ebCore Agreement Update Specification Version 1.0

Committee Specification Draft 01 / Public Review Draft 01

20 October 2015

Specification URIs

This 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

Previous version:

N/A

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:

- XML schemas: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/csprd01/schema/ebcore-au-v1.0.xsd
- Schema data dictionary: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/csprd01/documentation/
- XML document samples: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/csprd01/samples/

Related work:

This specification is related to:

- *Collaboration-Protocol Profile and Agreement Specification Version 2.0*. 23 September 2002. https://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf.
- *Collaboration Protocol Profile (CPP)*. https://www.oasis-open.org/committees/document.php?document_id=56327.
- *Message Service Specification Version 2.0*. 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*. Edited by Pete Wenzel. 01 October 2007. OASIS Standard. 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*. Edited by Jacques Durand and Pim van der Eijk. 23 January 2013. OASIS Standard. Latest version: http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/profiles/AS4-profile/v1.0/AS4-profile-v1.0.html.

Declared XML namespace:

- http://docs.oasis-open.org/ebcore/ns/AgreementUpdate/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 main initial application of the specification is the exchange of X.509 certificates for certificate rollover, but the schema offers extensibility for other types of updates. 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 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. 20 October 2015. OASIS Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/csprd01/ebcore-au-v1.0-csprd01.html. Latest version: http://docs.oasis-open.org/ebcore/ebcore-au/v1.0/ebcore-au-v1.0.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

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, but the schema offers extensibility for other types of updates. 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].

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/

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

[ebMS2] *Message Service Specification*. Version 2.0. 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*, 1 October 2007, OASIS Standard. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.pdf

[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] *XML Signature Syntax and Processing (Second Edition)*. W3C Recommendation 10 June 2008. http://www.w3.org/TR/xmldsig-core/

[XMLDSIG1] *XML Signature Syntax and Processing Version 1.1*. W3C Recommendation 11 April 2013. 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. Expired 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 v0.4. https://www.oasis-open.org/committees/document.php?document_id=56327

2 ebCore Agreement Update

2.1 Introduction

In B2B messaging, communication partners typically need to share configuration data for their message service handlers. This information requires regular maintenance with changing messaging service requirements between partners. This specification addresses the 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.

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 2.3. The main initial intended use of the specification is the exchange of certificates, defined in section 2.4. Section 2.5 is non-normative and compares this specification to the expired Internet Engineering Task Force (IETF) Certificate Exchange Message (CEM) Internet Draft.

The XML structures used in ebCore Agreement Update exchanges are defined in the ebCore Agreement Update XML schema, referenced in section 3. This schema provides extensibility options to support common configuration updates other than certificate changes. The XML structures are the payloads that are exchanged in messages. They can be produced and consumed by messaging products, either automatically or involving some service management approval workflow.

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 two communication partners, and SHOULD be universally unique. At any particular point in time one or multiple agreements MAY be in place between parties exchanging messages.

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 defines the use of ebCore Agreement Update with these protocols.

2.3 Agreement Update Message Exchange

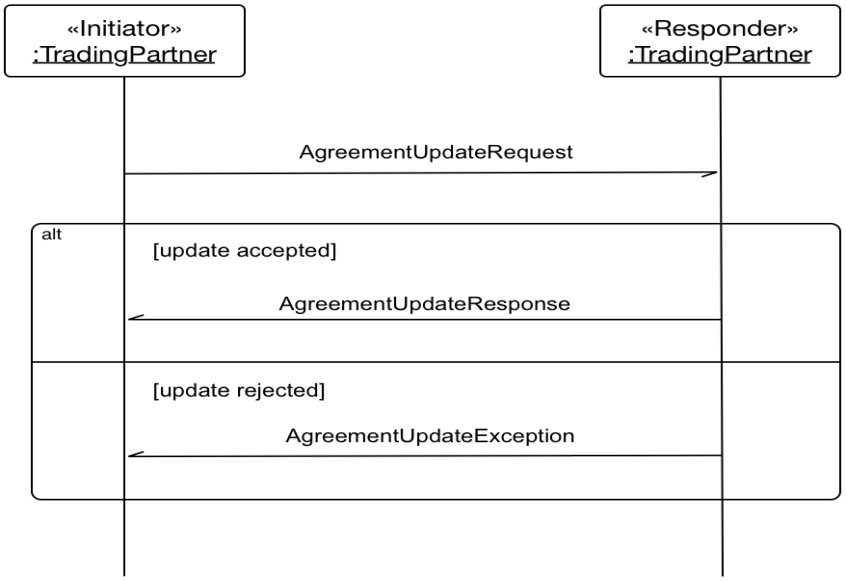
2.3.1 Overview

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

- A party can initiate an agreement update by sending an update request to a communication partner. The content of the agreement update request is described in section 2.3.2.
- If the other 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 other party does not accept the request, it MUST be rejected explicitly. 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, an agreement update exchange is typically an asynchronous process.

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.

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 these exchanges using ebMS2, ebMS3 or AS4 messaging protocols. These message protocols MUST be configured so that the exchange of agreement update messages is secure.

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. The non-normative section 4.5 similarly describes the use of Agreement Update with the draft 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 3. 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 3.2.

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 send a request to party B to update *existing_agreement* into *new_agreement*, party A sends an AgreementUpdateRequest document to party B. As the UpdateRequest element is an abstract element, the initiator MUST use a non-abstract element that substitutes the abstract element.

After receiving the request, recipient party B MUST check that the following requirements are met:

- The request MUST be valid syntactically.
- The identifier *existing_agreement* MUST resolve to an existing agreement.
- The request MUST be sent by a party that uses *existing_agreement* or by another party that party B knows to be authorized to update *existing_agreement*. The evidence that this is the case is message protocol and configuration dependent, but is typically based on successful verification of a message signature based on the X.509 certificate of the sender party.
- The message carrying the document SHOULD 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).
- The value for *new_agreement* MUST be acceptable to party B. This may fail to 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.
- If specified, the value for ExpireBy MUST be acceptable to party B.
- If specified, Party B MUST be able to reply to the request at the latest at RespondBy.
- If specified, Party B MUST be able to activate the agreement at the latest at ActivateBy, if it will respond positively.

If one of these requirements is not met, the recipient MUST reject the request without delay.

An example of an update request document is provided in section 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, party B MUST return an AgreementUpdateResponse document to party A. This document:

- 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.

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

Party B MUST send the response document at the latest at RespondBy, if provided.

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 universally unique, then party B:

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

An MSH MUST continue to support an updated previous agreement for incoming messages until there is appropriate evidence (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.

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 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.

For the common failure situations mentioned in section 2.3.2, specific error codes to be used in the AgreementUpdateException document are defined in section 3.3 using the errors codes AU:00** or AU:01** (where * is to be interpreted as any digit).

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

2.4 Using Agreement Update for X.509 Certificate Exchange

2.4.1 Overview

The ebCore Agreement Update specification 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, as is common in various regions and industries. However, recipients MAY require proposed certificates to meet certain requirements, including a requirement for those certificates to be issued by a particular Certificate Authority, and MUST reject update requests that propose new X.509 certificates that fail to meet these requirements.

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.

For Certificate Exchange, an AgreementUpdateRequest MUST include a CertificateUpdateRequest (which is a substitution for the abstract UpdateRequest element) to request an update for a specific certificate used in a particular existing agreement.

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.

2.4.2 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 certificates SHOULD be appended sequentially, from the leaf certificate as first element, via any intermediate certificates, to a root CA as last element.
- The provided root certificate and any intermediate certificates are provided only as context for the interpretation of 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. If this is the case, it MUST reject any requests involving root and/or intermediary certificates that are not trusted.
- 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.

2.4.3 Processing X.509 Certificate Update Requests

Processing a CertificateUpdateRequest is a special case of processing UpdateRequests. The functionality specified in section 2.3 for all update requests applies.

A CertificateUpdateRequest:

- MUST reference a known X.509 certificate using an X509Digest element, defined in the version 1.1 XML Signature specification [XMLDSIG1].
- MUST provide a new X.509 certificate as an XML Signature KeyInfo structure. This element is profiled as documented for the CertificateUpdateRequestType in the Agreement Update schema.

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 identified using the AU:02** error codes in section 3.3 (where * is to be interpreted as any digit).

- The AU:0201 error indicates that the specified CurrentCertificateIdentifier is unknown.
- The AU:0202 error indicates that the proposed certificate is rejected.

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 is not conform to 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 did 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 2.4.4 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.

2.4.4 Test Service

When using protocols like ebMS2, ebMS3 and AS4 that provide a “ping” or “test” service, an MSH SHOULD first test the new agreement shortly after it is activated. When used with a configuration that includes an encryption certificate, the test message SHOULD carry some (arbitrary) payload so that payload encryption using the new certificate can be tested.

If the test indicates a successful exchange, parties are ready to move to using the new agreement for outgoing non-test messages.

See section 4 for information on using this specification with the ebMS2, ebMS3 and AS4 messaging protocols.

2.5 Relationship to IETF CEM

The ebCore Agreement Update specification, when used to exchange certificates, is similar to the IETF Certificate Exchange Message [CEM] informational specification. The latest version of that specification is Draft 14, which expired in June 2012 and 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, XMLDSIG1] 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 the newly-introduced `dsig11:X509Digest` element instead.

3 Agreement Update XML Schema

3.1 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

This schema references the following schema:

xmldsig1-schema.xsd

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

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.

From the Agreement Update schema, the following namespaces are imported:

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

3.2 Alternative Documentation Format

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

documentation/index.html

This includes documentation for the three Agreement Update document types:

- AgreementUpdateRequest
- AgreementUpdateResponse
- AgreementUpdateException

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

3.3 Errors

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 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`.

Error Code	Short Description	Severity	Description or Semantics
AU:0101	RespondByRejected	Failure	The requested `RespondBy` date is not accepted. For example, the recipient needs more time to process the request.
AU:0102	ActivateByRejected	Failure	The requested `ActivateBy` date is not accepted. For example, the recipient needs more time to process the request and deploy the update.
AU:0102	ExpireByRejected	Failure	The requested `ExpireBy` date is not accepted.
AU:0103	UnknownAgreement	Failure	The `AgreementUpdateRequest` has a value for `CurrentAgreementIdentifier` that is unknown.
AU:0104	AgreementMismatch	Failure	The `AgreementUpdateRequest` has a value for `CurrentAgreementIdentifier` that differs from the agreement in the protocol message carrying the request.
AU:0105	InvalidUpdatedAgreement	Failure	The `AgreementUpdateRequest` has a value for `UpdatedAgreementIdentifier` that cannot be accepted. (For example, it may be an identifier that already is in use).
AU:0201	UnknownCurrentCertificate	Failure	The `CertificateUpdateRequest` references a current certificate that is unknown, or unknown in the context of the current agreement
AU:0202	CertificateRejected	Failure	The proposed new certificate carried as `KeyInfo` element is rejected.

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.

3.4 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 section Appendix B Sample Extensibility.

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 section 2.4.4) for these protocols. Section 4.1 describes profiling common to all three of ebMS2, ebMS3 and AS4.

Section 4.5 is non-normative, as the current version of the specification it references is a working draft.

4.1 Profiling for ebXML Messaging

Any agreement between two parties A and B 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 SHOULD be set to the value:

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

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

- **RequestAgreementUpdate** for the AgreementUpdateRequest XML document
- **ConfirmAgreementUpdate** for the AgreementUpdateResponse XML document
- **RejectAgreementUpdate** AgreementUpdateException XML document

The `eb:From/eb:Role` for the **RequestAgreementUpdate** message and the `eb:To/eb:Role` for the other two messages SHOULD be set to the value **Initiator**.

The `eb:To/eb:Role` for the **RequestAgreementUpdate** message and the `eb:From/eb:Role` for the other two messages SHOULD be set to the value **Responder**.

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 exchange can be configured either:

- As three separate One Way exchanges.
- As a Two Way exchange, in which the first leg is **RequestAgreementUpdate** and where **ConfirmAgreementUpdate** and **RejectAgreementUpdate** 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, 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 could 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

The 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.
In ebMS3, including the AS4 profile of ebMS3, the agreement identifier maps to the `AgreementRef` header, which "is a string value that identifies the agreement that governs the exchange. The P-Mode under which the MSH operates for this message should be aligned with this agreement. The value of an `AgreementRef` element MUST be unique within a namespace mutually agreed by the two parties." The value of `AgreementRef` element is configured using the **PMode.Agreement** parameter. In ebMS3, the `AgreementRef` is an optional element. To be able to distinguish messages using processing modes with identical values for the From and To Party Identifiers and identifier types, Service and Action, but different values for the **PMode.Agreement** parameter, the `AgreementRef` header MUST be included in the ebMS3 message.

Implementing the Agreement Update protocol for use with ebMS3 could involve copying 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 copy 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.AgreementRef** parameter.

Each agreement between parties A and B that is updated using ebCore Agreement Update that supports the ebMS3 test service MUST include exchange of ebMS3 test service messages in both directions. This message MUST be configured using the following PMode parameters and parameter values:

PMode[].Security.SendReceipt: true

PMode[].Security.SendReceipt.ReplyPattern: response

PMode[].ErrorHandling.Report.AsResponse: true

If the Receiving MSH returns a synchronous 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 consistently deployment the agreement.

4.5 ebXML Collaboration Protocol and Agreement 3.0

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 or as a Responder. A single implementation can also conform as both an Initiator and 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 X.509 Certificate Exchange protocol. Conformance for that protocol is defined in this section as well.

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 3 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 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, and furthermore supports the X.509 certificate exchange functionality defined in section 2.4.

5.3 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 to 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.
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.

5.4 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 5.3, and furthermore supports the X.509 certificate exchange functionality defined in section 2.4.

Appendix A Certificate Update Examples

This non-normative section provides examples of the three Agreement Update XML document types for certificate updates. 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#" >
    <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>
    <au:CertificateUpdateRequest>
        <au:CurrentCertificateIdentifier>
            <dsig11:X509Digest
                Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"
            >iR5cOjfMU2bOSyKgweFfbMy/Fgnowsh8vPzM33FK408fn/DbrKA==</dsig11:X509Digest>
        </au:CurrentCertificateIdentifier>
        <ds:KeyInfo>
            <ds:KeyName>Examplecompany Signing Certificate</ds:KeyName>
            <ds:KeyValue>
                <ds:RSAKeyValue>
                    <ds:Modulus>2va9v7/G/8YahjGMNmgZBzpjK6rxhrtcQ==</ds:Modulus>
                    <ds:Exponent>AQAB</ds:Exponent>
                </ds:RSAKeyValue>
            </ds:KeyValue>
            <ds:X509Data>
                <dsig11:X509Digest
                    Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"
                    >z4PhNX7vuL3xVChQ1m2ABkxvUdBeoGlODJ6+SfaPg==</dsig11:X509Digest>
                <ds:X509SubjectName>CN=PartyIdentifier,
                        OU=Transporter,O=Examplecompany,C=AT</ds:X509SubjectName>
                <ds:X509Certificate>MIIB9TCCeP1x/NryJXroM8tKw==</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </au: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:0102" severity="Failure">
        <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 B Sample Extensibility

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 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="../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 Agreement Update schema.

```
<?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>
        <fw:AllowIP>10.1.1.1</fw:AllowIP>
        <fw:DenyIP>10.1.1.2</fw:DenyIP>
    </fw:FirewallUpdateRequest>
</au:AgreementUpdateRequest>
```


Appendix C Acknowledgments

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

Participants:

Ernst Jan van Nigtevecht, Sonnenglanz Consulting BV
Farrukh Najmi, Individual
Kathryn Breininger, The Boeing Company
Pim van der Eijk, Sonnenglanz Consulting BV
Sander Fieten, Individual
Theo Kramer, Flame Computing Enterprises
Torsten Robert Kirschner, Individual

The editors also wish to thank the 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

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 2.4.3, 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 3.1, add note that the schema documentation profiles XML Signature's KeyInfo. • Separate Initiator and Responder Conformance.
WD07	2015-09-10	PvdE	• 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	PvdE	• 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	PvdE	• A change to the schema. In the AgreementUpdateResponse, the ActivateBy and ExpireBy elements are not used. • Some trivial editorial updates.