ebCore Agreement Update Specification Version 1.0

Committee Specification 01

18 September 2016

Specification URIs

This version:

http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/ebcore-au-v1.0-cs01.odt (Authoritative)
http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/ebcore-au-v1.0-cs01.html
http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/ebcore-au-v1.0-cs01.pdf

Previous version:

http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/csprd01/ebcore-au-v1.0-csprd01.odt (Authoritative)
http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/csprd01/ebcore-au-v1.0-csprd01.html
http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/csprd01/ebcore-au-v1.0-csprd01.pdf

Latest version:

http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/ebcore-au-v1.0.odt (Authoritative)
http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/ebcore-au-v1.0.html
http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/ebcore-au-v1.0.pdf

Technical Committee:

OASIS ebXML Core (ebCore) TC

Chairs:

Pim van der Eijk (pvde@sonnenglanz.net), Sonnenglanz Consulting
Sander Fieten (sander@fieten-it.com), Individual

Editors:

Pim van der Eijk (pvde@sonnenglanz.net), Sonnenglanz Consulting
Theo Kramer (theo@flame.co.za), Flame Computing Enterprises

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- Agreement Update XML schema: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/schema/ebcore-au-v1.0.xsd
- Certificate Update XML schema: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/schema/ebcore-cu-v1.0.xsd
- Schema data dictionaries: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/documentation/
- XML Signature 1.1 schema driver: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/schema/xmldsig1-schema.xsd
- XML document samples: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/samples/

Related work:

This specification is related to:

- *Collaboration-Protocol Profile and Agreement Specification Version 2.0*. OASIS Standard, 23 September 2002. .https://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf

- *Collaboration-Protocol Profile and Agreement Specification Version 3.0 (CPPA3)*. OASIS Working Draft. https://www.oasis-open.org/committees/download.php/57550.
- *Message Service Specification Version 2.0*. OASIS Standard, 01 April 2002. http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.
- *OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features*. OASIS Standard, 01 October 2007. Latest version: http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.html.
- *AS4 Profile of ebMS 3.0 Version 1.0*. OASIS Standard, 23 January 2013. Latest version: http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/profiles/AS4-profile/v1.0/AS4-profile-v1.0.html.

Declared XML namespaces:

- http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0
- http://docs.oasis-open.org/ebcore/ns/CertificateUpdate/v1.0

Abstract:

The ebCore Agreement Update specification defines message exchanges and an XML schema to support the exchange of messaging service communication agreement update requests and the associated responses to such requests. The schema offers extensibility for various types of updates. The main initial application of the specification is the exchange of X.509 certificates for certificate rollover, for which a separate extension schema is provided. The specification is based on the concept of messaging service communication agreements and the creation of new agreements as independently identified updated copies of existing agreements. The specification also provides an Agreement Termination feature. The specification supports ebMS2, ebMS3 and AS4 but can also be used with other protocols that have a concept of communication agreement. The specification is independent of storage or interchange formats for configuration information.

Status:

This document was last revised or approved by the OASIS ebXML Core (ebCore) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebcore.

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the Technical Committee’s web page at https://www.oasis-open.org/committees/ebcore/.

For information on whether any patents have been disclosed that may be essential to implementing this Work Product, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/ebcore/ipr.php).

Citation format:

When referencing this Work Product the following citation format should be used:
[ebcore-au-v1.0]
ebCore Agreement Update Specification Version 1.0. Edited by Pim van der Eijk and Theo Kramer. 18 September 2016. OASIS Committee Specification 01. http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/cs01/ebcore-au-v1.0-cs01.html. Latest version: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/ebcore-au-v1.0.html.

Notices

Table of Contents

1 Introduction

1.1 Overview

The ebCore Agreement Update specification defines message exchanges and an XML schema to support the exchange of agreement update requests with associated positive and negative responses to such requests. The main initial application of the specification is the exchange of X.509 certificates for certificate rollover, for which an extension is provided. The schema extensibility also supports potential other types of updates. The specification also provides an optional Agreement Termination feature. The specification is based on the concept of communication agreements and the creation of new configurations as independently identified updated copies of existing agreed configurations. The specification supports ebMS2, ebMS3 and AS4 but can in principle also be used with other protocols that have a concept of agreement. The specification is independent of storage or interchange formats for configuration information.

1.2 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All notes and examples in this specification, all appendices and sections 1.4, 2.5.2, 3.4.2, 3.6 and 4.5 are non-normative. All other text is normative.

1.3 Normative References

[AS4-Profile] *AS4 Profile of ebMS 3.0 Version 1.0*. OASIS Standard, 23 January 2013. Edited by J. Durand and P. van der Eijk. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/profiles/AS4-profile/v1.0/AS4-profile-v1.0.odt.

[ebCPPA] *Collaboration-Protocol Profile and Agreement Specification Version 2.0*, OASIS Standard, 23 September 2002. http://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf.

[ebMS2] *Message Service Specification*. Version 2.0. OASIS Standard, 1 April 2002. http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.

[EBMS3CORE] *OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features*. Edited by Pete Wenzel. OASIS Standard, 01 October 2007. Latest version: http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.html

[RFC2119] Bradner, S., “*Key words for use in RFCs to Indicate Requirement Levels*”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[XMLDSIG-CORE] *XML Signature Syntax and Processing (Second Edition)*, D. Eastlake, J. Reagle, D. Solo, F. Hirsch, T. Roessler, Editors, W3C Recommendation, June 10, 2008, http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/. Latest version available at http://www.w3.org/TR/xmldsig-core/.

[XMLDSIG-CORE1] *XML Signature Syntax and Processing Version 1.1*, D. Eastlake, J. Reagle, D. Solo, F. Hirsch, M. Nyström, T. Roessler, K. Yiu, Editors, W3C Recommendation, April 11, 2013, http://www.w3.org/TR/2013/REC-xmldsig-core1-20130411/. Latest version available at http://www.w3.org/TR/xmldsig-core1/.

1.4 Non-Normative References

[CEM]	*Certificate Exchange Messaging for EDIINT*. Edited by K. Meadors and D. Moberg. December 22, 2011, Expired June 18, 2012. Internet-Draft. https://tools.ietf.org/html/draft-meadors-certificate-exchange-14.
[CPPA3]	*Collaboration-Protocol Profile and Agreement Specification Version 3.0*. Edited by P. van der Eijk. OASIS Working Draft v0.5. https://www.oasis-open.org/committees/download.php/57550.

2 ebCore Agreement Update

2.1 Introduction

In B2B messaging, communication partners often need to share configuration data for their message service handlers. This information requires regular maintenance with changing messaging service requirements between partners. The ebCore Agreement Update specification addresses routine updates to configurations for existing communication partners by defining an exchange protocol and XML structures for the exchange of configuration updates. It does not address the separate task of configuring a new communication partner. The aim of the ebCore Agreement Update specification is to support automated or semi-automated configuration updates in order to lower operational cost and to reduce the risk of errors. It is expected that in common cases Agreement Update messages will be exchanged between the parties involved in the agreement that is being updated. However, this specification does not preclude use in other situations, such as situations involving third parties providing agreement management services.

The ebCore Agreement Update specification is based on the exchange of XML documents expressing update requests and responses and exceptions, which signal acceptance or rejection of the requests. These exchanges are specified in section 2. The XML structures used in ebCore Agreement Update exchanges are defined in the ebCore Agreement Update XML schema, referenced in section 2.5. This schema provides extensibility options to support common configuration updates other than certificate changes.

The main initial intended use of the specification is the exchange of certificates, using an extension specified in section 3. The XML structures used for certificate exchange are defined in the ebCore Certificate Update XML Schema, referenced in 3.4. Section 3.6 compares this specification to the expired Internet Engineering Task Force (IETF) Certificate Exchange Message (CEM) Internet Draft.

The Agreement Update and Certificate Update XML structures are payloads that are to be exchanged in messages. They can be produced and consumed by messaging products, either automatically or involving a manual service management approval workflow.

Analogously to Agreement Update, an Agreement Termination mechanism is provided and specified in section 2.4.

2.2 Agreements

The ebCore Agreement Update specification supports messaging protocols that use a concept of communication *agreements*. A communication agreement denotes a set of configuration parameters and parameter values used to control a particular exchange type, or sets of such sets controlling multiple exchange types. A communication agreement MUST have an identifier that MUST be unique in the context of the involved communication partners, and SHOULD be globally unique. At any particular point in time one or multiple agreements may be in place between parties exchanging messages.

An agreement is often valid in a particular interval. Specifically, the validity interval of agreements involving X.509 certificates is constrained by the validity period of the certificates involved.

Note that this concept of agreement only concerns the technical aspects of communication between parties. It is not to be confused with business agreements or contracts.

For various reasons, one or multiple parameter values associated with a particular agreement may cease to be valid. The ebCore Agreement Update specification allows partners to create new agreements based on existing agreements by proposing and confirming updates to these agreements. Updates may be confirmed (accepted) or rejected. Once a new agreement has been agreed, partners may stop using the agreement it is based on, or use the old and the new agreement in parallel.

This protocol is intended for situations where there is an existing communication service agreement which can be referred to using a mutually agreed identifier. The request references the existing agreement and specifies one or multiple specific changes that are to be applied to the existing agreement to create the new agreement. The creation of an initial agreement between two communication partners is out of scope for this specification.

Examples of B2B protocols supporting the concept of communication agreement are ebXML Messaging version 2.0 [ebMS2], 3.0 [EBMS3CORE] and AS4 [AS4]. Section 4 further specifies the use of ebCore Agreement Update with these protocols.

2.3 Agreement Update Message Exchange

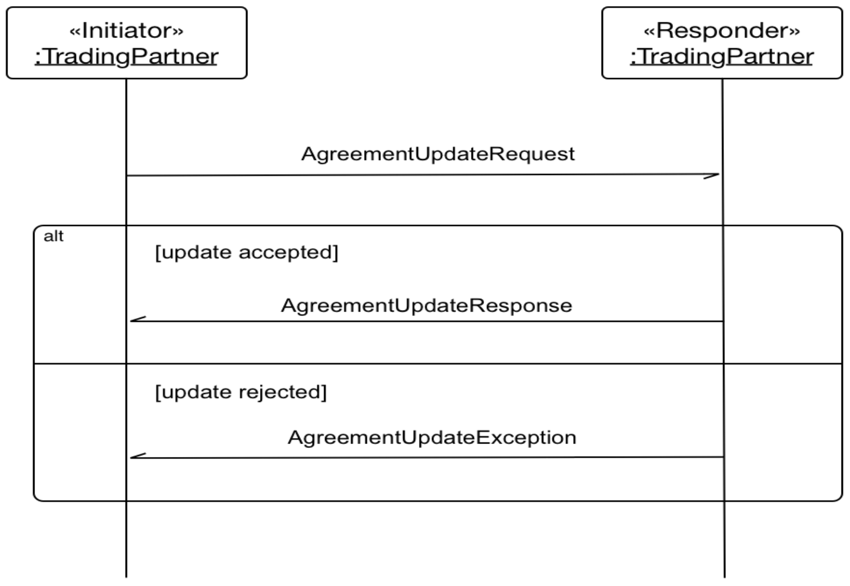
2.3.1 Overview

The Agreement Update protocol involves the exchange of XML documents expressing update requests, responses and exceptions defined in this specification:

- An initiator party can initiate an agreement update by sending an update request to a responding partner. The content of the agreement update request is described in section 2.3.2.
- If the responding party accepts the update request, it MUST notify the requester by sending an agreement update response. The content of the agreement update response is described in section 2.3.3.
- If the responding party does not accept the request, it MUST be rejected explicitly by sending an agreement update exception to the initiator. The content of the agreement update exception is described in section 2.3.4.

As the processing of update requests may involve business application processing and approval workflows involving manual steps, an agreement update exchange is typically an asynchronous process that may take hours or days.

The exchange is visualized in the following diagram:



If a request is rejected, either one of the parties MAY issue a new update request. This new request MAY either be accepted or rejected. If a request is accepted, a new agreement is established between the parties involved in the exchange. By accepting the new agreement, the old agreement is not automatically terminated. Parties MUST terminate the old agreement once they have successfully deployed the new agreement. This MAY be done using the Agreement Termination feature described in section 2.4.

The ebCore Agreement Update specification allows partners to set up, deploy and test new configurations before old configurations expire or are disabled. This allows communication to continue without disruption and reduces the risk that an error in an update results in breakdown of messaging service communication.

This section describes the exchanges in an abstract, protocol-neutral fashion. See section 4 for a specification of the exchange of Agreement Update documents using ebMS2, ebMS3 or AS4 messaging protocols. These message protocols MUST be configured so that the exchange of agreement update messages is secure. The use of message signing using X.509 certificates to authenticate the sender is RECOMMENDED. It is RECOMMENDED to configure reliable messaging to guarantee delivery of Agreement Update messages and to use timestamped Non-Repudiation Receipts.

The ebCore Agreement Update messages are independent of specific representation formats or storage mechanisms. If a party encodes configuration information for an agreement as an XML document based on the CPA 2.0 schema [ebCPPA], section 4.3 specifies how a new CPA 2.0 XML document can be created from the existing document and the update information specified in the request. Section 4.5 similarly describes the use of Agreement Update with a draft version of the CPPA 3.0 specification. However, this specification does NOT REQUIRE parties to use CPA or any other particular configuration format.

The XML schemas for AgreementUpdateRequest, AgreementUpdateResponse and AgreementUpdateException are defined in the ebCore Agreement Update XML schema, referenced in section 2.5.1. The documentation of the elements and types is included in the XML schema, which is a normative part of this specification. The documentation is also available in HTML format as described in section 2.5.2.

The Agreement Update protocol is independent of message protocols. Section 4 details the use of some message exchange protocols for the exchange of update requests.

2.3.2 Requesting an Agreement Update

To request a responder party B to update *existing_agreement* into *new_agreement*, initiator party A sends an AgreementUpdateRequest document to responder party B. As the UpdateRequest element is an abstract element, the initiator MUST use a non-abstract element that substitutes for the abstract element and that specializes Agreement Update to a specific situation. For example, the Certificate Update Request (see section 3) is a specialization of the Agreement Update Request to updating X.509 certificates.

When creating an Update Request, the initiating party A MUST ensure that the following requirements are met:

- The request MUST be valid syntactically.
- The value of the request ID MUST NOT have been used previously.
- The identifier *existing_agreement* MUST resolve to an existing agreement involving B under control of the Agreement Update service.
- Party A MUST be involved in *existing_agreement* or MUST be authorized to update *existing_agreement* involving party B. The evidence that this is the case is message protocol and configuration dependent, but is typically based on successful authentication (for example, by validating the message signature based on the X.509 certificate of the initiator party) and subsequent authorization of the Initiator.
- The message carrying the document SHOULD, if the message protocol supports this, use the configuration identified using the identifier *existing_agreement.* Specific requirements following from this request are message protocol-dependent. (For example, when using AS4, the value of

the `AgreementRef` header SHOULD match the value of CurrentAgreementIdentifier). This requirement does not apply to situations involving third parties.

- The request MUST be created and sent sufficiently in advance of the date and time at which the request is to be confirmed, to provide enough time to the counterparty to process the change request.
- The initiator MAY express the date and time by which it expect a response at the latest using the element RespondBy.
- The request MUST be created and sent sufficiently in advance of the date and time at which the request is to be implemented, to provide enough time to the counterparty to implement the change.
- The initiator MAY express the date and time by which it expects the change to be implemented at the latest using the element ActivateBy.
- The initiator MAY express, in natural language, the reason for the request using the element Reason.

After receiving the request, responder party B MUST check that these requirements are met. Furthermore, B MUST check the following additional constraints:

- The value for *new_agreement* is acceptable to party B. This may not be the case if party B is unable to interpret identifiers in the context of a particular communication partner and has an established agreement with a third party C that has the same value as the value proposed in the request.
- Party B is able to update the identified agreement using Agreement Update. (It may be that some agreements are under the control of the update service but that others are not).
- If specified, the value for ExpireBy MUST be acceptable to party B.
- If specified, Party B is able to reply to the request at the latest at RespondBy.
- If specified, Party B is able to activate the agreement at the latest at ActivateBy, if it will respond positively.

If one of these requirements is not met, the responder MUST reject the request without delay. Note that the use of a synchronous message exchange pattern is NOT REQUIRED.

An AgreementUpdateRequest may contain more than one UpdateRequest. The REQUIRED `id` attribute, in combination with the request ID, uniquely identifies a specific UpdateRequest.

An example of an update request document is provided in Appendix A.1 . An AgreementUpdateRequest includes one or more UpdateRequest elements. An UpdateRequest is an abstract element that has the abstract type UpdateRequestType.

2.3.3 Accepting an Agreement Update

To accept an Agreement Update request, responder party B MUST return an AgreementUpdateResponse document to initiator party A. This document:

- MUST be valid syntactically.
- The value of the response ID MUST NOT have been used previously.
- MUST be exchanged in a message that references the same agreement ID as the update request message. Specific requirements following from this requirement are message protocol-dependent. (For example, when using AS4, the value of the `AgreementRef` header in the response would need to match the value of the `AgreementRef` of the request message).

- MUST have the same values for CurrentAgreementIdentifier and UpdatedAgreementIdentifier as the corresponding values in the update request document.
- MUST use the ReferenceID set to the value of the ID of the request message.
- Responder party B MUST send the response document at the latest at RespondBy, if provided.

If A receives this response later than RespondBy, or if the response does not meet these requirements, party A MAY ignore the response. The Agreement Update protocol is limited to a pair of messages and does not provide messages to report on errors in the positive (or negative) response. It is RECOMMENDED that parties use other mechanisms (not specified in this specification) to handle such situations.

An AgreementUpdateResponse expresses that all UpdateRequests in the corresponding AgreementUpdateRequest document are accepted.

An example of a positive update response document is provided in section Appendix A.2 .

When a new agreement has been established between two parties for a particular exchange, sender and recipient MUST activate it at the latest at the date and time specified in the ActivateBy of the request, if provided.

If parties agree to use identifiers that are only guaranteed to be unique between two identified parties and that are not globally unique, then responder party B:

- MUST only update configurations with agreement *existing_agreement* for the initiator party A.
- MUST clearly restrict the updates associated with *new_agreement* to exchanges with initiator party A only.

Parties A and B MUST continue to support an updated previous agreement for incoming messages until there is appropriate evidence (successful exchange of a test message using the new agreement, followed by a switch of user messages from previous to new agreement) that the communication partner has successfully deployed the new agreement. Once parties are successfully using the new agreement, they MUST discontinue use of the old agreement. To coordinate the discontinuation of the old agreement, parties MAY use the Agreement Termination feature described in section 2.4.

2.3.4 Rejecting an Agreement Update

To reject an agreement update request, party B MUST return an AgreementUpdateException document to party A. This document:

- MUST be valid syntactically.
- The value of the response ID MUST NOT have been used previously.
- MUST be exchanged in a message that references the same agreement ID as the update request document. Specific requirements following from this requirement are message protocol-dependent. (For example, when using AS4, the value of the `AgreementRef` header in the response would need to match the value of the `AgreementRef` of the request message).
- MUST have the same values for CurrentAgreementIdentifier and UpdatedAgreementIdentifier as the corresponding values in the update request document.
- MUST use the ReferenceID set to the value of the ID of the request message.
- MUST contain Error elements for each error detected in the request. If an error relates to a specific UpdateRequest, the `refToUpdateInError` attribute MUST be set to the value of the corresponding `id` attribute.

If A receives this response later than RespondBy or ActivateBy, or if the response does not meet these requirements, party A MAY ignore the response. The Agreement Update protocol is limited to a pair of messages and does not provide messages to report on errors in the negative (or positive) response. It is

RECOMMENDED that parties use other mechanisms (not specified in this specification) to handle such situations.

An AgreementUpdateException expresses that none of the UpdateRequests in the corresponding AgreementUpdateRequest document are accepted.

An example of an agreement update exception response is provided in Appendix A.3 .

For the common failure situations mentioned in section 2.3.2, specific error codes to be used in the AgreementUpdateException document are defined. An `AgreementUpdateException` message contains one or more `Error` elements. The Agreement Update XML schema does not constrain the values of elements and attributes in `Error` elements. Instead, these values are defined in the following table which defines error codes to be used in these errors and for each code, a short description and an overview of its semantics. All these errors are failures.

Error Code	**Short Description**	**Severity**	**Description or Semantics**
AU:0001	InvalidMessage	Failure	The `AgreementUpdateRequest` is invalid. For example, it is not well-formed or not valid against the AgreementUpdate XML schema.
AU:0002	MessageRejected	Failure	The `AgreementUpdateRequest`is rejected for some other reason.
AU:0003	ProcessingError	Failure	An error occurred processing the `AgreementUpdateRequest`.
AU:0004	ServiceUnavailable	Failure	The request cannot be processed because the Agreement Update service is unavailable.
AU:0005	RespondByRejected	Failure	The requested `RespondBy` date and time is not accepted. For example, the recipient needs more time to process the request.
AU:0006	ActivateByRejected	Failure	The requested `ActivateBy` date and time is not accepted. For example, the recipient needs more time (after the request has been processed) to deploy the update.
AU:0007	ExpireByRejected	Failure	The requested `ExpireBy` date and time is not accepted.
AU:0008	UnknownAgreement	Failure	The `AgreementUpdateRequest` has a value for `CurrentAgreementIdentifier` that is unknown.
AU:0009	AgreementMismatch	Failure	The `AgreementUpdateRequest` has a value for `CurrentAgreementIdentifier` that is incompatible with the agreement in the protocol message carrying the request. (For example, the responder party may not accept update requests in AS4 messages with a value for `AgreementRef` different from the `CurrentAgreementIdentifier` in the change request.)
AU:0010	InvalidUpdatedAgreement	Failure	The `AgreementUpdateRequest`

Error Code	Short Description	Severity	Description or Semantics
			has a value for `UpdatedAgreementIdentifier` that cannot be accepted. (For example, it may be an identifier that already is in use).
AU:0011	AuthorizationFailed	Failure	The initiator of the request is not authorized to update the identified agreement.
AU:0012	ServiceNotApplicable	Failure	The identified agreement identifier exists but cannot be updated using Agreement Update.

Specific instances of Agreement Update may provide additional error types. For example, section 3.3.4 defines additional error types for Certificate Update.

Note that these errors relate to the Agreement Update service and are therefore not to be confused with message protocol errors (such as SOAP Faults or ebMS Errors) or other communication errors.

2.4 Agreement Termination Message Exchange

2.4.1 Overview

In addition to providing support for agreement updates, the ebCore Agreement Update specification also provides optional support for terminating an agreement. This feature supports the following situations:

- Party A and B have successfully created an updated agreement using Agreement Update. This update does not automatically terminate the existing agreement. The termination protocol and messages defined in this subsection allow parties A and B to agree on the phase out of the old agreement and on the date and time of that phase out.
- Separately from Agreement Update, it may be that some agreement is no longer needed and can be terminated. Agreement Termination allows parties to agree on the fact that an agreement is to be terminated and on the date and time by which this occurs.

The message exchange pattern of agreement termination mirrors the pattern of agreement update in providing three message types for requesting, accepting and rejecting termination.

Note that agreements may have an agreed date and time after which they are no longer valid. For updated agreements, this date and time can be agreed using the ExpireBy element. The termination message exchange is only needed to agree on a different termination date and time.

2.4.2 Requesting an Agreement Termination

To request a responder party B to terminate *existing_agreement*, initiator party A sends an AgreementTerminationRequest document to responder party B. An AgreementTerminationRequest document has a subset of the child elements of AgreementUpdateRequest, and adds a TerminateBy element. The activities to be performed and checks to be performed by Termination initiator and responder parties follow the description provided in section 2.3.2 for the message in general and for the child elements shared between the two request document types. In addition to this, responder party B MUST check that the value for TerminateBy, if specified, is acceptable to it.

An example of an update request document is provided in Appendix A.1

2.4.3 Accepting an Agreement Termination

To accept an Agreement Termination request, responder party B MUST return an AgreementTerminationResponse document to initiator party A. The activities to be performed follow the description provided for acceptance of an update in section 2.3.3.

2.4.4 Rejecting an Agreement Termination

To reject an Agreement Termination request, responder party B MUST return an AgreementTerminationException to initiator party A. The activities to be performed follow the description provided for rejection of an update in section 2.3.4. In addition to the error categories described in that section, the following error MAY be returned.

Error Code	**Short Description**	**Severity**	**Description or Semantics**
AT:0001	TerminateByRejected	Failure	The requested `TerminateBy` date and time is not accepted. For example, the recipient needs more time to process the request.

2.5 XML Schema

2.5.1 Normative Schema and Schema Documentation

The normative ebCore Agreement Update XML schema, which declares the **http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0** namespace, is specified in:

ebcore-au-v1.0.xsd

The documentation of this schema is embedded in this schema document and is a normative part of this specification.

For Agreement Update, the following XML elements are to be used as top-level elements in XML documents.

- AgreementUpdateRequest is the request document.
- AgreementUpdateResponse is the positive response document.
- AgreementUpdateException is the negative response document.

For Agreement Termination, the following XML elements are to be used as top-level elements in XML documents.

- AgreementTerminationRequest
- AgreementTerminationResponse
- AgreementTerminationException

2.5.2 Alternative Documentation Format (Non-Normative)

A non-normative export in HTML format of this embedded documentation is available at:

documentation/au.html

This includes documentation for the three Agreement Update and three Agreement Termination document types:

- AgreementUpdateRequest
- AgreementUpdateResponse

- AgreementUpdateException
- AgreementTerminationRequest
- AgreementTerminationResponse
- AgreementTerminationException

For convenience, this specification includes hyperlinks into the generated schema documentation.

2.5.3 Extensibility

The ebCore Agreement Update schema supports extensibility:

- The elements AgreementUpdateResponse and AgreementUpdateException optionally include element content from namespaces other than **http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0**.
- The UpdateRequest element is an abstract element that has the abstract type UpdateRequestType. This allows substitution by elements other than CertificateUpdateRequest.

A non-normative sample extension schema and document are provided in Appendix B Sample Extensibility (Non-Normative).

2.6 Test Service

When using protocols like ebMS2, ebMS3 and AS4 that provide a “ping” or “test” service feature, parties SHOULD first test the new agreement shortly after it is activated. If the test indicates a successful exchange, parties are ready to switch to the new agreement for outgoing non-test messages. The test service can be used in any environment, including production environments, as the receiving MSH is required to not deliver the test message to business applications.

See section 4 for information on using this specification with the Test Service feature provided by the ebMS2, ebMS3 and AS4 messaging protocols.

3 ebCore Certificate Update

3.1 Introduction

In messaging, X.509 certificates are often used for message or transport layer security: the signing certificate of the sender, the encryption certificate of the receiver and TLS client and server certificates. As certificates have a limited lifetime, they need to be updated before they expire. A key concern is to synchronize these updates:

- A party that has a new signing certificate needs to make sure any partners it sends messages to have adapted their configurations before sending new messages that use that certificate.
- A party that has a new encryption certificate needs to ensure other partners are successfully using it before disabling the old certificate.
- Parties may implement certificate rollovers at different times, meaning old certificates may need to be remain in place for some parties while other parties are already using an updated certificate.
- In very high frequency message exchanges, it may not be possible to define a date and time at which an MSH switches to a new certificate.

The ebCore Certificate Update protocol is an instance of the ebCore Agreement Update protocol. It enables parties to switch to using a new certificate by creating and agreeing on a new agreement that uses a new certificate, while still continuing to support (in a transition period) the existing agreement that uses the old certificate.

The ebCore Certificate Update protocol supports direct trust models based on the exchange and update of X.509 certificates. It does not impose any agreement on root or intermediate Certificate Authorities and also supports communication parties that use self-signed certificates. This is a requirement in various regions and industries. Recipients may also require proposed certificates to meet certain additional criteria, such as the requirement for certificates to be issued by a particular Certificate Authority, and reject update requests that propose new X.509 certificates that fail to meet these requirements.

For Certificate Update, an AgreementUpdateRequest MUST include a CertificateUpdateRequest (which is a substitution for the abstract UpdateRequest element) to request an update for a specific certificate (identified using CurrentCertificateIdentifier) used in a particular existing agreement. The Certificate Update extension elements are in the **http://docs.oasis-open.org/ebcore/ns/CertificateUpdate/v1.0** namespace.

If the update is accepted, a new agreement is established in which the new certificate is used in all contexts and for all purposes for which the current certificate is used.

3.2 X.509 Certificate Chains

A Certificate Update Request MAY include multiple `X509Certificate` elements in an `X509Data` element, representing a certificate chain from a leaf certificate to a root certificate, possibly via one or multiple intermediate certificates. In this situation:

- The leaf certificate is the certificate that is proposed to replace the certificate referenced by CurrentCertificateIdentifier.
- The certificate chain SHOULD be ordered, from the leaf certificate as first element, via any intermediate certificates, to a root CA as last element, such that each certificate C_{n+1} is the issuer of certificate $C_{n,}$.
- The provided root certificate and any intermediate certificates are provided only as context for the trust validation of the new certificate in the update request. The receiving party MAY require that

all non-leaf certificates are trusted security anchors for certificates of the type that the current certificate is used for.

- Accepting a certificate update request does not commit the accepting party to accept the included root and any intermediate certificates for any other particular purpose, or to add these certificates to any set of trust anchors they were not previously in.
- If one of the intermediate or root certificates is not trusted, the new proposed leaf certificate MUST NOT be trusted and therefore the update request MUST be rejected.

3.3 Certificate Update Message Exchange

3.3.1 Overview

The Certificate Update message exchange is an instance of the Agreement Update message exchange described in section 2.3.1. The structure of this subsection mirrors the structure of subsection 2.3.

3.3.2 Requesting a Certificate Update

Processing a CertificateUpdateRequest is a special case of processing UpdateRequests. The functionality specified in section 2.3.2 for all update requests applies. The values for ActivateBy and ExpireBy MUST be consistent with the validity interval of the proposed certificate.

A CertificateUpdateRequest:

- MUST reference an X.509 certificate used in the agreement that is to be updated using an `X509Digest` element, defined in the version 1.1 XML Signature specification [XMLDSIG-CORE1].
- MUST provide a new X.509 certificate as an XML Signature `KeyInfo` structure. This element is profiled for use in the CertificateUpdateRequestType.

The `KeyInfoType` type is defined in the XML Signature schema. In the Agreement Update specification, its use is profiled as follows:

- The `KeyName` and `KeyValue` elements MAY be present exactly once.
- The `RetrievalMethod` element MUST NOT be used.
- Exactly one `X509Data` element MUST be present.
- The `X509Data` element MUST include at least one `X509Certificate` element.
- A `X509Data` element MAY contain multiple `X509Certificate` elements.
- A `X509Data` element MAY contain one or multiple `dsig11:X509Digest` elements.
- If more than one `dsig11:X509Digest` element is present, each occurrence MUST have a different value for the `Algorithm` attribute.
- The element `X509IssuerSerial` MUST NOT be used.
- A `X509Data` element MAY contain at most one `X509SubjectName`.
- The elements `PGPData`, `MgmtData` and `SPKIData` MUST NOT be present.

A CertificateUpdateRequest MAY be rejected for the reasons described in section 2.3.4. Furthermore, it MAY be rejected for specific reasons related to the processing of certificates as described in section 3.3.4.

- The CU:0001 error indicates that the specified CurrentCertificateIdentifier is unknown.

- The CU:0002 error indicates that the proposed certificate is rejected.
- The CU:0003 error indicates that content in the `KeyInfo` element other than the `X509Certificate` is invalid or does not match the profiling defined in this section.

This certificate MAY be rejected for one or more of the following situations:

- A specified `X509Certificate` is not a valid X.509 certificate, or one or more of the certificate fields does not conform to a recipient's certificate policy for the use of the certificate in the agreement.
- The validity period of the certificate does not match the expected lifetime of the agreement. (For example, the certificate is valid but will expire before the agreement expires).
- The signature on the certificate is invalid.
- The certificate does not pass a CRL or OCSP check.
- One or more of the specified issuing intermediary or root certificates is unknown to, and/or not trusted by, recipient.

If supported by the messaging protocol, the updated agreement SHOULD be tested using the Test Service defined in section 3.5 before it is used to exchange regular messages to validate sender and recipient have both consistently deployed the new agreement.

Like any Agreement Update message, a CertificateUpdateRequest does not constrain how parties store certificates or how certificate configurations are managed.

3.3.3 Accepting a Certificate Update

Acceptance of a Certificate Update Request is an instance of the acceptance of an Agreement Update Request, as described in section 2.3.3, and involves the exchange of an AgreementUpdateException document.

3.3.4 Rejecting a Certificate Update

Rejection of a Certificate Update Request is an instance of the rejection of an Agreement Update Request, as described in section 2.3.4, and involves the exchange of an AgreementUpdateException document.

In addition to the errors described in section 2.3.4, an `AgreementUpdateException` message relating to a `CertificateUpdateRequest` MAY contain one or more `Error` elements using codes specific to certificate update processing, described in the following table.

Error Code	Short Description	Severity	Description or Semantics
CU:0001	UnknownCurrentCertificate	Failure	The `CertificateUpdateRequest` references a current certificate that is unknown, or unknown in the context of the current agreement
CU:0002	CertificateRejected	Failure	The proposed new certificate carried as `KeyInfo` element is rejected.
CU:0003	InvalidKeyInfo	Failure	The content of the `KeyInfo` element other than the `X509Certificate` is invalid or does not match the profiling defined in section 3.3.

Note that these errors relate to the Certificate Update service and are therefore not to be confused with message protocol errors (such as SOAP Faults or ebMS Errors) or other communication errors.

3.4 XML Schema

3.4.1 Normative Schema and Schema Documentation

The normative ebCore Certificate Update XML schema, which declares the **http://docs.oasis-open.org/ebcore/ns/CertificateUpdate/v1.0** namespace, is specified in:

ebcore-cu-v1.0.xsd

This schema defines a CertificateUpdateRequest element that is in the UpdateRequest substitution group.

The documentation of this schema is embedded in the schema and is a normative part of this specification.

This schema references the following schema:

xmldsig1-schema.xsd

The following namespaces are imported in the Certificate Update schema:

- **http://www.w3.org/2000/09/xmldsig#**
- **http://www.w3.org/2009/xmldsig11#**

3.4.2 Alternative Documentation Format (Non-Normative)

A non-normative export in HTML format of this embedded documentation is available at:

documentation/cu.html

For convenience, this specification includes hyperlinks into the generated schema documentation.

3.4.3 xmldsig1-schema.xsd

Schema ebcore-cu-v1.0.xsd imports xmldsig1-schema.xsd, which is an updated version of the XML Signature 1.1 Schema Driver file, published at https://www.w3.org/TR/2013/REC-xmldsig-core1-20130411/xmldsig1-schema.xsd. The update addresses known errata in the XML Signature 1.1 Schema Driver file.

3.5 Test Service for Certificate Update

When using Agreement Update to update certificates, the Test Service feature described in section 2.6 can be used to test the successful deployment of the certificate update.

- If an agreement specifies signed message exchanges, the positive or negative response from the receiver will reflect the successful or unsuccessful validation of the signature. Successful validation indicates consistent configuration of the signing certificate between sender and receiver.
- If an agreement specifies encrypted message exchanges, the positive or negative response from the receiver will reflect the successful or unsuccessful decryption of the message. Successful decryption indicates consistent configuration of the encryption certificate between sender and receiver.

In AS4 and other message protocols, encryption is only applied to the message payload and not to message headers. When used with a configuration that includes an encryption certificate, the test message MUST therefore carry some (arbitrary) encrypted payload so that payload encryption using the new certificate can be tested.

3.6 Relationship to IETF CEM (Non-Normative)

Other formats and protocols, including standardized formats and protocols, have been defined for certificate exchange and certificate updates. The ebCore Agreement Update specification, when used to exchange certificates, is most similar to the IETF Certificate Exchange Message [CEM] informational specification. The latest version of that specification is Draft 14, which expired in June 2012. Its editors have indicated that no further updates are planned. Compared with CEM, the ebCore Agreement Update protocol differs as follows:

- The ebCore Agreement Update schema is extensible using XML schema type and element substitution. While it currently only supports Certificate Exchange, it could be extended to support other types of updates.
- CEM is not based on the concepts of agreements. Instead, it requires message service handlers to support old and new certificates in parallel. The concept of agreements allows message service handlers to unambiguously select a specific agreement and agreement-specific certificates.
- CEM assumes the MIME-based packaging of AS1/AS2/AS3. The CEM specification defines an XML schema that references certificates that are exchanged in separate MIME parts. The ebCore Agreement Update schema exchanges all information in a single XML document payload. For the exchange of X.509 certificate data, it uses elements and types defined in the W3C XML Signature [XMLDSIG-CORE,]XMLDSIG-CORE1] recommendations.
- The CEM request specifies the `CertUsage` for a certificate. In the ebCore Agreement Update schema, a proposed new certificate is associated with the certificate it is to replace. This means that the new certificate will function for whatever usage the old certificate was used. The schema does not define or require a list of certificate usages.
- The CEM specification identifies certificates using `X509IssuerSerial`, which is deprecated in XML Signature 1.1. This specification uses the `dsig11:X509Digest` element introduced in XML Signature 1.1 instead.

4 Supported Message Exchange Standards

The Agreement Update protocol can be used with any messaging protocol that supports a concept of identified agreements and associated configuration parameters. Its main intended use is to support the ebXML Messaging Services specification version 2.0 [ebMS2] and 3.0 [EBMS3CORE] OASIS standards including the OASIS Standard AS4 profile of ebMS3 [AS4].

This section specifies the mapping of agreement identifiers in the Agreement Update protocol messages to ebMS2 (section 4.2) and ebMS3 (section 4.4) message protocol headers and to the OASIS ebXML Collaboration-Protocol Profile and Agreement Specification version 2.0 [ebCPPA] (section 4.3) or version 3.0 (see section 4.5). It also describes the use of the test service (see sections 2.6 and 3.5) for these protocols. Section 4.1 describes profiling common to all three of ebMS2, ebMS3 and AS4.

4.1 Profiling for ebXML Messaging

Any agreement between two parties A and B, that conform to this specification, MUST include support for the exchange of the three ebCore Agreement Update messages defined in section 2.3, in both directions. This allows each agreement to support use of this specification to exchange updates to itself.

The `Service` header for ebCore Agreement Update messages for a generic Agreement Update service SHOULD be set to the value:

- **http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0**

For Agreement Update, the `Action` header for this service SHOULD be set to the following values:

- **UpdateAgreement** for the AgreementUpdateRequest XML document
- **ConfirmAgreementUpdate** for the AgreementUpdateResponse XML document in response to a **UpdateAgreement** request.
- **RejectAgreementUpdate** AgreementUpdateException XML document in response to an **Up-dateAgreement** request

The `Service` header for ebCore Agreement Update messages for an Agreement Update service that is dedicated to Agreement Termination only (rather than to Agreement Updates in general) MAY be set to the value:

- **http://docs.oasis-open.org/ebcore/ns/AgreementTermination/v1.0**

For Agreement Termination, the `Action` header for this service SHOULD be set to the following values:

- **TerminateAgreement** for the AgreementTerminationRequest XML document.
- **ConfirmAgreementTermination** for the AgreementTerminationResponse XML document in response to a **TerminateAgreement** request.
- **RejectAgreementTermination** for the AgreementTerminationException XML document in response to a **TerminateAgreement** request.

The `eb:From/eb:Role` for the **UpdateAgreement** and **TerminateAgreement** messages and the `eb:To/eb:Role` for the **ConfirmAgreementUpdate**, **ConfirmTerminationUpdate**, **RejectAgreementUpdate**, **RejectTerminationUpdate** messages SHOULD be set to the value **http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator**. The `eb:To/eb:Role` for the **UpdateAgreement** and **TerminateAgreement** messages and the `eb:From/eb:Role` for the other four messages SHOULD be set to the value **http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder**. (Note: while these values are the default values from AS4, they are simply identifiers and can therefore be used with any version of ebXML Messaging, including the older version 2.0).

The `Service` header for ebCore Agreement Update messages for an Agreement Update service that is dedicated to Certificate Update only (rather than to Agreement Updates in general) MAY be set to the value:

- **http://docs.oasis-open.org/ebcore/ns/CertificateUpdate/v1.0**

The `Action` header for this service SHOULD be set to the following values:

- **UpdateCertificate** for the AgreementUpdateRequest XML document containing a CertificateUpdateRequest
- **ConfirmCertificateUpdate** for the AgreementUpdateResponse XML document in response to an **UpdateCertificate** request.
- **RejectCertificateUpdate** for the AgreementUpdateException XML document in response to an **UpdateCertificate** request.

Any exchange of messages relating to a particular `CurrentAgreementIdentifier` value SHOULD be exchanged in ebXML messages controlled by that particular agreement.

As the approval of an update request MAY require time consuming checks and even some human involvement (for examples, approval workflows), both positive and negative responses MUST be exchanged asynchronously.

The Agreement Update, Agreement Teremination and Certificate exchanges can be configured either:

- As nine separate One Way exchanges.
- As three Two Way exchanges, in which
  - the first leg is **UpdateAgreement** (or **TerminateAgreement** for Agreement Termination; or **UpdateCertificate** for Certificate Update).
  - and where **ConfirmAgreementUpdate** and **RejectAgreementUpdate** (or **ConfirmAgreementTermination**, **RejectAgreementTermination** for Agreement Termination; or **ConfirmCertificateUpdate, RejectCertificateUpdate** for Certificate Update) are alternatives for the second leg**.**

4.2 ebXML Messaging 2.0

In ebMS2, the agreement identifier maps to the REQUIRED `CPAId` element, which is defined to be “a string that identifies the parameters governing the exchange of messages between the parties. The recipient of a message MUST be able to resolve the `CPAId` to an individual set of parameters, taking into account the sender of the message” [ebMS2]. These parameters include security parameters, such as certificates to be used for the exchange. Therefore ebMS2 messages referencing different `CPAId` values are to be processed using the parameter sets associated with those values.

The ebMS2 Message Service Handler Ping Service defined in section 8 of [ebMS2] involves the exchange of a request message with a service **urn:oasis:names:tc:ebxml-msg:service** and action **Ping**, followed by a message with a **Pong** action if the request has been processed successfully using parameters from a specified `CPAId`. Each agreement between parties A and B that is updated using ebCore Agreement Update that supports the Ping Service MUST support exchange of **Ping** and **Pong** messages in both directions. This allows verification of correct deployment of the new agreement, including any certificates used in exchanges covered by it.

This exchange can be used by two parties to verify that they have successfully and consistently updated an agreement. As a successful **Ping** does not cause any business payload to be delivered to a business application, the Ping Service can be used in production environments.

4.3 ebXML Collaboration Protocol and Agreement 2.0

When using ebMS2 using version 2.0 ebXML Collaboration Protocol Agreements conforming to the OASIS ebXML Collaboration-Protocol Profile and Agreement Specification version 2.0 [ebCPPA], the value of the `CPAId` element MUST match the value of the `cpaid` attribute on the `CollaborationProtocolAgreement` element.

Implementing the Agreement Update protocol for use with ebMS2 MAY be implemented by copying the existing CPA that has a value for the `cpaid` attribute equal to the value of the `CurrentAgreementIdentifier`, changing this value of the `cpaid` attribute to the value of `UpdatedAgreementIdentifier`, applying the agreed changes to the copy and deploying the new CPA to the ebMS2 MSH.

4.4 ebXML Messaging 3.0

In ebMS3, the processing of messages is governed using contextual information called the Processing Mode (or P-Mode), which controls message structure and content, including headers values, and how the message is processed. Agreement Update can be used to update P-Mode configurations agreed between parties. An MSH is typically expected to use a number of P-Modes, reflecting different message types, processing and partner configurations. One of the P-Mode parameters is the **PMode.Agreement** parameter. The value of this parameter controls the value of the `AgreementRef` header in the ebMS3 header, if present. An MSH can use the value of the `AgreementRef` header in the selection of the P-Mode that applies to a message. In ebMS3, the `AgreementRef` header is an optional header. To be able to distinguish messages using Processing Modes with identical values for the other headers (such as `From` and `To` Party Identifiers and their identifier `type`, `Service` and `Action)`, but different values for the **PMode.Agreement** parameter, the `AgreementRef` header MUST be included in the ebMS3 message.

Note that ebCore Agreement Update does not provide an XML schema for ebMS3/AS4 processing modes. It is assumed that partners agree on an initial configuration using some other mechanism. The update protocol then allows them to update this configuration for common types of updates. Implementing the Agreement Update protocol for use with ebMS3 MAY be implemented by copying data structures representing the Processing Modes having a value for the **PMode.Agreement** parameter equal to the value of `CurrentAgreementIdentifier`, changing this value to the value of `UpdatedAgreementIdentifier`, applying the agreed changes to the copies and deploying the modified Processing Modes to the ebMS3 MSH.

Section 5.2.2 of the ebMS3 Core standard defines a `Service` similar to the ebMS2 **Ping** service, with value **http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/service** and `Action` value **http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/test**. Like the ebMS2 Ping service, it can be used to test successful update of an agreement if its Processing Mode references that agreement in its **Pmode.Agreement** parameter.

Each agreement that is updated using ebCore Agreement Update between parties A and B that support the ebMS3 test service MUST include exchange of ebMS3 test service messages. If there are regular (non-test) messages from A to B and/or from B to A, then it MUST be possible to send test messages from A to B and/or from B to A, respectively.

This message MUST be configured using the following PMode parameters and parameter value:

PMode[].Security.SendReceipt: true

If the Receiving MSH returns a Signal Message containing an `eb:Receipt` referencing the "test" User Message, the Sending MSH has evidence and that Receiving MSH has successfully and consistently deployed the new agreement.

If the Receiving MSH returns an ebMS error message or a SOAP Fault, the Sending MSH has evidence that the Receiving MSH has not successfully deployed the agreement.

4.5 ebXML Collaboration Protocol and Agreement 3.0 (Non-Normative)

The draft OASIS ebXML Collaboration-Protocol Profile and Agreement Specification version 3.0 [CPPA3] provides an XML schema for messaging service configurations.

Implementing the Agreement Update protocol with CPPA 3.0 is similar to the use with CPPA 2.0 described in section 4.3, except that:

- Instead of updating to the value of the `cpaid` attribute, references to agreement identifiers concern the `cppa:AgreementIdentifier` element in `cppa:AgreementInfo`.
- CPPA3 is not limited to ebMS 2.0 configurations, but also supports ebMS 3.0 and AS4, as well as the EDIINT AS1, AS2 and AS3 protocols. Use of Agreement Update with CPPA3 therefore supports messaging configurations for those protocols.

5 Conformance

An implementation can conform to the ebCore Agreement Update specification as an Initiator and/or as a Responder.
The Agreement Update protocol is an abstract protocol that depends on schema extensibility and specialization to specific applications. An implementation needs to support non-abstract specializations of the protocol, such as the Certificate Update protocol.

5.1 ebCore Agreement Update Conformance as an Initiator

An implementation conforms to the ebCore Agreement Update specification as an Initiator if it is able to create and send messages carrying ebCore Agreement Update Request XML documents that validate against the schema referenced in section 2.5 and to receive and process corresponding positive or negative response messages, implementing the transaction patterns described in section 2.3.

If the implementation uses ebMS2 as messaging protocol for agreement update messages, it MUST conform to the behavior in sections 4.1 and 4.2. If it uses ebCPPA version 2.0, in addition it MUST conform to section 4.3.

If the implementation uses ebMS3 or AS4 as messaging protocol for agreement update messages, it MUST conform to the behavior in sections 4.1 and 4.4.

5.2 ebCore Agreement Update Conformance as an Initiator for Termination

An implementation conforms to the ebCore Agreement Update specification as an Initiator for Termination if it is able to create and send messages carrying ebCore Agreement Termination Request XML documents that validate against the schema referenced in section 2.5 and to receive and process corresponding positive or negative response messages, implementing the transaction patterns described in section 2.4.

If the implementation uses ebMS2 as messaging protocol for agreement update messages, it MUST conform to the behavior in sections 4.1 and 4.2. If it uses ebCPPA version 2.0, in addition it MUST conform to section 4.3.

If the implementation uses ebMS3 or AS4 as messaging protocol for agreement update messages, it MUST conform to the behavior in sections 4.1 and 4.4.

5.3 ebCore Agreement Update Conformance as an Initiator for X.509 Certificate Updates

An implementation conforms to the ebCore Agreement Update specification as an Initiator for X.509 Certificate Updates if it:

- conforms to the ebCore Agreement Update Conformance as an Initiator clause defined in section 5.1.
- is able to create and send messages carrying ebCore Certificate Update Request XML documents that validate against the schema referenced in section 3.4 and to receive and process corresponding positive or negative response messages.
- supports the X.509 certificate exchange functionality defined in section 3.

5.4 ebCore Agreement Update Conformance as a Responder

An implementation conforms to the ebCore Agreement Update specification as a Responder if it is able to receive and process messages carrying ebCore Agreement Update Request XML documents, and create and send the corresponding positive or negative response messages, implementing the transaction patterns described in section 2.3.

If the implementation uses ebMS2 as messaging protocol for agreement update messages, it MUST conform to the behavior in sections 4.1 and 4.2. If it uses ebCPPA version 2.0, in addition it MUST conform to section 4.3.

5.5 ebCore Agreement Update Conformance as a Responder for Termination

An implementation conforms to the ebCore Agreement Update specification as a Responder for Termination if it is able to receive and process messages carrying ebCore Agreement Termination Request XML documents, and create and send the corresponding positive or negative response messages, implementing the transaction patterns described in section 2.4.

If the implementation uses ebMS2 as messaging protocol for agreement update messages, it MUST conform to the behavior in sections 4.1 and 4.2. If it uses ebCPPA version 2.0, in addition it MUST conform to section 4.3.

5.6 ebCore Agreement Update Conformance as a Responder for X.509 Certificate Updates

An implementation conforms to the ebCore Agreement Update specification as a Responder for X.509 Certificate Updates if:

- it conforms to the ebCore Agreement Update Conformance as a Responder clause defined in section 5.4.
- it is able to receive and process messages carrying ebCore Certificate Update Request XML documents that validate against the schema referenced in section 3.4, and create and return corresponding positive or negative response messages.
- it supports the X.509 certificate exchange functionality defined in section 3.

Appendix A Certificate Update Examples (Non-Normative)

This section provides examples of the three Agreement Update XML document types for certificate updates, and one example termination request. Base64 encoded data is shortened for readability.

Appendix A.1 Example Request

The following is a simplified example of an Agreement Update Request document used to update a certificate:

```
<?xml version="1.0" encoding="UTF-8"?>
<au:AgreementUpdateRequest
    xmlns:au="http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:dsig11="http://www.w3.org/2009/xmldsig11#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:cu="http://docs.oasis-open.org/ebcore/ns/CertificateUpdate/v1.0"
    xsi:schemaLocation="http://docs.oasis-open.org/ebcore/ns/CertificateUpdate/v1.0
            ../schema/ebcore-cu-v1.0.xsd">
    <au:ID>81ff47c6-84e7-4e09-8f6b-76f7807622d1</au:ID>
    <au:CreatedAt>2015-05-12T13:18:33.289487Z</au:CreatedAt>
    <au:RespondBy>2015-05-19T13:18:33.289487Z</au:RespondBy>
    <au:CurrentAgreementIdentifier>oldagreement</au:CurrentAgreementIdentifier>
    <au:UpdatedAgreementIdentifier>newagreement</au:UpdatedAgreementIdentifier>
    <cu:CertificateUpdateRequest id="ur1">
        <cu:CurrentCertificateIdentifier>
            <dsig11:X509Digest
                Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"
                >iR5cOjfMU2bOSyA8c+5owPGOFc6cK408fn/DbrKA==</dsig11:X509Digest>
        </cu:CurrentCertificateIdentifier>
        <ds:KeyInfo>
            <ds:KeyName>Examplecompany Signing Certificate</ds:KeyName>
            <ds:KeyValue>
                <ds:RSAKeyValue>
                    <ds:Modulus>2va9v7/1jNbMXZroYENaJS6ZBzpjK6rxhrtcQ==</ds:Modulus>
                    <ds:Exponent>AQAB</ds:Exponent>
                </ds:RSAKeyValue>
            </ds:KeyValue>
            <ds:X509Data>
                <dsig11:X509Digest
                    Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"
                    >5H0NE8XYXysP+DGNKHfuwvY7kxvUdBeoGlODJ6+SfaPg==</dsig11:X509Digest>
                <ds:X509SubjectName>CN=PartyIdentifier,OU=Transporter,
                      O=Examplecompany,C=AT</ds:X509SubjectName>
                <ds:X509Certificate>ZGl0IQgZW5jb2RlZCBzdHJpbmc=</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </cu:CertificateUpdateRequest>
</au:AgreementUpdateRequest>
```

The XML version of this document is available at:

samples/sample_request.xml

Appendix A.2 Example Positive Response

The following is an example of an Agreement Update Positive Response document:

```
<?xml version="1.0" encoding="UTF-8"?>
<au:AgreementUpdateResponse xmlns:au="http://docs.oasis-
open.org/ebcore/ns/AgreementUpdate/v1.0"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#" >
    <au:ID>bbfc1126-fe1d-4d5d-b5ca-5c222211471e</au:ID>
    <au:ReferenceID>81ff47c6-84e7-4e09-8f6b-76f7807622d1</au:ReferenceID>
    <au:CreatedAt>2015-05-12T15:22:11</au:CreatedAt>
    <au:CurrentAgreementIdentifier>existing_agreement</au:CurrentAgreementIdentifier>
    <au:UpdatedAgreementIdentifier>new_agreement</au:UpdatedAgreementIdentifier>
</au:AgreementUpdateResponse>
```

The XML version of this document is available at:

samples/sample_confirmation.xml

Appendix A.3 Example Negative Response

The following is an example of an Agreement Update Negative Response document:

```
<?xml version="1.0" encoding="UTF-8"?>
<au:AgreementUpdateException
    xmlns:au="http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 >
    <au:ID>587fe7a9-d831-4f1f-9a86-163a9b0af533</au:ID>
    <au:ReferenceID>81ff47c6-84e7-4e09-8f6b-76f7807622d1</au:ReferenceID>
    <au:CreatedAt>2015-05-12T15:22:11</au:CreatedAt>
    <au:CurrentAgreementIdentifier>existing_agreement</au:CurrentAgreementIdentifier>
    <au:UpdatedAgreementIdentifier>new_agreement</au:UpdatedAgreementIdentifier>
    <au:Error errorCode="AU:0006" severity="Failure"
        shortDescription="ActivateByRejected" refToUpdateInError="url">
        <au:Description xml:lang="en">Certificate updates require a minimum of 10
              working days</au:Description>
    </au:Error> </au:AgreementUpdateException>
```

The XML version of this document is available at:

samples/sample_exception.xml

Appendix A.4 Example Termination Request

The following is an example of an Agreement Termination request document:

```
<?xml version="1.0" encoding="UTF-8"?>
<au:AgreementTerminationRequest
    xmlns:au="http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0
       ../schema/ebcore-au-v1.0.xsd">
    <au:ID>09f6d9ea-d42c-4913-9251-14d32936ec18</au:ID>
    <au:CreatedAt>2016-02-11T19:11:04.862423Z</au:CreatedAt>
    <au:RespondBy>2016-02-18T19:11:04.862423Z</au:RespondBy>
    <au:TerminateBy>2016-02-29T23:59:59.000000Z</au:TerminateBy>
    <au:CurrentAgreementIdentifier>oldagreement</au:CurrentAgreementIdentifier>
</au:AgreementTerminationRequest>
```

The XML version of this document is available at:

samples/sample_termination_request.xml

Appendix B Sample Extensibility (Non-Normative)

This non-normative appendix illustrates ebCore Agreement Update extensibility.

Appendix B.1 Sample Schema

The following sample schema defines a `FirewallUpdateRequest` element and a `FirewallUpdateRequestType` type in the **http://namespaces.example.com/firewallconfig** namespace. The schema suggests a protocol for parties to exchange information on IP addresses that are to be allowed access in firewalls, or that can be denied access because they are no longer needed. Note that the schema only serves to illustrate extensibility.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    xmlns:fw="http://namespaces.example.com/firewallconfig"
    xmlns:au="http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0"
    targetNamespace="http://namespaces.example.com/firewallconfig">

    <xs:import namespace="http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/v1.0"
        schemaLocation="../schema/ebcore-au-v1.0.xsd" />

    <xs:annotation>
        <xs:documentation>
            <p>Sample schema that illustrates extensibility of the Agreement Update
schema.</p>
        </xs:documentation>
    </xs:annotation>

    <xs:element name="FirewallUpdateRequest" substitutionGroup="au:UpdateRequest"
type="fw:FirewallUpdateRequestType">
        <xs:annotation>
            <xs:documentation>
                <p>This element defines a firewall rule change request.</p>
            </xs:documentation>
        </xs:annotation>
    </xs:element>

    <xs:complexType name="FirewallUpdateRequestType">
        <xs:annotation>
            <xs:documentation>
                <p>A request can add or remove one or multiple IP addresses.</p>
            </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="au:UpdateRequestType">
                <xs:sequence>
                    <xs:element  name="AllowIP" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element  name="DenyIP" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

</xs:schema>
```

Appendix B.2 Sample Request

The following sample message based on the schema from Appendix B.1 shows the defined extension element `FirewallUpdateRequest` can be used as instance of the `UpdateRequest` in the same way as the `CertificateUpdateRequest` defined in the Certificate Update schema.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<au:AgreementUpdateRequest xmlns:au="http://docs.oasis-
open.org/ebcore/ns/AgreementUpdate/v1.0"
    xmlns:fw="http://namespaces.example.com/firewallconfig"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://namespaces.example.com/firewallconfig
firewallconfig.xsd">
    <au:ID>81ff47c6-84e7-4e09-8f6b-76f7807622d1</au:ID>
    <au:CreatedAt>2015-05-12T13:18:33.289487Z</au:CreatedAt>
    <au:RespondBy>2015-05-19T13:18:33.289487Z</au:RespondBy>
    <au:CurrentAgreementIdentifier>oldagreement</au:CurrentAgreementIdentifier>
    <au:UpdatedAgreementIdentifier>newagreement</au:UpdatedAgreementIdentifier>
    <fw:FirewallUpdateRequest id="fur1" >
        <fw:AllowIP>10.1.1.1</fw:AllowIP>
        <fw:DenyIP>10.1.1.2</fw:DenyIP>
    </fw:FirewallUpdateRequest>
</au:AgreementUpdateRequest>
```

Appendix C Acknowledgments (Non-Normative)

This specification was created in the OASIS ebCore Technical Committee, whose voting members at the time of writing included the following individuals:

Participants:

Albert Kappe, Justitiele Informatiedienst
Erlend Klakegg Bergheim, Difi-Agency for Public Management and eGovernment
Ernst Jan van Nigtevecht, Sonnenglanz Consulting
Farrukh Najmi, Individual
Kathryn Breininger, The Boeing Company
Pim van der Eijk, Sonnenglanz Consulting
Sander Fieten, Individual
Theo Kramer, Flame Computing Enterprises
Torsten Robert Kirschner, NAV

The editors also wish to thank members of the ITC Kernel Group of European Network of Transmission Operators for Gas (ENTSOG) for their input to and review of draft versions of this specification.

Appendix D Revision History (Non-Normative)

Revision	Date	Editor	Changes Made
WD01	2015-06-09	Pim van der Eijk	First version using the template by TC admin; based on Upload 55644.
WD02	2015-07-07	Theo Kramer	Review.
WD03	2015-08-22	Pim van der Eijk	New content covering draft CPPA 3.0. Editorial clean-up. Error codes and descriptions. Acknowledgments.
WD04	2015-08-27	Pim van der Eijk	Switch from deprecated X509IssuerSerial to the dsig11:X509Digest element. Feedback from users on restriction to leaf certificates and description of typical process steps added.
WD05	2015-08-31	Pim van der Eijk	Minor editorial cleanups Changed RespondByDate to RespondBy for consistency. Updated samples accordingly. Added error code to sample exception. Fixed URLs to generated documentation.
WD06	2015-09-09	Pim van der Eijk	Processed review comments from Ernst Jan van Nigtevecht. Added note that CPPA3 is in draft, and references to it are non-normative. In Abstract, sections 1.1, 2.3 and 3.3.2, explain that the AU messages do not constrain how parties manages configuration information. In the XML schema for AgreementUpdateRequest, remove the <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>. This caused ambiguity with schemas extending UpdateRequest. New appendix Appendix B to illustrate extensibility. Updates due to feedback from Sander Fieten: • Diagram added to show the message flows. • If identifiers are not universally unique, a recipient needs to create an agreement within a partner context, since another partner may have another agreement with the same identifier. Some care is needed to avoid overwriting an agreement with another partner (potential vulnerability). Updated 2.3.1 to state that if Party B needs universally unique identifiers, it can reject requests that would violate this. • Additional references to schema documentation. • Added a note that the protocol bindings must secure the exchange. • In 2.3.2, clarified that the first bullet is that the AS4 AgreementRef has to

			match the agreement in AU request message. The second is about the request and response. • Add a note to 4.1 that exchanges can be implemented as One Way or Two Way, if the latter, the RefToMessageId would be used for response. • The intention in W3C XML Signature seems to be that a certificate chain should be represented as different X509Certificates, not necessarily excluding other options. • State that RetrievalMethod must not be used because access to the referenced certificate may not be secure. • Clarified Certificates to be restricted to X.509 tokens, not the other token types that XML Signature supports. • In Error: Reference source not found, add note that the schema documentation profiles XML Signature's KeyInfo. • Separate Initiator and Responder Conformance.
WD07	2015-09-10	Pim van der Eijk	• Fixed an error in the conformance section. • Fixed inconsistency on requirement to de-activate old agreement (MUST or SHOULD resolved to MUST). • Many editorial improvements.
WD08	2015-09-15	Pim van der Eijk	• Fixed error in sample request. • Some fixes in 2.3.2 for situations involving third parties and for optionality of RespondBy, ActivateBy, ExpireBy. • Incomplete sentence in Appendix B.1 • Minor editorial improvements. • In schema, relaxed presence of KeyName and KeyValue to MAY.
WD09	2015-09-24	Pim van der Eijk	• A change to the schema. In the AgreementUpdateResponse, the ActivateBy and ExpireBy elements are not used. • Some trivial editorial updates.
CSD01 / PRD01	2015-10-20	TC Admin	Approved as CSD and published as PRD
WD10	2016-02-14	Pim van der Eijk	PR comments processed: • See comment disposition • Lots of editorial comments • Split AU and CU in spec and in schema • Also a separate namespace for CU • Error codes in respective sections. (Certificate Update has its own error codelist). • Added second defined namespace to front page • Fixed CPPA3 reference in related work on front page • Generated documentation is now in

			two separate HTML files. References from spec updated. • No documentation generated for imported XML Signature schemas. • Added a third CU error. • Errors and error codes now included in the *Rejecting ..* sections. • Added additional AU errors. • Moved the two sections on XML schema in AU/CU sections. • A bit more background on use of agreements for certificate exchange in section 3.1. • In 4.1, use the AS4 URIs for initiator and responder roles. • Renumbered the error codes. • Consistent formating for namespaces. • Added an `id` attribute to `UpdateRequest`, and a `refToUpdateInError` on `Errors` to correlate requests and updates. • New Agreement Termination feature. Related samples, schema, conformance clause etc.
WD11	2016-03-05	Pim van der Eijk	Review comments EJVN processed. • In section 2.1, added a general statement about third parties. • Other editorial clarifications / improvements. Other: • Editorial fixes. • Reference to and explanation for *xmldsig1-schema.xsd* added. • In the package, added updated *xmldsig1-schema.xsd* with embedded comments. • Added OASIS copyright notice to AU and CU schemas. • Updated CPA3 link to latest draft. • In ebMS bindings, Agreement Termination may be a separate `Service`. • New optional `Reason` in Update Request.
WD12	2016-04-11	Theo Kramer, Pim van der Eijk	All editorial comments from Theo Kramer approved (see his commented version in the attachment at https://lists.oasis-open.org/archives/ebcore/201604/msg00000.html). Some additional changes in response to his questions and comments. Preparing for CSPRD02.