction ... ix
1. Scope ... 1
2. Normative references ... 1
3. Terms and definitions ... 3
4. Relevant messaging concepts ... 6
4.1. Web services and their role in an eBusiness messaging framework ... 6
4.2. Caveats and assumptions ... 7
4.3. XML notation ... 7
4.4. Namespace prefixes ... 7
4.5. Example domains ... 8
5. Messaging model ... 8
5.1. Model components ... 8
5.1.1. Components of the model ... 8
5.1.2. Messaging roles ... 9
5.1.3. Abstract messaging operations ... 10
5.2. Message exchange patterns ... 10
5.2.1. Rationale ... 10
5.2.2. General definition ... 11
5.2.3. MEP bindings ... 11
5.2.4. Relationship to SOAP MEPs ... 13
5.2.5. The One-Way/Push MEP ... 13
5.2.6. The One-Way/Pull MEP ... 14
5.2.7. The Two-Way/Sync MEP ... 15
5.2.8. Other transport-channel-bound MEPs ... 16
6. Message pulling and partitioning ... 17
6.1. Objectives ... 17
6.2. Supporting message pulling ... 17
6.3. Combining pulling with security and reliability ... 19
6.4. Message partition channels ... 20
6.4.1. Concept and purpose ... 20
6.4.2. Some use cases ... 22
6.4.3. Definition and usage requirements ... 23
7. Processing modes ... 24
7.1. General ... 24
7.2. Messaging service processing model ... 25
7.3. Processing mode features ... 26
7.4. Default features for processing mode ... 27

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

8. Message packaging ... 28
8.1. Message envelope and message parts ... 28
8.1.1. MIME structure and SOAP profile ... 28
8.1.2. MIME and XML considerations ... 31
8.1.3. ebXML SOAP envelope extension ... 32
8.1.4. ebMS header ... 33
8.1.5. Payload containers ... 34
8.2. The eb:Messaging container element ... 35
8.2.1. General ... 35
8.2.2. eb:Messaging element specification ... 36
8.2.3. eb:Messaging/eb:UserMessage ... 37
8.2.4. eb:Messaging/eb:SignalMessage ... 43
8.2.5. Message unit bundling ... 44
8.3. Examples of ebMS messages (informative) ... 45
8.3.2. UserMessage example ... 45
8.3.3. PullRequest message example ... 47
8.3.4. Error message example ... 47
8.3.5. Receipt message example ... 48
8.3.6. "Bundled" message example ... 48
9. Error handling ... 50
9.1. General ... 50
9.2. Packaging of ebMS errors ... 50
9.2.1. eb:Error element ... 50
9.2.2. eb:Error/@origin ... 50
9.2.3. eb:Error/@category ... 51
9.2.4. eb:Error/@errorCode ... 51
9.2.5. eb:Error/@severity ... 51
9.2.6. eb:Error/@refToMessageInError ... 51
9.2.7. eb:Error/@shortDescription ... 51
9.2.8. eb:Error/Description ... 51
9.2.9. eb:Error/ErrorDetail ... 51
9.3. ebMS Error message ... 51
9.4. Extensibility of the eb:Error element ... 52
9.4.1. Adding new ebMS errors ... 52
9.5. Generating ebMS errors ... 52
9.6. Error reporting ... 52
9.7. Standard ebMS errors ... 53
9.7.1. General ... 53
9.7.2. ebMS processing errors ... 53
9.7.3. Security processing errors ... 54
9.7.4. Reliable messaging errors ... 55

Documents
Document Sharing Hub

© ISO 2021 – All rights reserved

Full document details: BS ISO 15000-1:2021

10. Security module 55
10.1. General 55
10.2. Security element 55
10.3. Signing messages 56
10.4. Signing SOAP with attachments messages 56
10.5. Encrypting messages 57
10.6. Encrypting SOAP with attachments messages 57
10.7. Signing and encrypting messages 57
10.8. Security token authentication 57
10.9. Security policy errors 57
10.10. Secured message examples 58
10.10.1. Digitally signed and encrypted ebXML message 58
10.10.2. Digitally signed and encrypted ebXML SOAP with attachments message 60
10.10.3. Digitally signed receipt signal message 62
10.11. Message authorization 63
10.12. Securing the PullRequest signal 65
10.12.1. Authentication 65
10.12.2. Authorization 65
10.12.3. Preventing replay attacks 65
10.13. Countermeasure technologies 65
10.13.1. Persistent digital signature 65
10.13.2. Persistent signed receipt 66
10.13.3. Non-persistent authentication 66
10.13.4. Non-persistent integrity 66
10.13.5. Persistent confidentiality 66
10.13.6. Non-persistent confidentiality 66
10.13.7. Persistent authorization 66
10.13.8. Non-persistent authorization 66
10.14. Security considerations 66
11. Reliable messaging module 68
11.1. The reliable messaging model 68
11.1.1. General 68
11.1.2. Message processing 68
11.1.3. The reliable messaging processor in the MSH 68
11.2. Reliable delivery of ebMS messages 71
11.2.1. General 71
11.2.2. Reliability contracts for the RMP 71
11.2.3. Reliability contracts for the MSH 72
11.2.4. Reliability for signal messages 73
11.2.5. Handling of delivery failures 73

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

11.3. Reliability of ebMS MEPs ... 74
11.3.1. General ... 74
11.3.2. Reliability of the One-Way/Push MEP ... 75
11.3.3. Reliability of the One-Way/Pull MEP ... 76
11.3.4. Reliability of the Two-Way/Sync MEP ... 77
11.3.5. Reliability of other transport-channel-bound MEPs ... 78
Annex A (informative) The ebXML SOAP extension element schema ... 79
Annex B (informative) Reliable messaging bindings ... 83
Annex C (informative) SOAP format and bindings ... 90
Annex D (informative) Processing modes ... 94
Annex E (informative) P-Mode values and ebMS MEP bindings ... 103
Annex F (informative) Compatibility mapping to ebMS 2.0 ... 106
Annex G (informative) Conformance ... 114
Bibliography ... 115

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by the OASIS ebXML (Electronic business eXtensible Markup Language) Messaging Services Committee (as “OASIS ebXML Messaging Services Version 3.0”) and drafted in accordance with its editorial rules. It was assigned to Technical Committee ISO/TC 154, *Processes, data elements and documents in commerce, industry and administration* and adopted under the "fast-track procedure".

This first edition incorporates content from ISO/TS 15000-2:2004 and ISO/TS 15000-1:2004, which have been withdrawn.

The main changes compared to ISO/TS 15000-2:2004 and ISO/TS 15000-1:2004 are as follows:

- The original ISO/TS 15000-2:2004 specification for the ebXML Messaging Service (ebMS) has been updated and refactored into multiple parts, including this document, the “Core” specification for ebMS, resubmitted to become this document.
- A separate “AS4 Profile of ebMS 3.0 Version 1.0” is submitted separately to ISO/TC 154. It provides a select limited profile of the ebMS3 specification sufficient for Web Services business-to-business messaging applications over the HTTP transport protocol, and is to become ISO 15000-2.
- This document defines the basic (and some optional) features necessary for reliable electronic messaging and the transactional interactions that support such messaging.
- This document noted the availability of several newer methodologies, represented by normative references and informative references included here, that were not available as of the 2004 version.

A list of all parts in the ISO 15000 series can be found on the ISO website.

Documents
Document Sharing Hub

© ISO 2021 – All rights reserved

Full document details: BS ISO 15000-1:2021

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

Introduction

This document describes a communication-protocol neutral method for exchanging electronic business messages. It defines specific enveloping constructs supporting reliable, secure delivery of business information. Furthermore, the specification defines a flexible enveloping technique, permitting messages to contain payloads of any format type. This versatility ensures that legacy electronic business systems employing traditional syntaxes (i.e. UN/EDIFACT, ASC X12, or HL7) can leverage the advantages of the ebXML infrastructure along with users of emerging technologies.

The prime objective of the ebXML messaging service (ebMS) is to facilitate the exchange of electronic business messages within an XML framework that leverages common Internet standards, without making any assumption on the integration and consumption model these messages will follow on the back-end. These messages may be consumed in different ways that are out of scope of this document: they may bind to a legacy application, to a service, be queued, enter a message workflow process, be expected by an already-running business process, be batched for delayed processing, be routed over an Enterprise Service Bus (ESB) before reaching their consumer application, or be dispatched based on header data or payload data, etc.

It is becoming critical for broad adoption among all partners – large or small - of a supply-chain, to handle differences in message flow capacity, intermittent connectivity, lack of static IP addresses or firewall restrictions. Such new capabilities played an important role in the motivation that led to ebMS 3.0, along with the need to integrate and profile the emerging SOAP-based QoS-supporting standards. The message header profiling that provided, in ebMS 2.0, a standard business-level header, has also been extended to better address the diversity of back-end binding models, as well as the emerging trend in business activity monitoring, the eBusiness side of which a message handler should be able to support.

The ebXML messaging framework is not a restrictive one: business messages, identified as the 'payloads' of ebXML messages, are not limited to XML documents. Traditional EDI formats may also be transported by ebMS. These payloads can take any digital form–XML, ASC X12, HL7, AIAG E5, database tables, binary image files, etc. Multiple payloads, possibly of different MIME types, can be transported in a single ebMS message. An objective of ebXML Messaging protocol is to be capable of being carried over any available transfer protocol. This version of the specification provides bindings to HTTP and SMTP, but other protocols to which SOAP may bind can also be used. The choice of an XML framework rather reflects confidence in a growing XML-based Web infrastructure and development tools infrastructure, the components of which can be leveraged and reused by developers.

Documents
Document Sharing Hub

© ISO 2021 – All rights reserved

Full document details: BS ISO 15000-1:2021

Documents
Document Sharing Hub

Electronic business eXtensible Markup Language (ebXML) —

Part 1:
Messaging service core specification

1. Scope

This document provides a communication-protocol neutral method for exchanging electronic business messages. It defines specific enveloping constructs supporting reliable, secure delivery of business information. Furthermore, this document defines a flexible enveloping technique, permitting messages to contain payloads of any format type.

It specifies each of the following:

- Messaging model
- Message pulling and partitioning
- Processing modes
- Message packaging
- Error handling
- Security module
- Reliable messaging module

This document is applicable to all types of organizations (e.g., commercial enterprises, government agencies, not-for-profit organizations) that exchange documents or data electronically using messaging.

2. Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

INTERNET ASSIGNED NAMES AUTHORITY (IANA). *MIME Media Types*, Available from http://www.iana.org/assignments/media-types/.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2045. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, 1996. Edited by N Freed, et al. Available from http://www.ietf.org/rfc/rfc2045.txt.

Documents
Document Sharing Hub

© ISO 2021 – All rights reserved

Full document details: BS ISO 15000-1:2021

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2279. *UTF-8, a transformation format of ISO 10646*, 1998. Edited by F. Yergeau. Available from http://www.ietf.org/rfc/rfc2279.txt.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2387. *The MIME Multipart/Related Content-type*, 1998. Edited by E. Levinson, Available from http://www.ietf.org/rfc/rfc2387.txt.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2392. *Content-ID and Message-ID Uniform Resource Locators*, 1998. Edited by E. Levinson. Available from http://www.ietf.org/rfc/rfc2392.txt.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2396. *Content-ID and Message-ID Uniform Resource Locators*, 1998. Edited by T. Berners-Lee, et al. Available from http://www.ietf.org/rfc/rfc2396.txt.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2616. *Hypertext Transfer Protocol -- HTTP/1.1*, 1999. Edited by R. Fielding, et al. Available from http://www.ietf.org/rfc/rfc2616.txt.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2822. *Internet Message Format*, 2001. Edited by P. Resnick. Available from http://www.ietf.org/rfc/rfc2822.txt.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2821. *Simple Mail Transfer Protocol*, 2001. Edited by J. Klensin. Available from http://www.ietf.org/rfc/rfc2821.txt.

OASIS. *WS-Reliability 1.1*, 2004. Edited by Kazunori Iwasa, et al. Available from http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf.

OASIS. *Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1*, 2007. Edited by D. Davis, et al. Available from http://docs.oasis-open.org/ws-rx/wsrm/v1.1/wsrm.pdf.

OASIS. *Web Services Reliable Messaging Policy (WS-RM Policy) Version 1.1*, 2007. Edited by D. Davis, et al. Available from http://docs.oasis-open.org/ws-rx/wsrmp/v1.1/wsrmp.pdf.

OASIS.*Web Services Security: SOAP Message Security 1.0*, 2004. Edited by Anthony Nadalin, et al Available from http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

OASIS.*Web Services Security UsernameToken Profile 1.0*, 2004. Edited by P. Hallam-Baker, et al. Available from http://docs.oasis-open.org/wss/2004/01/.

OASIS.*Web Services Security X.509 Certificate Token Profile*, 2004. Edited by P. Hallam-Baker, et al Available from http://docs.oasis-open.org/wss/2004/01/.

ASIS.*Web Services Security: SOAP Message Security 1.1*, 2005. Edited by Anthony Nadalin, et al Available from http://docs.oasis-open.org/wss/v1.1/.

OASIS.*Web Services Security UsernameToken Profile 1.1*, 2006. Edited by A. Nadalin, et al. Available from http://docs.oasis-open.org/wss/v1.1/.

OASIS.*Web Services Security X.509 Certificate Token Profile 1.1*, 2006. Edited by A. Nadalin, et al. Available from http://docs.oasis-open.org/wss/v1.1/.

WEB SERVICES INTEROPERABILITY ORGANIZATION. *Attachments Profile Version 1.0*, 2004. Edited by Chris Ferris, et al. Available from http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html.

WEB SERVICES INTEROPERABILITY ORGANIZATION. *Basic Security Profile Version 1.0*, 2005. Edited by Abbie Barbir, et al. Available from http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html.

Documents
Document Sharing Hub

© ISO 2021 – All rights reserved

Full document details: BS ISO 15000-1:2021

WORLD WIDE WEB CONSORTIUM. *Simple Object Access Protocol (SOAP) 1.1*, 2000. Edited by D. Box, et al. Available from http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

WORLD WIDE WEB CONSORTIUM. SOAP Version 1.2 Part 1: Messaging Framework, 2003. Edited by M. Gudgin, et al Available from http://www.w3.org/TR/soap12-part1/.

WORLD WIDE WEB CONSORTIUM. *SOAP Messages with Attachments*, 2000. Edited by J. Barton, et al. Available from http://www.w3.org/TR/SOAP-attachments.

WORLD WIDE WEB CONSORTIUM.*Extensible Markup Language (XML) 1.0 (Third Edition)*, 2004. Edited by Tim Bray, et al. Available from http://www.w3.org/TR/2004/REC-xml-20040204/.

WORLD WIDE WEB CONSORTIUM. *XML Schema Part 1: Structures Second Edition*, 2004. Edited by Henry S. Thompson, et al. Available from http://www.w3.org/TR/xmlschema-1/.

WORLD WIDE WEB CONSORTIUM. *XML-Signature Syntax and Processing*, 2002. Edited by Donald Eastlake, et al. Available from http://www.w3.org/TR/xmldsig-core/.

WORLD WIDE WEB CONSORTIUM. *XML Encryption Syntax and Processing*, 2002. Edited by D. Eastlake, et al. Available from http://www.w3.org/TR/xmlenc-core/.

WORLD WIDE WEB CONSORTIUM. *Namespaces in XML*, 1999. Edited by Tim Bray, et al. Available from http://www.w3.org/TR/REC-xml-names/.

3. Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
ebMS error generation
Operation of creating an *ebMS error* (3.14) based on some failure or warning condition

3.2
ebMS error reporting
Operation of communicating an *ebMS error* (3.14) to some other entity

3.3
ebMS message
SOAP message (3.29) that contains SOAP header(s) qualified with the ebMS namespace and that conforms to this document

3.4
ebMS message exchange pattern
ebMS MEP
Choreography of *ebMS user messages* (3.11) which are all related through the use of the referencing feature (`eb:RefToMessageId`)

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

3.5
ebMS MEP transport channel binding
Binding of an *ebMS message exchange pattern* (3.4) that defines how the ebMS MEP maps abstractly to the channels allowed by an underlying transport protocol

3.6
ebMS MEP transport protocol binding
Binding of an *ebMS message exchange pattern* (3.4) that defines further how an *ebMS MEP transport channel binding* (3.5) is implemented over a specific underlying transport protocol

3.7
ebMS message unit
Logical unit of data that is a subset of an *ebMS Message* (3.3)

3.8
ebMS messaging service handler
ebMS MSH
Entity that is able to generate or process *ebMS messages* (3.3) that conform to this document

3.9
ebMS user message unit
ebMS message unit (3.7) that is represented by the XML infoset `eb:Messaging/eb:UserMessage`, together with any referenced payload items

Note 1 to entry: This is the part of the ebMS message that is submitted by a producer (via the "submit" operation) and that is subject to delivery to a consumer.

3.10
ebMS signal message unit
ebMS message unit (3.7) that is represented by the XML infoset `eb:Messaging/eb:SignalMessage`

Note 1 to entry: Its role is to activate a specific function in the receiving MSH. It is not intended to be delivered to a message consumer.

3.11
ebMS user message
ebMS message (3.3) that contains an *ebMS user message unit* (3.9)

3.12
ebMS signal message
ebMS message (3.3) that contains an *ebMS signal message unit* (3.10)

3.13
ebMS pull signal message
ebMS signal message (3.12) that contains an `eb:PullRequest` element

3.14
ebMS error
Particular case of *Error* (3.15) that is generated by the ebMS module in conformity with this document

3.15
Error
Object representing some failure or warning condition that originates in one of the defined modules (ebMS module, reliability module, security module)

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

3.16
Escalated ebMS error
ebMS error (3.14) that originates in a module other than the ebMS Module (i.e. security module, or reliability module)

3.17
Fault
Object representing some failure or warning condition that originates in SOAP processing

Note 1 to entry. It shall be generated and processed according to the W3C SOAP 1.1 or SOAP 1.2 specifications.

3.18
Message consumer
Entity that interacts with a receiving *ebMS MSH* (3.8) (i.e. an MSH in the receiving role) to consume data from a received *ebMS user message* (3.11)

3.19
Message partition channel
Mechanism to partition the flow of messages from a sending MSH to a receiving MSH into several flows that can be controlled separately and consumed differently

3.20
Message producer
Entity that interacts with a sending *ebMS MSH* (3.8) (i.e. an MSH in the sending role) to initiate the sending of an *ebMS user message* (3.11)

3.21
Message-in-error
Flawed message causing an *Error* (3.15) of some kind

3.22
One-Way message exchange pattern
one-way MEP
Exchange of a single *ebMS user message unit* (3.9) unrelated to other user messages

3.23
Processing mode
P-Mode
Contextual information that governs the processing of a particular *ebMS message* (3.3)

Note 1 to entry: A P-Mode is not provided on a per-message basis, but is common to a set of messages exchanged between or among parties. It may be interpreted as configuration data for a deployed MSH.

3.24
Processing mode operation set
Set of all *P-Modes* (3.23) that are supported by an *ebMS MSH* (3.8) during operation

3.25
Pull channel binding
ebMS MEP transport channel binding (3.5) in which the transfer of an *ebMS message* (3.3) is initiated by the receiver

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

3.26
Push channel binding
ebMS MEP transport channel binding (3.5) in which the transfer of an *ebMS message* (3.3) is initiated by the sender

3.27
Reliability error
Error (3.15) generated by the reliability module

3.28
Security error
Error (3.15) generated by the security module

3.29
SOAP message
Message that conforms to the SOAP 1.1 or SOAP 1.2 W3C specifications

3.30
Two-way message exchange pattern
two-way MEP
Exchange of two *ebMS user message units* (3.9) in opposite directions, in which the first one to occur is labeled "request" and the second one "reply".

4. Relevant messaging concepts

4.1. Web services and their role in an eBusiness messaging framework

A major design choice in ebMS 3 is the specification of the MSH and its associated processing rules using Web Services (WS) standards. The intent is to make use of other relevant Web Services specifications that fulfill certain messaging requirements, and build upon that base by adding what is necessary for a complete and coherent eBusiness messaging service. ebMS 3 brings this all together into a single, coherent framework.

In order to achieve this, message security and reliability requirements are met through the use of other Web Services standards and their implementations. The message SOAP body has been freed for business payload. The ebMS header is just a SOAP extension among others. As a result, ebMS 3 is significantly more compliant than ebMS 2 with the SOAP processing model, and apt at composing Web services standards that are defined as SOAP extensions. Compliance of ebMS 3 implementations with the latest version of WS-I profiles - once approved as final material by the organization - will be addressed in the definition of conformance profiles that are adjunct to this document (see Annex G). A compatibility mapping from ebMS 2 to ebMS 3 is provided in Annex F.

Compliance with Web services standards does not remove the rationale behind an Internet-based messaging middleware. Often, document-centric eBusiness and eGovernment exchanges need to clearly dissociate messaging functions from the way these messages are consumed on the back-end. Such consumption may take place according to various models. The use of SOAP message header elements that represent standard business metadata (user or company ID, business conversation, business service and action, etc.), is a key feature for supporting a decoupled binding with back-end business processes. At the same time, experience has demonstrated that the messaging layer needs to be more supportive of business transactions: messages are parts of basic choreographies that map to higher-level business exchanges between partners. To this end, ebMS 3 supports a notion of message exchange pattern (MEP) the properties of which (reliability, security, binding to underlying transport, error handling, and other quality of service aspects such as timing, etc.) are controlled in a contract-based manner by the message producer and consumer layers.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

4.2. Caveats and assumptions

The target audience for this document is the community of software developers who will implement the ebXML messaging service.

It is assumed the user has an understanding of communications protocols, MIME, XML, SOAP, SOAP Messages with Attachments and security technologies.

All examples are informative. If inconsistencies exist between the specification and the examples, the specification supersedes the examples.

Implementers are strongly advised to read and understand the OASIS Collaboration Protocol Profile & Agreement specification (CPPA) and its implications prior to implementation.

This document presents some alternatives regarding underlying specifications (e.g. SOAP 1.1/1.2, WSS1.0/1.1, and Web Services specifications that support the reliability function). This does not imply that a conforming implementation supports all of them, nor that it is free to support any option. The definition of conformance profiles - out of scope for this document, and to be described in an adjunct OASIS document - will complement this document by asserting which option(s) shall be supported in order to claim support for a particular conformance profile. Conformance to compatible profiles is a prerequisite to interoperability. See Annex G for more details on conformance profiles.

4.3. XML notation

When describing concrete XML schemas and information items, this document uses a convention in which each XML element or attribute is identified using abbreviated XPath 1.0 notation (e.g., `/x:MyHeader/x:SomeProperty/@attribute`).

4.4. Namespace prefixes

Table 1 maps various prefixes that appear in XML examples to their intended corresponding namespaces.

Table 1: Namespace Prefixes

Prefix	*Namespace*
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope
Ds	http://www.w3.org/2000/09/xmldsig#
eb, eb3	http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/
eb2	http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
enc	http://www.w3.org/2001/04/xmlenc#
wsr	http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

Prefix	*Namespace*
wsrx	http://docs.oasis-open.org/ws-rx/wsrm/200702
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
ebbpsig	http://docs.oasis-open.org/ebxml-bp/ebbp-signals-2.0

4.5. Example domains

Hostnames used in the examples are fictitious and conform to RFC 2606. The *example.org* domain is intended to refer generically to a relevant industry standards organization, while the *example.com* domain represents a participant in a message exchange (whether commercial, government, or other entity).

5. Messaging model

5.1. Model components

5.1.1. Components of the model

The ebMS messaging model assumes the following components:

- **ebMS MSH (Messaging Service Handler)**: An entity that is able to generate or process messages that conform to this document, and to act in at least one of two ebMS roles defined in 5.1.2: sending and receiving. In terms of SOAP processing, an MSH is either a SOAP processor or a chain of SOAP processors. In either case, an MSH shall be able to understand the `eb:Messaging` header (qualified with the ebMS namespace).

- **Producer (or Message producer)**: An entity that interacts with a sending MSH (i.e. an MSH in the sending role) to initiate the sending of a user message. Some examples are: an application, a queuing system, another SOAP processor (though not another MSH).

- **Consumer (or Message consumer)**: An entity that interacts with a receiving MSH (i.e. an MSH in the Receiving role) to consume data from a received user message. Some examples are: an application, a queuing system, another SOAP processor.

Figure 1 shows the entities and operations involved in a message exchange.

NOTE 1: In all figures, the arrows do not represent control flow, i.e. they do not represent a component invoking an operation on another component. They only represent data transfer under the control of an operation which may be implemented in either component.

NOTE 2: Producer and consumer are always MSH endpoints, and *Submit* and *Deliver* operations occur at the endpoints only once per message lifetime. Any actions performed by an intermediary will be defined in different terms.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

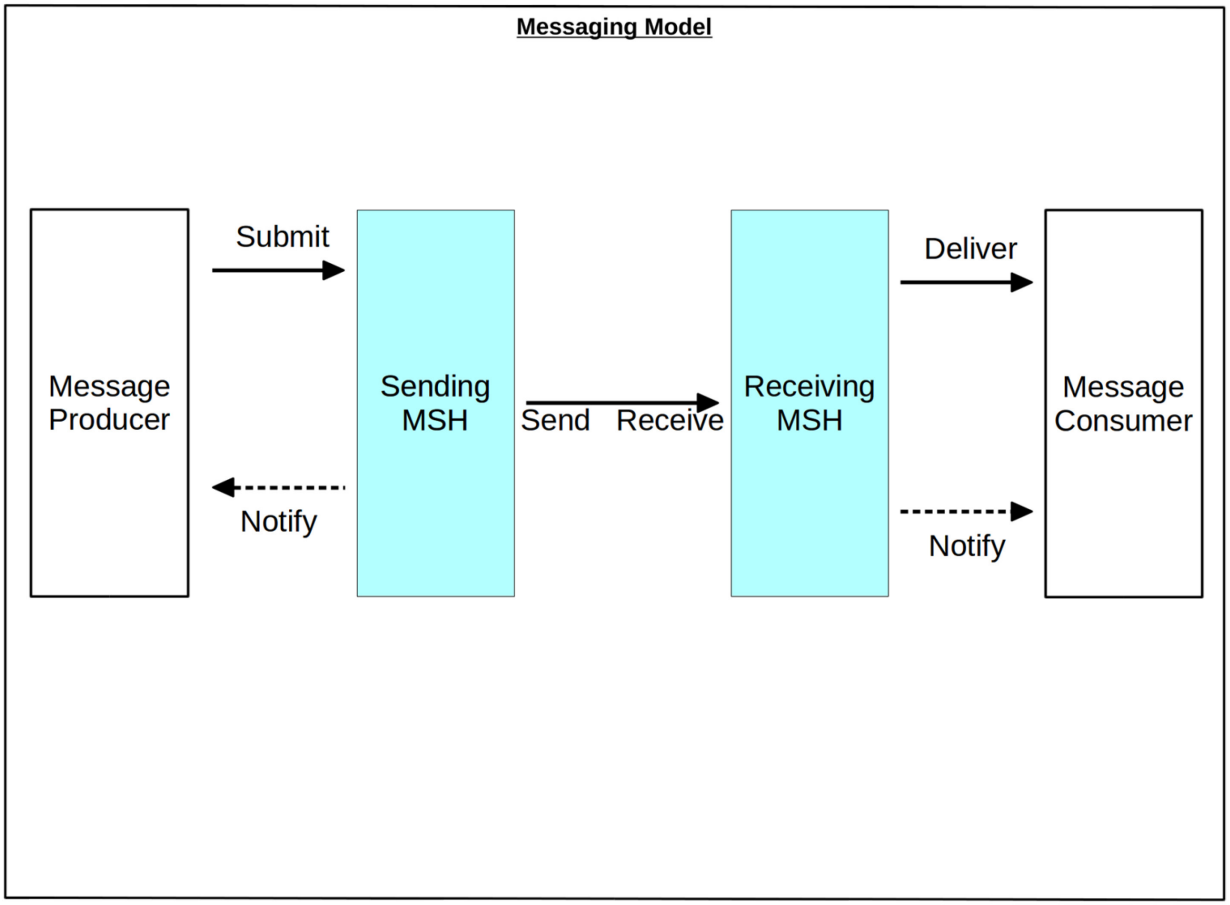

Figure 1: Entities of the Messaging Model and Their Interactions

5.1.2. Messaging roles

The Messaging Model assumes the following roles for an MSH:

- **Sending**: When an MSH acts in the *sending* role, it performs the functions associated with generating an ebMS user message and sending this message to another MSH. The abstract operations *Submit*, *Send* and *Notify* are supported by this role. (Note that even in a sending role, an MSH may be required to receive and process some types of signal messages, depending on the conformance profile in use.)

- **Receiving**: An MSH acting in the *receiving* role performs the functions associated with the receiving and processing of an ebMS user message. The abstract operations *Receive*, *Deliver* and *Notify* are supported by this role. (Note that even in a receiving role, an MSH may be required to generate and send ebMS signal messages related to the reception of messages, such as error messages or pull request signals.)

The transmission of an ebMS user message requires a pair of sending and receiving MSHs. Note that these roles are defined as only relevant to ebMS user messages, as are the abstract operations provided in 5.1.3.

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

5.1.3. Abstract messaging operations

An ebMS MSH supports the following abstract operations, depending on which role it is operating in:

- **Submit:** This operation transfers enough data from the producer to the sending MSH to generate an ebMS user message unit.
- **Deliver:** This operation makes data of a previously received (via *Receive* operation) ebMS user message unit available to the consumer.
- **Notify:** This operation notifies either a producer or a consumer about the status of a previously submitted or received ebMS user message unit, or about general MSH status.
- **Send**: This operation initiates the transfer of an ebMS user message from the sending MSH to the receiving MSH, after all headers intended for the receiving MSH have been added (including security and/or reliability, as required).
- **Receive**: This operation completes the transfer of an ebMS user message from the sending MSH to the receiving MSH. A successful reception means that a contained user message unit is now available for further processing by the receiving MSH.

5.2. Message exchange patterns

5.2.1. Rationale

Two communicating partners may agree to conduct business transactions as message sequences that follow well defined patterns, or message exchange patterns (MEP). Enforcing these patterns is usually done above the messaging layer. However it has proved useful to support some aspects of such MEPs in the messaging layer. In particular:

- The correlation between messages, when expressed directly via a referencing mechanism that appears in the message header, allows for efficient monitoring and enforcement of MEPs.
- As an MSH has to bind messages to the transport protocol, these binding requirements may be better expressed and controlled at MEP level. For example, different messages of the same MEP (such as a request and a response) may be required to bind differently to the transport.

An ebMS MEP represents the part of such exchange patterns that is controlled and implemented by an MSH, thus making an abstraction of the business semantics. Although the notion of MEP was not explicitly supported by ebMS 2.0, it can be noted that it provided some informal support for MEPs, such as message referencing (`eb2:RefToMessageId`) and the `eb2:SyncReply` element that controls the use of the back-channel of the underlying protocol. In the following, the acronym "MEP" implicitly means ebMS MEP, unless otherwise qualified.

The goal of this document is to introduce a model for ebMS MEPs, rather than a formal representation of them. This model is the basis for partners agreeing to which MEPs their exchanges will conform. Such agreements are manifested in processing modes, or P-Modes, the representation of which is outside the scope of this document. The P-Mode also defines which message profile is associated with which MEP, and the role it plays in this MEP. Processing modes are described in detail in Clause 7.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

5.2.2. General definition

An **ebMS MEP** defines a typical choreography of ebMS user messages which are all related through the use of the referencing feature (`eb:RefToMessageId`). Each message of an MEP instance refers to a previous message of the same instance, unless it is the first one to occur. Messages are associated with a label (e.g. "request", "reply") that precisely identifies their direction between the parties involved and their role in the choreography.

NOTE: Because `eb:RefToMessageId` more accurately defines a referencing between user message units than between user messages (SOAP messages), MEPs are preferably defined here as exchanges of message units, rather than of ebMS Messages.

Two MEPs are defined in this document, not exclusive of others:

- The **One-Way MEP** which governs the exchange of a single user message unit unrelated to other user messages. Its label is "one-way" and is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/oneWay`.

- The **Two-Way MEP** which governs the exchange of two user message units in opposite directions, the first one to occur is labeled "request", the other one "reply". In an actual instance, the "reply" shall reference the "request" using `eb:RefToMessageId`. This MEP is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/twoWay`.

The MEP definitions are primarily concerned with the transfer of ebMS user message units. Instances of such MEPs may involve or cause the transfer of additional messages or the piggy-backing of additional elements (e.g. ebMS signal messages or units such as errors, receipts, pull requests, and low-level acknowledgments when using reliability), but these are not taken into account in the MEP definition. Instead, the different ways these additions can be associated with the MEPs defined here, are considered as part of the execution mode of the MEP, which is controlled by some agreement/configuration external to the MEP definition (see P-Modes in Clause 7). Some extra messages (signal messages) may also be mandated by the binding of an ebMS MEP (see channel-binding), but are not relevant to the ebMS MEP definition itself.

MEP definitions in this document are restricted to exchanges between two MSHs.

5.2.3. MEP bindings

The previous definition of ebMS MEP is quite abstract, and ignores any binding consideration to the transport protocol. This is intentional, so that application-level MEPs can be mapped to ebMS MEPs independently from the transport protocol to be used. In addition to agreeing on MEP usage, the following notions of MEP bindings should be subject to agreements between partners:

- An **ebMS MEP Transport Channel Binding** defines how the MEP maps to the channels allowed by the underlying transport protocol, while making an abstraction of this underlying transport. In case of a two-way transport, the transport channel binding defines whether each message of the MEP maps to the fore-channel (or first leg) or back-channel (second leg). It also tells if an ebMS signal is needed to initiate the transfer - e.g. by pulling - and which one. Annex E shows possible options for combining headers supporting reliable messaging as well as error reporting, when binding basic ebMS MEPs to a two-way protocol such as HTTP. Annex E also shows how these combinations can be controlled with P-Mode parameters.

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

- An **ebMS MEP Transport Protocol Binding** defines further how an MEP transport channel binding is implemented over a specific underlying transport protocol such as HTTP or SMTP. For example, an HTTP transport protocol binding will define the usage of HTTP headers and methods for each message. A transport protocol binding usually relies on standard SOAP bindings when these exist.

A transport channel binding is a critical complement to an MEP, to be agreed on in order for partners to interoperate. The rationale in using different transport channel bindings for an ebMS MEP is to accommodate different connectivity constraints (e.g. firewall restrictions, intermittent availability, non-static IP address) by dictating how each message transfer is initiated over the underlying protocol. Because such connectivity constraints usually exist independently from the details of the transport protocol, the transport channel binding is the right level to address them. The transport channel bindings identified in this document are:

- **Push**: maps an MEP user message to the 1st leg of an underlying two-way transport protocol, or of a one-way protocol. This binding is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/push`.

- **Pull**: maps an MEP user message to the second leg of an underlying two-way transport protocol, as a result of an ebMS Pull Signal sent over the first leg. This binding is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pull`.

- **Sync**: maps an exchange of two user messages respectively to the first and second legs of a two-way underlying transport protocol. This binding is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/sync`.

NOTE 1: An underlying transport protocol qualifies as "two-way" if (a) it guarantees a transport channel for transferring the response of every message (request) initiated by an MSH, back to this MSH without need for explicit addressing information in SOAP headers, and regardless of connectivity restrictions such as inability to accept incoming new connections; and (b) it provides to the MSH initiator of the exchange, some means for correlating the response with the request, without relying on the SOAP header. For example, HTTP qualifies as two-way, but SMTP and FTP do not (although FTP has a notion of session, it does not inherently support the coupling of (b)). The channel offered in (a) is also called "back-channel" in this document.

NOTE 2: "Pull" and "sync" cannot be used with a one-way underlying protocol.

NOTE 3: Communicating parties should agree on a transport channel binding: a sending MSH will treat a message submitted for pulling differently from a message submitted for pushing.

An MEP that is associated with a particular transport channel binding is also called a transport-channel-bound MEP. A transport-channel-bound MEP is identified by a pair <MEP name / transport-channel-binding name>. For example, a two-way ebMS MEP that executes over a single request-response exchange of the underlying transport (e.g. HTTP), is called a **Two-Way/Sync** MEP.

A channel-bound MEP has an **initiating MSH**, or **initiator**, which is the one that triggers the execution of the MEP. The other MSH is called the **responding MSH**, or **responder**. These MSH roles do not change for the duration of the MEP, regardless of the number of messages exchanged and of their direction. Due to endpoint addressing or availability restrictions, some MSHs may be required to act only as initiator, and never as responder.

On the wire, the only method by which messages from the same MEP instance are associated is through a referencing link (`eb:RefToMessageId`). This referencing is decided above the MSH layer (by the producer entity). A receiving MSH relies on both this referencing and the interpretation of the P-Mode for associating a message with a specific MEP and for validating this association.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

5.2.4. Relationship to SOAP MEPs

In theory, the transport-channel-bindings previously defined could be expressed in terms of SOAP MEPs instead of channels of the underlying transport protocol. However, the notion of SOAP MEP has only been introduced with SOAP 1.2, and would need to be extended to SOAP 1.1.

Also, only the SOAP Request-Response MEP and Response MEP have been formally defined, as of the time this document was written. A SOAP One-way MEP could also be defined, but how such an MEP may or may not bind to a two-way underlying protocol is yet to be determined.

Expressing the transport-channel-binding in terms of SOAP MEPs is only helpful if there is a published, non-ambiguous, standard way for these to map to the underlying protocol(s). This is currently only the case for some SOAP MEPs and some transport protocols. Consequently, this document has chosen to express its transport-channel-bindings directly in terms of how to use the channels of the transport protocol, abstracting such a transport as either "one-way" or "two-way".

5.2.5. The One-Way/Push MEP

This transport-channel-bound MEP involves the transfer of a single ebMS user message unit (label: "one-way").

To conform to this MEP, the ebMS user message unit that is exchanged shall not relate to any other user message unit (no `eb:RefToMessageId` element). Figure 2 illustrates the exchange pattern and MSH operations involved in this MEP.

In case the One-Way/Push MEP is performed over a Two-way underlying transport protocol, the response message may carry an ebMS signal message, such as an error message, or other SOAP headers. Such an option is controlled by the P-Mode (see Clause 7). However, the response message shall not carry an ebMS user message that refers to the request message. If the P-Mode allows faults to be reported on the two-way protocol's back-channel, the MEP can be qualified as a **robust** MEP, but is still an ebMS One-Way/Push MEP.

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

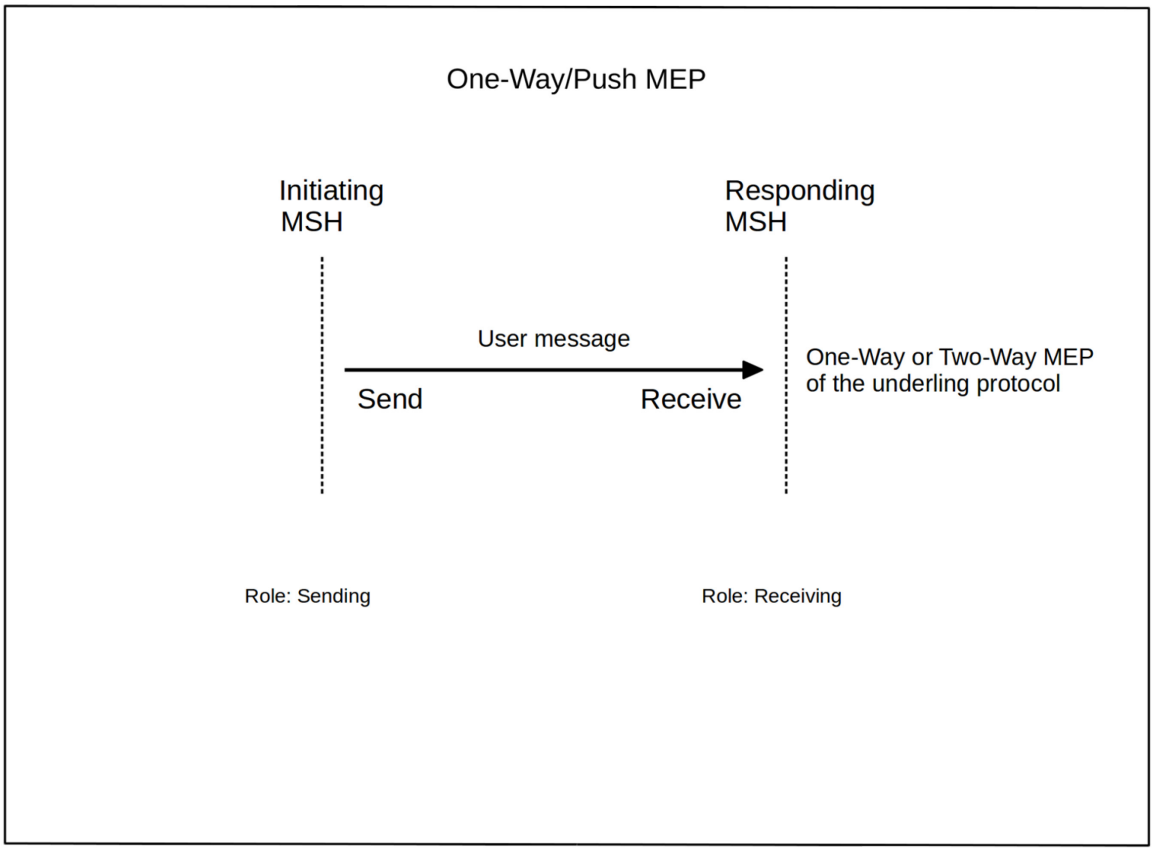

Figure 2: One-Way/Push MEP

5.2.6. The One-Way/Pull MEP

This transport-channel-bound MEP involves the transfer of a single ebMS user message unit (label: "one-way"). This MEP is initiated by the receiving MSH, over a two-way underlying transport protocol. The first leg of the protocol exchange carries a pull signal message. The second leg returns the pulled user message unit. To conform to this MEP the pulled user message unit shall not include an `eb:RefToMessageId` element. In case no message is available for pulling, an ebMS error signal of severity level "warning" and short description of "EmptyMessagePartitionChannel", as listed in 9.7.2, shall be returned over the response leg. Figure 3 illustrates this MEP.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

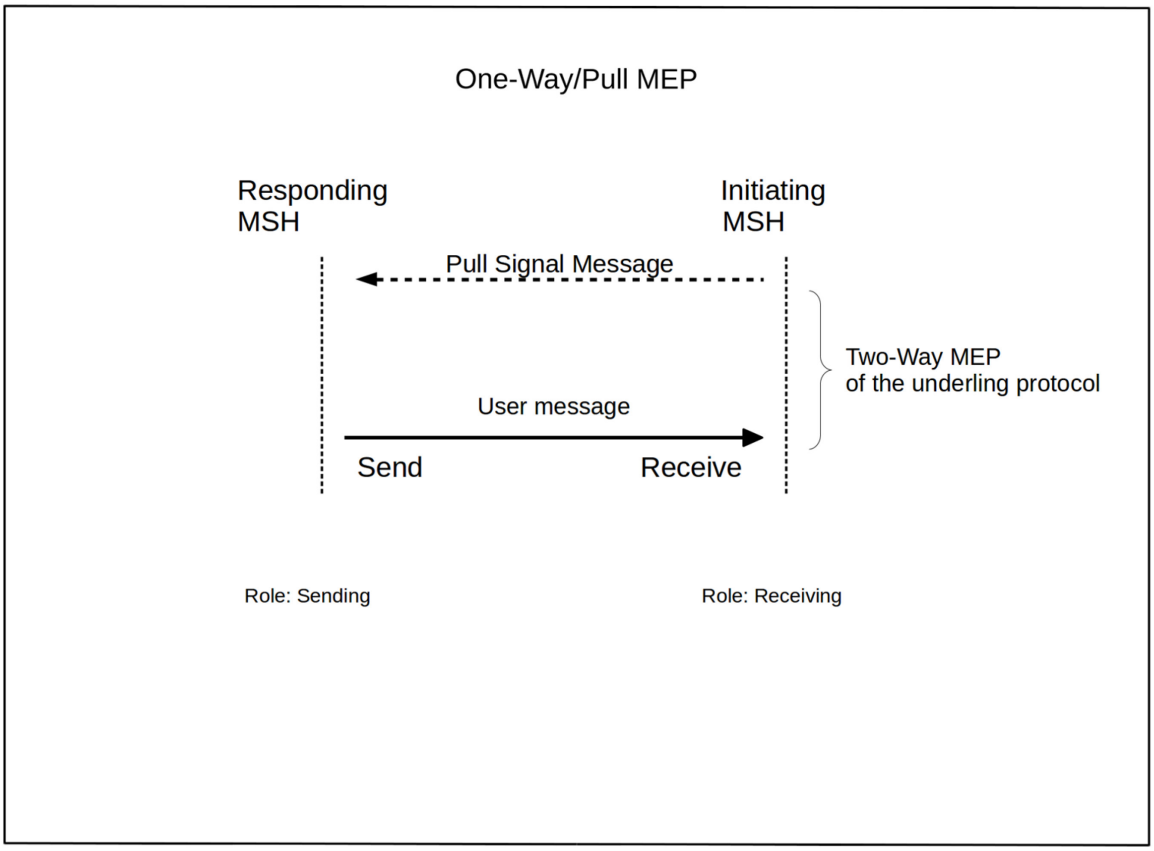

Figure 3: One-Way/Pull MEP

5.2.7. The Two-Way/Sync MEP

This transport-channel-bound MEP transfers two ebMS user message units over a single request-response exchange of the underlying two-way transport protocol. The initiator MSH sends the first user message called the "request". In the second leg of the MEP, a related user message unit called the "reply" is sent back. To conform to this MEP, the request shall not relate to any other user message unit (no `eb:RefToMessageId` element), and the reply shall refer to the request via the `eb:RefToMessageId` header element, as described in 8.2.3.2. Figure 4 illustrates this MEP.

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

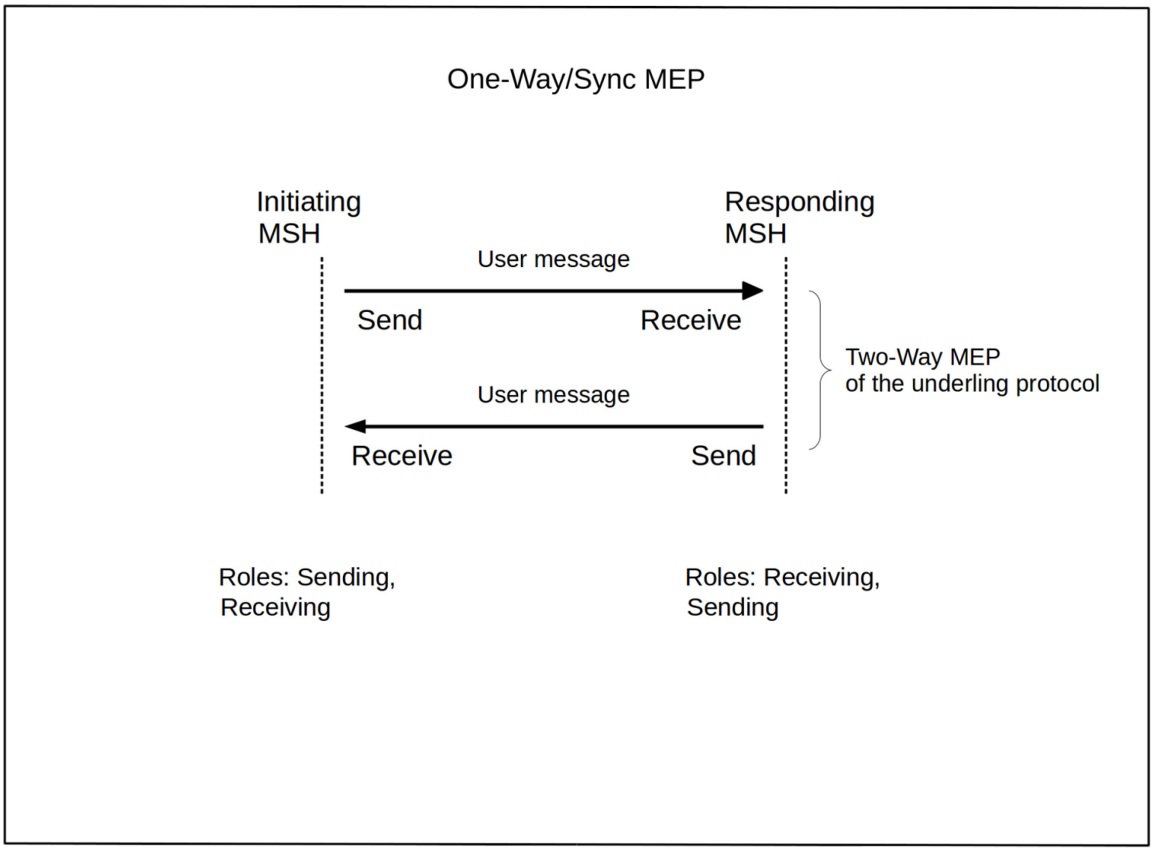

Figure 4: Two-Way/Sync MEP

5.2.8. Other transport-channel-bound MEPs

So far, message exchanges conforming to either one of the three previous transport-channel-bound MEPs were only using one instance of an underlying transport protocol MEP for their binding. The following channel-bound ebMS MEPs are expected to be among the most common ebMS MEPs that use more than one underlying transport protocol MEP instance between the Initiating MSH and the Responding MSH. In this sense, they may be qualified as asynchronous.

- The **Two-Way/Push-and-Push MEP** composes the choreographies of two One-Way/Push MEPs in opposite directions, the user message unit of the second referring to the user message unit of the first via `eb:RefToMessageId`. This MEP is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pushAndPush`.

- The **Two-Way/Push-and-Pull MEP** composes the choreography of a One-Way/Push MEP followed by the choreography of a One-Way/Pull MEP, both initiated from the same MSH (Initiator). The user message unit in the "pulled" message shall refer to the previously "pushed" user message unit. This MEP is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pushAndPull`.

- The **Two-Way/Pull-and-Push MEP** composes the choreography of a One-Way/Pull MEP followed by the choreography of a One-Way/Push MEP, with both MEPs initiated from the same MSH. The user message unit in the "pushed" message shall refer to the previously "pulled" user message unit. This MEP is identified by the URI `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pullAndPush`.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

In all these MEPs, the messages labels are respectively "request" and "reply". The two last MEPs support asynchronous exchanges where one party is constrained in terms of addressing or connectivity capability.

6. Message pulling and partitioning

6.1. Objectives

Business partners may experience differences in their ability to handle message flow, intermittent connectivity, lack of static IP addresses or firewall restrictions. In addition, when a message is transferred and successfully acknowledged, the responsibility for its management shifts sides. For these reasons, a receiver may want (a) to retain control over the transfer procedure of the underlying protocol by initiating transfers, and/or (b) to decide which messages it wants to receive first and when. Two features have been introduced in ebMS 3 that support this:

- Message pulling
- Message partition channels (MPCs)

Message pulling is defined in an abstract way by the One-Way/Pull ebMS MEP (see 5.2.6). This MEP allows an MSH to initiate the transfer of a message as a receiver. When used in combination with the One-Way/Push ebMS MEP, it allows an MSH full control over initiating asynchronous transfers with another MSH in both directions, engaging in a client-server type of interaction with the remote MSH, without any need to open a TCP/IP port to incoming requests. This MEP also supports exchanges with a partner that is intermittently connected: instead of periodically polling for partner presence, a sending MSH will simply wait for the partner MSH to pull its messages.

Example*: A mobile, occasionally connected device without static IP address and with limited storage capability can only initiate requests and receive messages as synchronous responses to these requests. The One-Way/Pull MEP allows this device to enable and control the flow of received messages, and to adjust it to its own resources.*

Message partition channels (see 6.4) allow for partitioning the flow of messages from an MSH to another MSH into separate flows, so that each one of these flows can be controlled independently by either MSH, in terms of transfer priorities. A sending MSH shall be able to determine whether a submitted message should be pulled or pushed, and to which message partition channel (MPC) it shall be assigned. Similarly, the receiving MSH is aware of which MPC(s) should be pulled from, and which ones will be used for push. This knowledge is based on an agreement shared between parties prior to the exchanges, and modeled in this document as the P-Mode operation set (see clause 7).

6.2. Supporting message pulling

Using message pulling requires the ability of an MSH to support the One-Way/Pull MEP. The pull request signal that initiates this MEP is described in 8.2.4.2. Because there is always at least one MPC open between a sending MSH and a receiving MSH –the default MPC– the Pull mode can be supported regardless of the ability to support several MPCs.

When sending an `eb:PullRequest` signal, the name of the MPC to pull messages from shall be specified (in `eb:PullRequest/@mpc` attribute), unless the default value is to be assumed.

The processing model for a pulled message is as follows, for a typical and successful instance of One-Way/Pull MEP:

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

On responding MSH side:

1. Submit: submission of message data to the MSH by the producer party, intended for the consumer on the Initiator side. The message is associated with an MPC. If no MPC name is provided by the submitter, or if the MSH implementation has not been provided with a way to determine this association by itself, the default MPC is used. The MEP associated with this message (e.g. as specified by P-Mode.MEP; see 7.3) is a One-Way/Pull.

On initiating MSH side:

2. Sending of a pull request signal by the MSH. The pull request signal specifies the MPC from which to pull messages.

On responding MSH side:

3. Reception of the pull request signal. For every pull request signal received the Responder MSH (acting in sending role) selects a previously submitted message. Messages should be selected according to a FIFO policy with respect to the *Submit* operation. If there is no user message available in the specified MPC for sending, a warning signal with short description: "EmptyMessagePartitionChannel" (see 9.7.2) shall be sent back instead.

4. Send: the selected message is sent over the SOAP response to the pull request.

On initiating MSH side:

5. Receive: the pulled message is available for processing by the MSH. The header `@mpc` attribute indicates from which MPC it has been pulled, and is the same as the value of `@mpc` in the corresponding pull request signal.

6. Deliver: after processing of ebMS headers, delivery of the pulled message data to the consumer of the MSH.

Example 1: An example of `eb:Messaging` header for the pull request signal:

```
<S11:Envelope>
<S11:Header>
<eb:Messaging S11:mustUnderstand="1">
       <eb:SignalMessage>
              <eb:MessageInfo>
                     <eb:Timestamp>2006-10-01T10:01:00</eb:Timestamp>
                     <eb:MessageId>UUID-4@receiver.example.com</eb:MessageId>
              </eb:MessageInfo>
              <eb:PullRequest mpc="http://sender.example.com/mpc123"/>
       </eb:SignalMessage>
</eb:Messaging>
</S11:Header>
<S11:Body/>
</S11:Envelope>
```

Example 2: An outline of `eb:Messaging` header for the response to the pull request signal example from Example 1:

```
<S11:Envelope>
<S11:Header>
<eb:Messaging S11:mustUnderstand="1" >
   <eb:UserMessage mpc="http://sender.example.com/mpc123">
      <eb:MessageInfo>
         <eb:Timestamp>2006-10-01T10:02:00</eb:Timestamp>
         <eb:MessageId>UUID-5@sender.example.com</eb:MessageId>
```

Documents
Document Sharing Hub

© ISO 2021 – All rights reserved

Full document details: BS ISO 15000-1:2021

```
         <eb:RefToMessageId>UUID-4@receiver.example.com</eb:RefToMessageId>
      </eb:MessageInfo>
      <eb:PartyInfo>
         ...
      </eb:PartyInfo>
      <eb:CollaborationInfo>
         ...
      </eb:CollaborationInfo>
      <eb:PayloadInfo>
         ...
      </eb:PayloadInfo>
   </eb:UserMessage>
</eb:Messaging>
</S11:Header>
<S11:Body>
...
</S11:Body>
</S11:Envelope>
```

6.3. Combining pulling with security and reliability

Reliability of a pulled message is usually associated with the reliability of the corresponding pull request signal. The reliability of the One-Way/Pull MEP instance is addressed in 11.3.

Security for the pull request signal is described in details in 10.12.

Example 1: An outline of a secure and reliable `eb:Messaging` header for the pull request signal follows. The reliability header used in the example assumes the use of WS-Reliability, and specifies At-Least-Once delivery, with an acknowledgment to be returned on the MEP response message:

```
<S11:Envelope>
<S11:Header>
<eb:Messaging S11:mustUnderstand="1" >
       <eb:SignalMessage>
              <eb:MessageInfo>
                     <eb:Timestamp>2006-10-01T10:01:00</eb:Timestamp>
                     <eb:MessageId>UUID-4@receiver.example.com</eb:MessageId>
              </eb:MessageInfo>
              <eb:PullRequest mpc="http://sender.example.com/mpc123"/>
       </eb:SignalMessage>
</eb:Messaging>
<wss:Security>
       ...
</wss:Security>
<wsr:Request S11:mustUnderstand="1">
       ...
       <ReplyPattern>
              <Value>Response</Value>
       </ReplyPattern>
       <AckRequested/>
       ...
</wsr:Request>
</S11:Header>
<S11:Body/>
</S11:Envelope>
```

Example 2: An outline of secure and reliable `eb:Messaging` header for the response to the pull request signal in Example 1:

```
<S11:Envelope>
<S11:Header>
<eb:Messaging S11:mustUnderstand="1" >
   <eb:UserMessage mpc="http://sender.example.com/mpc123">
```

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

```
      <eb:MessageInfo>
         <eb:Timestamp>2006-10-01T10:02:00</eb:Timestamp>
         <eb:MessageId>UUID-5@sender.example.com</eb:MessageId>
         <eb:RefToMessageId>UUID-4@receiver.example.com</eb:RefToMessageId>
      </eb:MessageInfo>
      <eb:PartyInfo>
      ...
      </eb:PartyInfo>
      <eb:CollaborationInfo>
      ...
      </eb:CollaborationInfo>
      <eb:PayloadInfo>
      ...
      </eb:PayloadInfo>
   </eb:UserMessage>
</eb:Messaging>
<wsr:Response S11:mustUnderstand="1">
       ...
</wsr:Response>
<wss:Security>
       ...
</wss:Security>
</S11:Header>
<S11:Body>
...
</S11:Body>
</S11:Envelope>
```

NOTE: In example 2, the reliability header, which assumes the use of WS-Reliability, is a `wsr:Response` element. It contains the reliability acknowledgment for the pull request signal. In this example there is no `wsr:Request` reliability header. A `wsr:Request` header could be present, in addition to `wsr:Response`, in case some specific reliability requirement is associated with the pulled message (see 11.3).

6.4. Message partition channels

6.4.1. Concept and purpose

Message partition channels (MPCs) allow for partitioning the flow of messages from a sending MSH to a receiving MSH into several flows that can be controlled separately and consumed differently. They also allow for merging flows from several sending MSHs, into a unique flow that will be treated as such by a receiving MSH. In particular, MPCs allow for:

1. setting transfer priorities: some messages may be transferred with higher priority than others regardless in which order they all have been submitted. For example, when using pulling mode, a receiving MSH may decide from which MPC to pull messages first, based on business needs and readiness to incur responsibility in managing these messages.

2. organizing the inflow of messages on receiving side, so that each flow can be consumed in a distinct way, yet without having to filter messages based on various header elements or payload content. The agreement between two parties on when messages are to be transferred and how they are to be consumed may then be reduced to which MPC will be used.

NOTE 1: The notion of MPC is abstract from any particular implementation device such as ports or queues: an implementation can implement MPCs using queues and a FIFO policy, though it is not required to.

NOTE 2: Although MPCs are most obviously beneficial to message pulling operations, MPCs can be used in association with pushed messages as well. The benefits of doing so, listed above, apply to the push case as well.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

***Example**: A pair of business partners – a large buyer and a small supplier - have decided to create two MPCs for transferring messages sent by the buyer. Urgent messages that require immediate processing (e.g. high priority purchase orders, and updates to prior purchase orders) are assigned to one MPC; and less urgent messages (payments, catalog requests, confirmations, acknowledgments of receipts, etc.) are assigned to the other MPC. The buyer determines the level of urgency of a posting, which can be manifested inside the message. Per an agreement with the buyer, the supplier will pull and process first all messages from the "urgent" MPC; then, once that is exhausted, only the messages from the less urgent MPC. This way, the low-capacity receiving MSH (supplier) is able to prioritize the reception of its messages, focusing its resources on the most urgent messages and avoiding the overhead and risk in managing (persistence, recovery, security) less urgent but important messages that it cannot process in the short term.*

Any more complex filtering mechanism that requires checking a filter condition on header data, is out of scope of this document. Such filtering could be implemented in a sending MSH and/or in a receiving MSH as a complement to, or instead of, different MPCs. The notion of MPC is a simple and robust solution with low interoperability risk: it allows for partitioning messages based on prior agreement between producer and consumer on which type of message will use which MPC, without a need to communicate and process filter expressions for each message transfer.

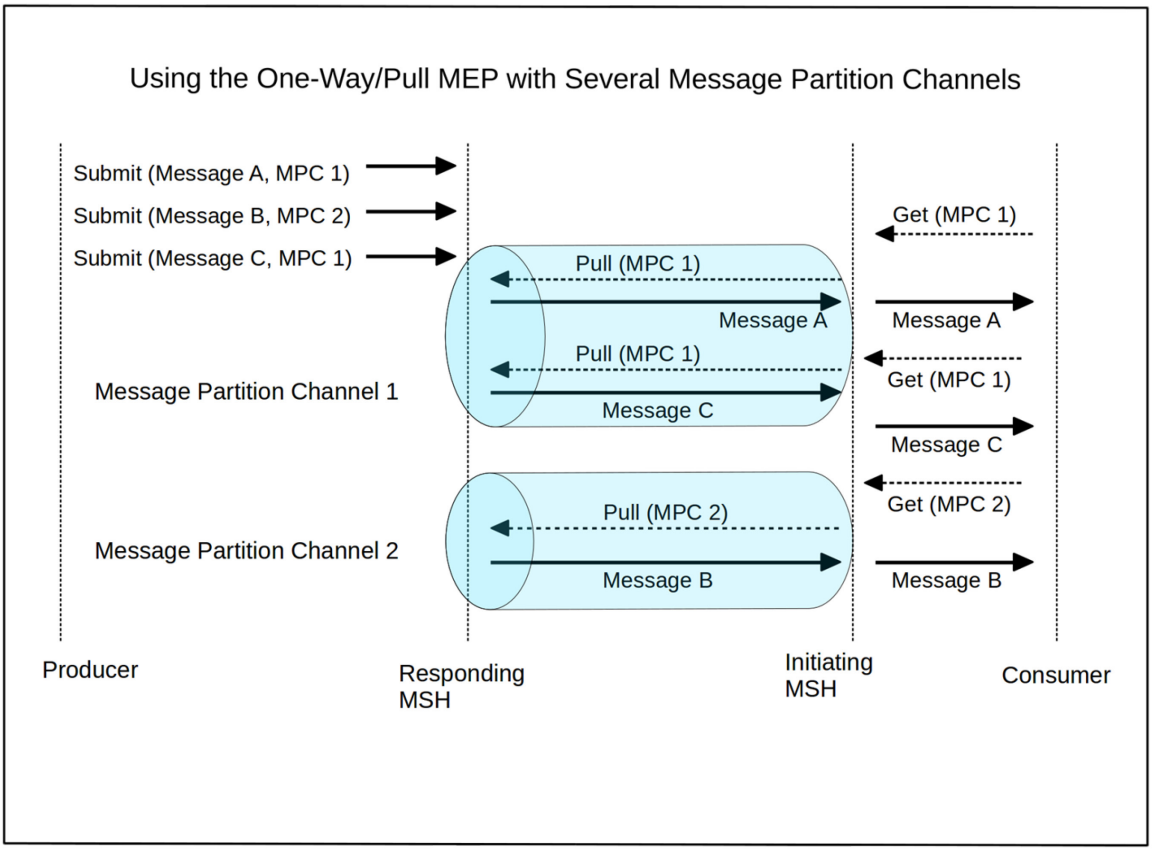

Figure 5: One-Way/Pull with Message Partition Channels

Figure 5 illustrates how MPCs and the One-Way/Pull MEP can be used by a consumer party to control the order of the messages it wants to receive and process. Messages on MPC 1 are "pulled" in priority by the consumer side.

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

There is no requirement for ordering messages in an MPC, unless specified otherwise by the reliability requirements to which these messages are subjected. The transfer of messages over an MPC is controlled by:

- The MEPs in which these messages participate. Messages over the same MPC can either be pulled or pushed, based on the different MEPs that govern the transfer of these messages.
- The regular addressing means used for sending messages (e.g. URL of receiving MSH when pushing messages). MPCs do not have any routing or addressing capability.

Before it is transferred from a sending MSH to a receiving MSH, regardless of whether it is pushed or pulled, a message is always assigned to an MPC. If no explicit assignment is requested (e.g. by the message producer at submission time or per configuration of the MSH), the default MPC name `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultMPC` is assigned.

6.4.2. Some use cases

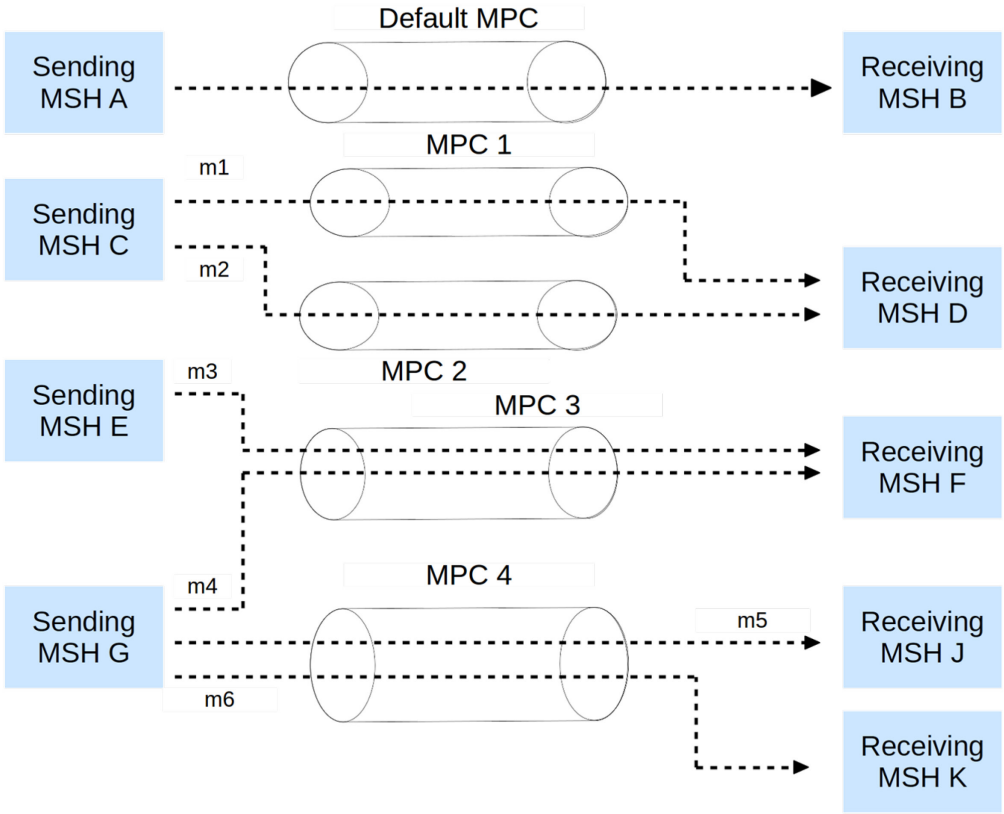

Figure 6: Message Partition Channel Use Cases

Figure 6 illustrates various cases in using MPCs. In Figure 6, each arrow represents the transfer of a user message, which could be either pushed or pulled.

Between MSHs A and B, no MPC has been explicitly defined or assigned. All messages transferred from A to B – whether pushed or pulled – will implicitly use the default MPC.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

MSHs C and D have been configured to use MPCs 1 and 2 (in addition to the default MPC). Messages sent may be assigned to either one of these MPCs. In case these messages are pulled, MSH D may choose from which MPC to pull first.

MPC 3 is shared by two sending MSHs, E and G. The effect of using this MPC is to define on the receiving MSH F a merged inflow of messages from E and G, which may be presented to the consumer as a single flow. If messages m3 and m4 are pulled, MSH F has control over which MSH from which to pull first.

MPC 4 is used by MSH G to send either to MSH J or MSH K. When combined with message pulling, this use case allows for various scenarios. For example, the message flow might initially go exclusively from G to J. In case MSH J fails, another MSH (K) may immediately take over the message flow without any change on the sender side (assuming K is authorized) nor any knowledge by K of where the initial flow was intended for. Or, two receiving MSHs (J and K) that are remote from each other but used by equivalent applications may split the processing of messages submitted to the same sending MSH G. This may be, for example, two agencies equally qualified to process trouble tickets, indiscriminately pulling messages from the same MPC at the pace allowed by their processing capacity. MPC 4 may also be used by concurrent, pushed message flows. Using the same MPC does not introduce any dependency between the processing of m5 and m6 in J and K, but may be associated with a particular business meaning (i.e. is meaningful to consumers of J and K).

6.4.3. Definition and usage requirements

An MPC is a flow of messages from a set of sending MSHs to a set of receiving MSHs, in the sense given in flow networks theory. It is identified by a name–a string of characters–that is assigned to every message of the flow. For every message it sends or receives, an MSH shall be aware of which MPC this message is assigned to. MPC is a dynamic notion, the elements of which do not need to be fully defined prior to initiating this flow. For example, additional MSHs (either sending or receiving) may join the flow at any time, assuming they have knowledge of the MPC name, and assuming there is no other reason preventing them from transferring messages over this MPC (e.g. security).

The association between a user message and an MPC is apparent in the ebMS header of the message (see 8.2). Except for the default MPC, the MPC name shall appear in the header of a user message transferred over this MPC.

NOTE: As defined above, an MPC can involve more than a sending MSH and a receiving MSH. In particular, two unrelated pairs of sending/receiving MSHs (e.g. in the previous figure, C and D on the one hand, E and F on the other hand) could transfer messages using the same MPC name (e.g. MPC 3 in the figure could also be renamed MPC 2). Formally speaking, all these messages would be transferred over the same MPC. There can be some business significance in deciding whether two pairs of MSHs that have unconnected message flows use the same MPC to transfer these messages, even though as far as the MSHs are concerned, they will process these two separate sub-flows of messages independently from each other.

Only user messages may be assigned to MPCs, not signal messages.

A pull request signal message always indicates in its header (see 8.2.4.2) the MPC on which the message shall be pulled. If no MPC is explicitly identified, the default MPC shall be pulled from. The pulled message sent in response shall have been assigned to the indicated MPC.

The association of a message with an MPC shall be done either at submission time, e.g. requested by the message producer; or at any time between *Submit* and *Send*, e.g. based on configuration or processing mode (see Clause 7). This is left to the implementation.

Support for assigning messages to MPCs–e.g. by automatically mapping messages submitted by a producer to a particular MPC based on some rules, queries or filters–is out of scope of this document. Similarly, there is no requirement on what criteria (e.g. query expression, FIFO policy) can be used to

AVA Documents
Document Sharing Hub

© ISO 2021 – All rights reserved

Full document details: BS ISO 15000-1:2021

select messages when pulling messages from an MPC. This document only describes the properties of MPCs, and how their use affects the message protocol. It does not prescribe a particular way to implement MPCs or to use them.

A message associated with an MPC could fail to be transferred for various reasons (transport issue, security, intermediaries, etc.) and therefore could be removed from the MPC at any time. In other words, there is no additional delivery contract for messages over an MPC, other than that specified by the reliability agreement.

There is no specific quality of service associated with an MPC. Security and reliability remain associated with parties or with MSHs, in a way that is orthogonal to MPCs; although an implementation is free to associate QoS with MPCs as long as this conforms to an agreement between parties.

7. Processing modes

7.1. General

An MSH is operating–either for sending or receiving messages–with knowledge of some contextual information that controls the way messages are processed. This contextual information that governs the processing of a particular message is called processing mode (or P-Mode). Because different messages may be subject to different types of processing, an MSH generally supports several P-Modes.

A P-Mode represents some MSH input data that typically is not provided on a per-message basis, but that is common to a set of messages exchanged between or among parties. To this extent, the P-Mode may be interpreted as configuration data for a deployed MSH. On a sending MSH, together with the information provided by the application layer for each submitted message, the P-Mode fully determines the content of the message header. For example, the "security" part of the P-Mode will specify certificates and keys, as well as which messages will be subject to these. This in turn will determine the content of the `wsse:Security` header. The set of all P-Modes that are supported by an MSH during operation is called the P-Mode operation set of the MSH.

The association of a P-Mode with a message may be based on various criteria, usually dependent on header data (e.g. `eb:Service`, `eb:Action`, `eb:ConversationId`, or other message properties). Which security and/or which reliability protocol and parameters, as well as which MEP is being used when sending a message, is determined by the P-Mode associated with this message.

A data model for P-Modes is described in Annex D. Although this document does not require support for any particular representation of a P-Mode, a conformance profile for this document may require support for a particular representation. An MSH shall conform the processing of its messages to the values in the P-Mode associated with this message. The details of which P-Mode parameters shall be supported by an implementation is governed by the features associated with the conformance profile claimed by this implementation, i.e. by its profile feature set (see Annex G on Conformance). An MSH shall not process a message to normal completion if it has no matching P-Mode in its P-Mode operation set: i.e. the MSH shall not deliver such a message when in receiving role, or shall not send it when in sending role. When it cannot match a message to a P-Mode, an MSH shall generate a ProcessingModeMismatch (`EBMS:0010`) error.

NOTE: It is important to distinguish between Conformance Profiles (Annex G) and P-Modes. A conformance profile qualifies an MSH implementation and does not vary with the usage made of the MSH. A P-Mode qualifies the dynamic exchange and processing of messages, and is generally user defined. A P-Mode is set within the capabilities allowed by the conformance profile claimed by the MSH on which it is deployed.

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

7.2. Messaging service processing model

Although different P-Modes may apply from one message to the other, the overall processing model remains the same for all messages. The P-Modes set may be seen as configuring the execution parameters for the general model.

The ebXML messaging service may be conceptually broken down into the following three parts:

1. An abstract service interface,
2. Functions provided by the MSH and
3. The mapping to underlying transport service(s).

Figure 7 depicts a logical arrangement of the functional modules existing within one possible implementation of the ebXML Messaging Services architecture. These modules are arranged in a manner to indicate their inter-relationships and dependencies.

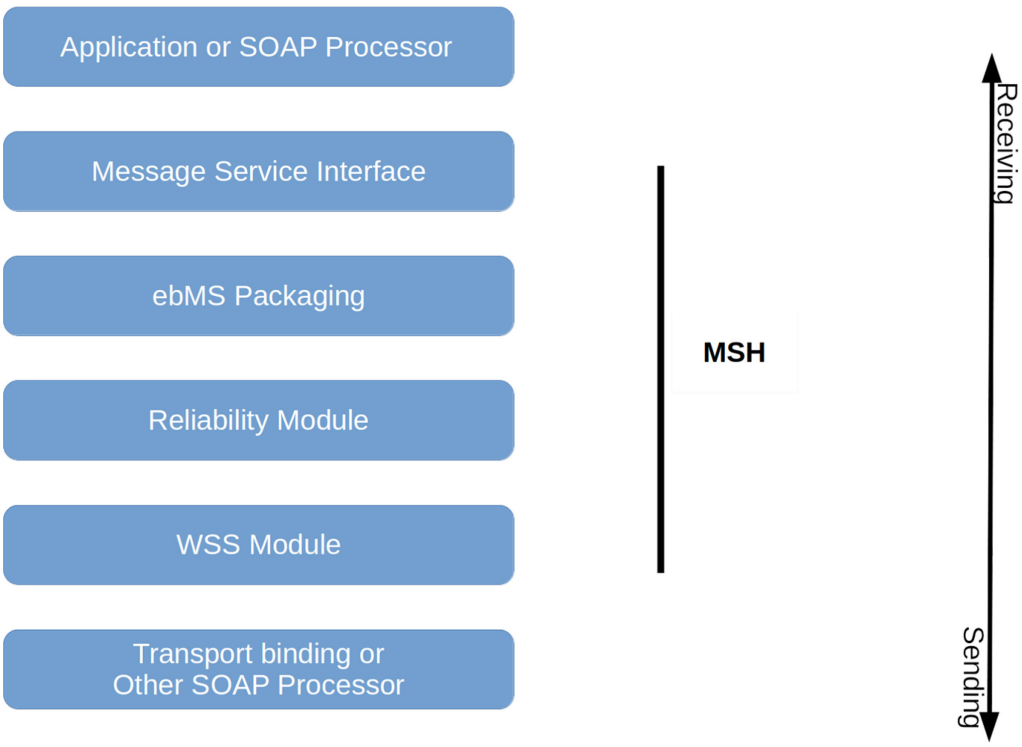

Figure 7: Component Relationships

Following is a description of each module illustrated in Figure 7. It should be noted that the stack diagram in the Figure is abstract, and this document does not mandate that implementations adopt the architecture suggested by it, although the processing order shown here should be used, especially in regard to security and reliability modules.

- **Application or SOAP processor** - This is where the business logic for a message exchange / business process exists.
- **Messaging service interface** - This is the interface through which messages are channelled between the MSH core and the ebXML Application.

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

- **ebMS packaging** - Handling, (de)enveloping and execution of payload services are performed by this module.
- **Reliable message processing** - This module fulfills the quality of service requirements for a message.
- **Web Services security processing** - Encryption/decryption of any SOAP message content and generation/verification of any digital signatures occurs in this module.
- **Transport protocol bindings** - These are the actual transport protocol bindings. This document defines bindings for HTTP and SMTP in Annex C, and supports the addition of other protocols.

7.3. Processing mode features

The P-Mode is partitioned into functional groups called P-Mode features. Each P-Mode feature covers one of the functional areas that is critical to achieving interoperability between two partners: security, reliability, transport, business collaboration, error reporting, message exchange patterns (MEPs) and message partition channels (MPCs).

The main P-Mode features are here identified by names of the form: P-Mode.<featurename>:

- **P-Mode.Protocol**: includes all transport related information that is necessary to achieve transport-level interoperability. This feature determines the type of transport involved (e.g. HTTP, SMTP, FTP) between two MSHs, and related configuration parameters. This feature usually treats all messages between two MSHs similarly. It also includes information about which SOAP version is to be used (SOAP 1.1 or SOAP 1.2).
- **P-Mode.Reliability**: includes all reliability contracts, or references to them, that will govern the reliability of messages exchanged. This feature determines the content of the reliability headers.
- **P-Mode.Security**: includes all security contracts, or references to them, including the security context and related resources (certificates, SAML assertions, etc.) that govern the message exchange. This feature determines the content of the `wsse:Security` header.
- **P-Mode.BusinessInfo**: includes all message-relevant data related to a collaboration between two parties. It also indicates which MPCs are to be used by these parties. This feature will complement or validate message data that is expected to be provided by the application on a per-message basis for these header elements:
    - `eb:UserMessage/eb:PartyInfo`
    - `eb:UserMessage/eb:CollaborationInfo`
    - `eb:UserMessage/eb:MessageProperties`
- **P-Mode.ErrorHandling**: defines how each ebMS Error type is to be reported by this MSH. E.g. if the reporting is done using ebMS signal messages, it defines the address of the destination MSH. It also may include the policy chosen for raising ebMS Errors from the errors generated by functional modules (reliability, security). This P-Mode feature shall define reporting mode parameters that will allow a receiving MSH to decide:
    - whether an error generated on reception of a message shall be returned as response over the same SOAP MEP. (e.g. **errorHandling.report.asResponse** = true/false).

© ISO 2021 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021

- whether an error generated on reception of a message shall be returned to sender or to a third party over a new SOAP MEP. (e.g. **errorHandling.report.ReceiverErrorsTo** = <URL>).

- whether the consumer and/or producer (e.g. **errorHandling.Report.ProcessErrorNotifyConsumer**) of a message shall be notified of an error generated on reception of the message.

In this document, a P-Mode feature is abstractly considered to apply to both sending and receiving roles, although implementations may choose to represent only the subset relevant to the role in which they operate. A single P-Mode instance is also intended to govern all messages involved in an ebMS MEP. (The ebMS MEP and its transport channel binding are attributes of a P-Mode.) Because messages involved in an MEP (e.g. request and reply) may use different qualities of service, a single P-Mode may use different vectors of values for its parameters, depending on the message in the MEP. An outline of the data model for P-Modes is given in Annex D.

Agreeing on a P-Mode operation set is essential for two parties in order for their MSHs to interoperate. P-Modes are the MSH-level expression of a prior agreement between partners. A reference to such an agreement may be present in the message header (see `eb:AgreementRef` element in 8.2.3.8).

7.4. Default features for processing mode

In order to facilitate interoperability testing, or during the early phase of a deployment, it may be useful to drive message exchanges without relying on user-agreed P-Modes, without interfacing with any application, and (initially) without the added complexity of security and reliability features. To this end, a default semantics of each P-Mode feature is defined as follows:

- **Default P-Mode.MEP:** `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/oneWay`:

- **Default P-Mode.MEPbinding**: `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/push`

- **Default P-Mode.Protocol**: HTTP 1.1 transport is assumed, with default configuration (on standard port), using SOAP 1.2.

- **Default P-Mode.Reliability**: No reliable messaging assumed (no reliability header will be present).

- **Default P-Mode.Security**: No secure messaging assumed (no security header will be present.)

- **Default P-Mode.BusinessInfo**: In the absence of any application input at message level as well as for this P-Mode feature, the following default header element values will be used (shown here for a message sent by an initiator to a responder party). Any of these may be overridden by application input.

  - `eb:UserMessage/eb:PartyInfo`: The `eb:From` element contains a PartyId with value: `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultFrom`.
    The `eb:To` element contains a PartyId with value: `http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultTo`.

© ISO 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 15000-1:2021