eDelivery AS4 - 1.13

Status	eDelivery Specification
Publication date	30.mai.2018
Obsoletes	**e-SENS AS4 - 1.12**
Obsoleted by	-

Table of Contents

1. Description

The eDelivery AS4 Profile is a modular profile of the ebMS3 and AS4 OASIS specifications. Its core is a mandatory Common Profile that selects, extends and profiles the AS4 ebHandler Conformance Profile and AS4 Additional Features and provides a common Usage Profile. This Common Profile can be implemented using open source or closed source AS4 software implementations. It is aligned with, and corresponds to a subset of, the AS4 profile for TSOs (Transmission System Operators) developed by ENTSOG (the European Network of Transmission System Operators for Gas).

In addition to the Common Profile, this specification provides a number of optional Profile Enhancement modules that specify functionality enhancements covering AS4 message exchange in four corner topologies, the use of AS4 in conjunction with the UN/CEFACT Standard Business Document Header (SBDH) specification, and Dynamic Receiver and Dynamic Sender behavior.

2. Base Specifications

This specification is based on the following OASIS specifications:

- [AS4] AS4 Profile of ebMS 3.0 Version 1.0. OASIS Standard, 23 January 2013. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/profiles/AS4-profile/v1.0/
- [EBMS3] OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features. OASIS Standard. 1 October 2007. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/os/

2.1. Features

The following table summarizes the features provided by the ebMS3 and AS4 specifications. The eDelivery profile is a profiled extended subset of the features of the AS4 specification.

Functionality	**ebMS 3.0 AS4**
Core Messaging	Web Services
Internet Transport	HTTP 1.1
Transport Layer Integrity, Sender Authentication, Receiver Authentication and Message Confidentiality (Non-Persistent)	Transport Layer (SSL / TLS) Security
Message and Payload Packaging	SOAP 1.2 with attachments
Routing and Dispatching, SOA integration	Mandatory "Service" and "Action" header elements
Exchange Patterns	One Way or Two Way (*)
Exchange Pattern Bindings	Push, Pull and Sync (*)
Payload Compression	Gzip (**)
Message Identification	ebMS 3.0 "MessageId"
Message Correlation	ebMS 3.0 "RefToMessageId" and "ConversationId"
Message Timestamp	ebMS 3.0 "Timestamp" and WS-Security "Timestamp"
Party Identification	ebMS 3.0 "From" and "To" party identifiers.
Non-Repudiation of Origin	WS-Security 1.1 using XML Signature
Message Confidentiality	WS-Security 1.1 using XML Encryption
Non-Repudiation of Receipt	Signed Receipt Signal Message
Reliable Message	AS4 reception awareness feature for lightweight, interoperable reliable messaging (**)

Table 1. ebMS3/AS4 Functional Overview. (*) in ebMS3, not in AS4 (**) AS4 extension to ebMS3

2.2. Benefits

The AS4 technical specification [AS4] defines a secure and reliable messaging protocol. It can be used for message exchange in Business-to-Business (B2B), Administration-to-Administration (A2A), Administration-to-Business (A2B) and Business-to-Administration (B2A) contexts.

AS4 messages can carry any number of payloads. Payloads may be structured or unstructured documents or data.

Message packaging provided by AS4 as an add-on feature relies on ebMS 3.0 [EBMS3] support for the SOAP 1.2 specification [SOAP12]. AS4 combines the traditional functional support of payload compression in line with ebMS 3.0 message packaging norms. Compression in AS4, if used, must be applied prior to the application of any message-level security such as digital signing or encryption. AS4 does not define a maximum message size, though implementations will have practical limits based on available memory, disk or database storage etc.

AS4 offers a secure exchange protocol for use over the Internet that leverages the MIME envelope structure to transport arbitrary payloads. Support for Message Security is provided by AS4 via ebMS 3.0 and the WS-Security 1.1 and 1.1.1. specifications. This includes combinations of XML Digital Signature and XML Encryption X.509 security tokens for signing and encrypting as primary means for authenticating messages, ensuring privacy, and guaranteeing safe data transmission. Additionally, AS4 supports the use username/password tokens as access control to message pull channels.

The ebMS 3.0 and AS4 specifications provide support for Non-Repudiation of Receipt (NRR) by using a Signed Receipt Signal Message. The receipt is returned using a special signal message. Signal messages may also contain error handling information if there was some problem with the exchange.

AS4 makes use of the message receipt as a signal to the original message sender that the business payload (or payloads) has (have) been received. AS4 supports duplicate message detection and message retry/resending scenarios for when receipts for messages are not received by the sender.

Other technical highlights are:

- Payload agnosticism: document or data types (e.g. purchase order, invoice, XML, PDF etc.) are not tied to any defined SOAP action or operation;
- Support for single or multiple payloads contained either within the SOAP body or as SOAP (MIME) attachment(s);
- Support for the ebMS 3.0 One-Way/Push message exchange pattern with support for either synchronous or asynchronous signal responses;
- Support for the ebMS 3.0 One-Way/Pull message exchange pattern, which is beneficial for exchanges with non-addressable endpoints;
- Reception Awareness features and Duplicate Detection capabilities make use of the eb:Receipt as the sole type of acknowledgment.

Note that this version of this eDelivery AS4 specification does not use all features of ebMS3 and AS4. Specifically, it does not use the *Pull* pattern. Future versions of this profile may require additional support for *Pull* or other additional features not currently included. For a complete feature list, see section 3.2.

2.3. Notation

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

As in the ebMS3 Core Specification [EBMS3], this specification uses the term *P-Mode* to refer to an ebMS3 Processing Mode and **PMode** in P-Mode parameter labels. Furthermore, this specification uses the notation **PMode[]** to generalize over parameters that apply to the user message in the single leg in a One Way exchange, i.e. **PMode[1]**, or that are specified separately for the user messages in the two legs in a Two Way exchange, i.e. **PMode[1]** and **PMode[2]**. In section 4.4.2, there is a need for a parameter to configure the X.509 certificate used to sign the receipt or error signal message in a leg. This is done using the **PMode[][s]** sequence in the **PMode[][s].Security.X509.Signature.Certificate** parameter, following the pattern introduced in section D.2.1 of the ebMS3 Core Specification [EBMS3].

3. Common Profile

The eDelivery AS4 Common Profile profiles the OASIS ebMS3 and AS4 specifications. The eDelivery AS4 Common Profile selects the AS4 ebHandler Conformance Profile, a specific conformance profile defined in the AS4 specification [AS4], and makes a selection of AS4 Advanced Features. The selected Conformance Profile and selected Advanced Features are profiled further for increased consistency, ease of configuration and up to date security. Finally, this Common Profile provides a common AS4 Usage Profile. An AS4 Usage Profile defines how to use a conformant implementation for general exchange of any arbitrary payload combinations. This eDelivery AS4 Common Profile explains that some features available in the AS4 ebHandler Profile are not used (such as Pull exchange pattern bindings and WS-Security Username token authentication), whereas other optional features (TLS, XML Signature and XML Encryption) are mandatory and used with

specific algorithms. The Common Profile extends AS4 by using more up to date versions of some underlying specifications and mandating support for the Two-Way MEP.

This eDelivery Common Profile is based on the ENTSOG AS4 profile for TSOs (Transmission System Operators). ENTSOG, the European Network of Transmission System Operators for Gas, developed ENTSOG AS4 as a secure interoperability profile for AS4 [ENTSOGAS4]. This profile incorporates state-of-the-art security guidelines and has been reviewed by experts from the European Union Agency for Network and Information Security (ENISA). This eDelivery AS4 Common Profile is the same as the ENTSOG profile, for general AS4 messaging aspects.

3.1. AS4 and Conformance Profiles

The eDelivery AS4 profile is based on the AS4 Profile of the ebMS 3.0 Version 1.0 OASIS specification [AS4]. AS4 is based on other specifications, in particular on OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features OASIS specification [EBMS3], which in turn is based on various Web Services specifications. The OASIS Technical Committee responsible for maintaining the AS4, ebMS 3.0 Core and other related specifications is tracking and resolving issues in the specifications [EBERRATA]. These resolutions will eventually be published as a consolidated Specification Errata but should already be taken into account by implementers, to avoid functional or interoperability issues.

The AS4 specification defines multiple conformance profiles, which define specific functional subsets of the version 3.0 ebXML Messaging, Core Specification. A conformance profile corresponds to a class of conformant applications. This Common Profile of AS4 is based on an extended subset of the **AS4 ebHandler Conformance Profile**, a selection of AS4 Advanced Features and a Usage Profile. It supports point-to-point communication from Senders to Receivers using eDelivery Access Points using the ebMS3 “Push” transport channel binding. The optional Four Corner Topology feature extends this to interconnection of messaging infrastructures.

By using “Push”, messages that are submitted by a Producer to a Sending ebMS3 Message Service Handler (MSH) are transmitted to the Receiving ebMS3 MSH immediately, without the (unpredictable) delay of a “Pull” transport channel binding. Assuming the latency of the transmission of the request message from the Receiving MSH to the Consumer, the business processing of the request message content by the Consumer and the reverse flow of the response message from the Consumer back to the Producer, using their Message Service Handlers in reverse sequence, can be minimized, this profile can also support time-critical business processes that need “interactive” responses.

3.2. eDelivery AS4 ebHandler Feature Set

The feature set of the eDelivery AS4 Common Profile is, with some exceptions, a subset of the feature set of the AS4 ebHandler Conformance Profile specified in section 2.1.1 of the AS4 specification [AS4]. This section selects specific options in situations where the AS4 ebHandler provides more than one option to choose from. This section can therefore be used as a checklist of features to be provided in AS4 implementations. The structure of this section mirrors the structure of the ebMS3 Core Specification [EBMS3]. The ebMS 3.0 protocol can support synchronous as well as asynchronous communication and provides full convergence with Web Services. It reuses the SOAP 1.2, WS-Security 1.1, and SOAP-with-attachments specifications. It conforms to the WS-I Basic Profile (BP) and Basic Security Profile (BSP) and provides additional features of particular relevance to small and medium-size enterprise, in particular message pulling. AS4 drops the requirement for synchronous exchange of user messages but the AS4 ebHandler Conformance Profile allows synchronous and asynchronous signal messages.

Compared to the AS4 ebHandler Conformance Profile, this eDelivery Common Profile profile updates, and adds, some functionality:

- There is an added requirement to support Two Way MEPs.
- Transport Layer Security, if handled in the AS4 handler, is profiled and its use is mandatory.
- WS-Security is mandatory and the WS-Security specification version is the 1.1.1 version.
- Algorithms specified for securing messages at the Message Layer are updated to current guidelines and use of signing and encryption is mandatory using specific algorithms.
- Support for IPv4 and IPv6 is explicitly required.

It also relaxes some requirements:

- Support for **Pull** mode in AS4 is not required in this version.
- All payloads are exchanged in separate MIME parts, never in the SOAP Body.
- Receipts and errors are reported synchronously only.
- WS-Security support is limited to the X.509 Token Profile. The use of *UserName* Tokens is not supported.

3.2.1. Message Exchange Patterns

The following paragraphs summarize some key concepts and terminology defined in the ebMS 3.0 core specification [EBMS3]:

1. **Messaging Service Handler (MSH), Producer, Consumer.** An *MSH* is an entity that is able to generate or process messages that conform to the ebMS specification, and to act as sender or receiver. A *Producer* is an entity (e.g. application) that interacts with a Sending MSH (i.e. an MSH in the Sending role) to initiate the sending of a user message. A *Consumer* is an entity that interacts with a Receiving MSH (i.e. an MSH in the Receiving role) to consume data from a received user message.
2. **Message, User Message, Signal Message.** A *Message* is a logical unit which consists of User Messages or Signal Messages or both. A *User Message* is a message that contains a User Message unit (an **eb:Messaging/eb:UserMessage** XML structure). A *Signal Message* is an ebMS message that contains a Signal Message unit (an **eb:Messaging/eb:SignalMessage** XML structure). In other words, there exist two types of messages in the ebMS specification: the first type allows transmitting data interpreted by a Consumer and the second type allows transmitting data interpreted by an MSH as a signal (e.g. a pull signal, receipt or error).
3. **Message Exchange Pattern (MEP)**, **One-Way/Push, One-Way/Pull, Two-Way/Sync MEP**. An *MEP* is an agreement between sending and receiving MSHs. Some aspects of MEPs supported in the messaging layer include:
   - Specifying the correlation between messages sent and received in the message header.
   - Message binding to the underlying transfer-protocol.

   One-Way/Push, One-Way/Pull and *Two-Way/Sync MEPs* describe agreements between MSHs.
4. **Processing Mode (P-Mode)** - A P-Mode is the contextual information that governs the processing of a particular message (thus is basically a set of configuration parameters). The P-Mode associated with a message determines, among other things, which security and/or which reliability protocol and parameters, as well as which MEP is being used when sending a message. The technical representation of the P-Mode configuration is implementation-dependent.

The **Messaging Model** of the AS4 profile constrains the channel bindings of message exchanges between two AS4 MSHs. The following diagram shows the AS4 Messaging Model, various actors and operations in message exchange:

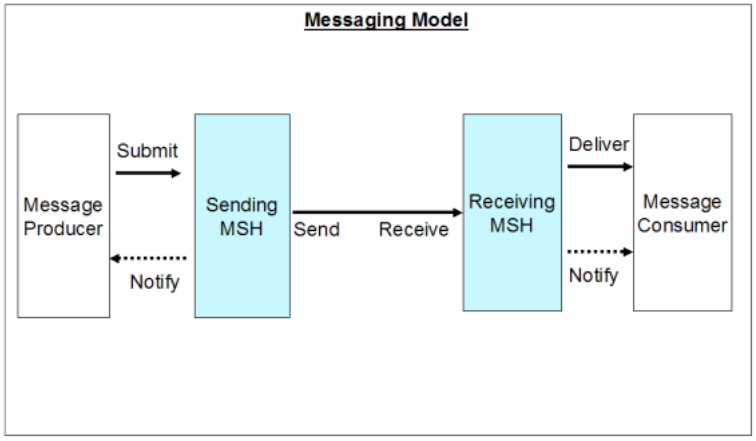

Figure 1. Entities of the AS4 Messaging Model and their Interactions [EBMS3].

Business applications or middleware, acting as *Producer*, *Submit* message content and metadata to the Sending MSH, which packages this content and *Send*s it to the Receiving MSH of the business partner, which *Receives* it and in turn *Delivers* the message to another business application or middleware that *Consumes* the message. Subject to configuration, Sending and Receiving MSH may *Notify Producer* or *Consumer* of particular events. Note that there is a difference between *Sender* and *Initiator*. For **Push** exchanges, the Sending MSH initiates the transmission of the message. For **Pull** exchanges (not supported in the eDelivery Common Profile), the transmission is initiated by the Receiving MSH. Also note that a business application can include MSH functionality, leaving the MSH as an abstract concept.

The concepts of Producer, Sender, Receiver and Consumer are separate from, and are not to be confused with, the corners in the Four Corner Topology concept (see section 4.1). While separate components, the Message Producer and Sending MSH entities MAY be enterprise-internal components only and MAY not be associated with different communication, business or legal identities. Sending and Receiving MSH are messaging endpoints, not intermediaries.

Different Message Producers MAY share a single Sending MSH and a single Receiving MSH MAY serve different Message Consumers and provide message routing functionality, using message metadata or payload content as routing delivery criteria. For example, different

values of the AS4 **eb:Service** header could be used to select a specific Consumer. Specification of this functionality is out of scope for this specification.

A single MSH is said to be multi-tenant if it MAY be configured to send messages on behalf of multiple sender entities or to receive messages on behalf of multiple receiver entities using distinct identities, each possibly associated with distinct processing mode configurations including different party identifiers, certificates and endpoint URLs. Implementation and configuration of this feature is implementation-dependent.

The AS4 ebHandler Conformance Profile is the AS4 conformance profile that provides support for Sending and Receiving roles using **Push** channel bindings. Support is required for the following Message Exchange Patterns:

- *One Way / Push*
- *Two Way / Push-and-Push*

In the context of ebMS message exchanges, *pushing* means that the sender initiates the message exchange. For HTTP, this implies that the Sending MSH is acting as an HTTP client, and the Receiving MSH as an HTTP server. *Pulling* means that the receiver initiates the message exchange. For HTTP, in that situation the Receiver MSH is an HTTP client and the Sending MSH is an HTTP server.

The *One-Way/Push MEP* specifies a situation when a Sending *MSH* which has agreed to use the *One-Way/Push MEP* sends a message to a Receiving *MSH* which has agreed to use *One-Way/Push MEP* as well. After the successful reception of the message, the receiving *MSH* returns a non-user message (i.e. a signal message) to the sending *MSH* to confirm the reception. Different user messages do not have any reference to each other, except possibly if they are part of the same conversation.

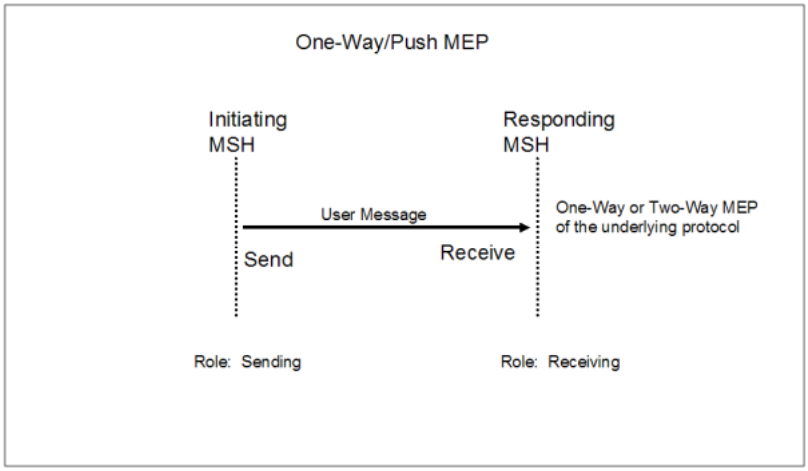

Figure 2. One-Way/Push MEP [EBMS3].

While the AS4 ebHandler does not require support for the Two-Way MEP, support for this MEP is REQUIRED in this eDelivery Common Profile. It can be used for business processes involving correlated pairs of request and response messages. A message handler that supports Two Way MEPs allows the Producer submitting a response user message unit to set the optional **eb:RefToMessageId** element in the **eb:MessageInfo** section to identify the corresponding request user message unit.

For **PMode.MEP**, support is therefore required for the following values:

- http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/oneWay
- http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/twoWay

For **PMode.MEPbinding,** support is required for:

- http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/push
- http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pushAndPush

Note that these URI values are identifiers only, which are defined in section 2.2.8 of the ebMS3 specification. They do not resolve to content on the OASIS site.

Time-critical processes require the *Push* channel binding, because it allows the *Sender* to control the timing of transmission of the message. Interactive, request-response communication between parties A and B can be provided as a combination of two *Push* messages, one from the Sending MSH, on behalf of A, to the Receiving MSH, which in turn acts as Receiver for B, followed by a separate (asynchronous) response message from B using its MSH to A's MSH.

The *Two-Way/Sync MEP* specifies a situation in which the Sending MSH and the Receiving MSH use a single HTTPS connection to exchange both a request message and the corresponding response message. The Sending MSH is the Initiating MSH, which acts as an HTTPS client and sends the request message. The Receiving MSH is the Responding MSH, acting as an HTTPS server. It receives the request message and returns a response message on the HTTPS backchannel. The *Sync* transport channel binding is not part of the OASIS AS4 profile and not currently part of this eDelivery AS4 Common Profile.

The *Two-Way/Push-and-Push MEP* is very similar to a sequence of two *One-Way/Push* exchanges, in which the *Sender* and *Receiver* roles are reversed. The Responding MSH, which received the request User Message in the first exchange, becomes the Sending MSH in a second, separate exchange in which the business response User Message is transmitted. This second exchange is separately initiated and, unlike the *Sync* channel binding, does not depend on any connection timeout intervals of the underlying HTTP transport. The *Two-Way/Push-and-Push MEP* MUST be supported in conformant eDelivery AS4 implementations.

In any Two Way Message Exchange, the response User Message MUST have a **eb:RefToMessageId** element with the value set to the value of the **eb:MessageId** in the corresponding request message.

3.2.2. AS4 Message Structure and UserMessage

The AS4 **Message Structure** provides a standard message header that addresses common data exchange requirements and offers a flexible packaging mechanism based on SOAP and MIME enveloping. Dashed line style is used for optional message components.

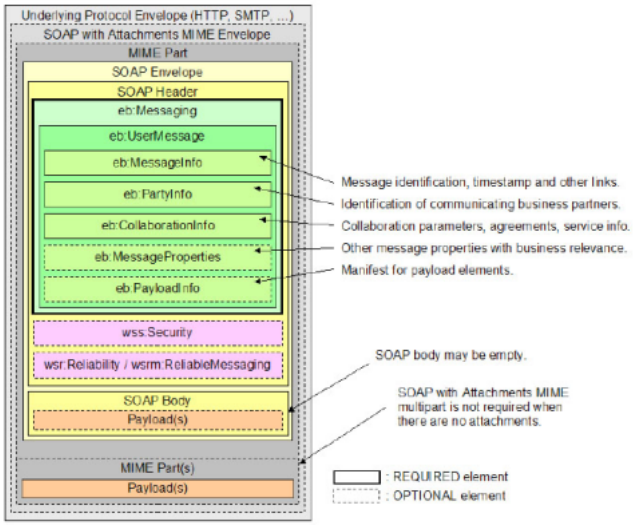

Figure 3. AS4 Message Structure, UserMessage.

The SOAP envelope SHOULD be encoded as UTF-8 (see [EBMS3], section 5.1.2.5). If the SOAP envelope is correctly encoded in UTF-8 and the character set header is set to UTF-8, receivers MUST support the presence of the Unicode Byte Order Mark (BOM; see [BP20], section 3.1.2).

AS4 defines the ebMS3 **eb:Messaging** SOAP header, which in user messages is an envelope for **eb:UserMessage** XML structures, which provide business metadata to exchange payloads. In AS4, ebMS3 messages other than receipts or errors carry a single **eb:UserMessage**.

A conformant implementation, acting as Sending MSH, MUST allow the Producer, when submitting a user message, to:

- set the value for **eb:ConversationId**. This enables the Consumer to correlate the user message to related user messages that are part of the same conversation.
- set the value for **eb:RefToMessageId**, for business response messages in a Two WAY MEP. This allows the Consumer to indicate to which previous AS4 request user message the user message is a response. Note that a shared value for **eb:ConversationId** is not sufficient for correlating requests and responses as there may be more than one outstanding request in a single conversation.

A conformant implementation, acting as Receiving MSH, MUST provide the values of **eb:MessageId**, **eb:ConversationId** and, if available, **eb:RefToMessageId**, as metadata with any user message it delivers to the Consumer.

To be able to relate a business response message to a previous business request, the Producer MUST know the **eb:MessageId** of this request message. The implementation of this requirement is left to implementations, but MAY be one of the following:

- The MSH allows the Producer to specify, when submitting the user message unit, a specific value to be used for **eb:MessageId**.
- The MSH generates the **eb:MessageId** value, and then notifies the Producer of the generated value.

The ebMS3 and AS4 specifications do not constrain the value of **eb:MessageId** beyond requiring conformance to the Internet Message Format [RFC2822], which requires the value to be unique and use the *msg-id* format defined in section 3.6.4 of [RFC2822]. It is RECOMMENDED that the value include a universally unique identifier in the *id-left* part of the *msg-id* identifier.

Conformant implementations MUST be able to configure the presence and value of the **eb:AgreementRef** header content and its **type** attribute by configuring the applicable **PMode.Agreement** parameter.

A conformant implementation MUST allow the Producer, when submitting messages, to set a value for **eb:AgreementRef**, and to use this to select a particular P-Mode.

A conformant implementation, acting as Receiver, MUST take the presence, values and attribute values of all ebMS3 headers set using P-Mode parameters, including the **eb:From/eb:PartyID**, **eb:From/Role, eb:To/PartyID, eb:To/Role**, **eb:Service**, **eb:Action** and **eb:AgreementRef** header into account when selecting the applicable P-Mode. It MUST be able to send and receive messages in which the optional **pmode** attribute of **eb:AgreementRef** is not set.

As in the AS4 ebHandler profile, support for **eb:MessageProperties** is REQUIRED in this profile. It MUST be possible to set the **type** attribute for message properties (see https://issues.oasis-open.org/browse/EBXMLMSG-2). No use of specific Message Properties is specified in this Common Profile, but the feature is used in the optional Four Corner Topology enhancement.

3.2.3. Packaging Payloads

Section 5.1.1 of the ebMS3 Core Specification [EBMS3] requires implementations to process both non-multipart (simple SOAP) messages and multipart (SOAP-with-attachments) messages, and this is a requirement for the AS4 ebHandler Conformance Profile. AS4 messages based on this eDelivery AS4 profile MUST NOT include any payload content in the SOAP body. Due to the mandatory use of the AS4 compression feature in this profile (see section 2.2.3.3), XML payloads MAY be converted to binary data, which is carried in separate MIME parts and not in the SOAP Body. Conformant eDelivery AS4 messages therefore always have an empty SOAP Body.

The ebMS3 mechanism of supporting "external" payloads via hyperlink references (as mentioned in section 5.2.2.12 of the ebMS3 Core Specification [EBMS3]) MUST NOT be used.

If AS4 is used to exchange multiple payload parts in a single message, and there are cross-references between payloads encoded using *Content-ID* resource locator syntax [RFC2392], the submitting Producer MUST be able to set *Content-ID* values for MIME payload parts and the Receiving MSH MUST make payload part *Content-ID* information available to the Consumer. Note that as an alternative to the use of the MIME Content-ID AS4 offers the option to assign part properties to each payload included in the message which can also be used for cross-references between payloads. To obviate the need for cross-references between payloads the Producer can also package them into one container part before submitting to the MSH.

A Producer that submits a message with multiple payloads to an MSH MUST be able to control the order of the corresponding **eb:PartInfo** elements in the ebMS3 **eb:PayloadInfo** element. An MSH that delivers a message with multiple payloads to a Consumer MUST provide information on the order of the corresponding **eb:PartInfo** elements in the ebMS3 **eb:PayloadInfo** element.

Support for **eb:PartProperties** is REQUIRED. A Producer MUST be able to specify properties and values for a submitted payload part and the Consumer MUST be able to read specified properties and values for delivered payload parts.

3.2.4. Test Service

Section 5.2.2 of [EBMS3] defines a server test feature that allows a party to "Ping" a communication partner. The feature is based on messages with the values of:

- **eb:UserMessage/eb:CollaborationInfo/eb:Service** set to http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/service
- **eb:UserMessage/eb:CollaborationInfo/eb:Action** set to http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/test

This feature allows communication partners to perform a basic test of the communication configuration (including security at network, transport and message layer, and reliability) in any environment, including the production environment. A conformant implementation MUST allow configuration of processing modes for exchanges involving these values for service and action. It MUST be possible to configure an MSH such that messages with these values are not delivered to any Consumer business application.

3.2.5. Error Handling

For the error handling feature this profile specifies that errors MUST be reported and transmitted synchronously to the Sender and SHOULD be reported to the Consumer and the Producer. Note that the two error notification parameters do not affect interoperability between Sender and Receiver MSH.

- The parameter **PMode[].ErrorHandling.Report.AsResponse** MUST be set to the value *true*.
- The parameter **PMode[].ErrorHandling.Report.ProcessErrorNotifyConsumer** SHOULD be set to the value *true*.
- The parameter **PMode[].ErrorHandling.Report.ProcessErrorNotifyProducer** SHOULD be set to the value *true.*

The AS4 option to transmit errors using asynchronous messages MUST NOT be used.

As the interface between Producer and MSH and between MSH and Consumer is not defined in AS4, the formats and protocols used to report errors to Producer or Consumer are implementation-dependent.

Configuration of additional error handling specific to Reliable Messaging and Reception Awareness is described in section 3.3.2.

3.2.6. Security

AS4 message exchanges can be secured at multiple communication layers: the network layer, the transport layer, the message layer and the payload layer. The first and last of these are not normally handled by B2B communication software and therefore out of scope for this sub-section. Transport layer security is addressed, even though its functionality may be offloaded to another infrastructure component.

This section provides parameter settings based on multiple published sets of best practices. It is noted that after publication of this specification, previously unknown vulnerabilities may be discovered in the security algorithms, formats and exchange protocols specified in this section. Such discoveries SHOULD lead to revisions to this specification.

Transport Layer Security

When using AS4, Transport Layer Security (TLS) is an option to provide message confidentiality and authentication. Server authentication, using a server certificate, allows the client to make sure the HTTPS connection is set up with the right server. When a message is pushed, the Sending MSH authenticates the HTTPS server of the Receiving MSH.

Guidance on the use of Transport Layer Security is published in the ENISA *Algorithms, Key Sizes and Parameters Report 2013* [ENISAAKSP] and in a *Mindeststandard* of the *Bundesamt für Sicherheit in der Informationstechnik* [BSITLS]. If TLS is handled by the AS4 message handler (and not off-loaded to some infrastructure component), then:

- It MUST be possible to configure the accepted TLS version(s) in the AS4 message handler. The ENISA and BSI reports state that TLS 1.0 and TLS 1.1 SHOULD NOT be used in new applications. Older version such as SSL 2.0 [RFC6176] and SSL 3.0 MUST NOT be used. Implementations conformant with this profile MUST therefore support TLS 1.2 [RFC5246].
- It MUST be possible to configure accepted TLS cipher suites in the AS4 message handler. IANA publishes a list of TLS cipher suites [TLSSP], only a subset of which the ENISA Report considers future-proof (see [ENISAAKSP], section 5.1.2). Implementations MUST support cipher suites included in this subset. Vendors SHOULD add support for newer, safer cipher suites, as and when such suites are published by IANA/IETF.
- Support for SSL 3.0 and for cipher suites that are not currently considered secure MUST be disabled by default.
- Perfect Forward Secrecy, which is required in [BSITLS], is supported by the TLS_ECDHE_* and TLS_DHE_* cipher suites, which are therefore preferred and SHOULD be supported.

If TLS is not handled by the AS4 message handler, but by another component, then these requirements MUST be addressed by that component.

Transport Layer client authentication authenticates the Sender (when used with the Push MEP binding) or Receiver (when used with Pull). Since this profile uses WS-Security for message authentication, the use of client authentication at the Transport Layer can be considered redundant. Whether or not client authentication is to be used depends on the deployment environment. To support deployments that do require client authentication, implementations MUST allow Transport Layer client authentication to be configured for an AS4 HTTPS endpoint. Mutual Authentication or two way TLS Authentication is a combination of client and server authentication.

Message Layer Security

To provide message layer protection for AS4 messages, this profile REQUIRES the use of the following Web Services Security version 1.1.1 OASIS specifications, profiled in ebMS3.0 [EBMS3] and AS4 [AS4]:

- Web Services Security SOAP Message Security [WSSSMS].
- Web Services Security X.509 Certificate Token Profile [WSSX509].
- Web Services Security SOAP Message with Attachments (SwA) Profile [WSSSWA].

The X.509 Certificate Token Profile supports the signing and encryption of AS4 messages. This profile REQUIRES the use of X.509 tokens for message signing and encryption, for all AS4 exchanges. The AS4 option of using Username Tokens, which is supported in the AS4 ebHandler Conformance Profile, MUST NOT be used. The AS4 message MUST be signed prior to being encrypted (see section 7.6 of [EBMS3CORE]).

AS4 message signing is based on the W3C XML Signature recommendation used by WS-Security. AS4 can be configured to use specific digest and signature algorithms based on identifiers defined in this recommendation. At the time of publication of the AS4 specification [AS4], the current version of W3C XML Signature was the June 2008, XML Signature, Second Edition specification [XMLDSIG]. The current version is the April 2013, Version 1.1 specification [XMLDSIG1], which defines important new algorithm identifiers, including identifiers for SHA2, and deprecates SHA1, in line with guidance from ENISA [ENISAAKSP].

This eDelivery AS4 profile uses the following AS4 parameters and values:

- The **PMode[].Security.X509.Sign** parameter MUST be set in accordance with section 5.1.4 and 5.1.5 of [AS4].
- The **PMode[].Security.X509.Signature.HashFunction** parameter MUST be set to *http://www.w3.org/2001/04/xmlenc#sha256* .
- The **PMode[].Security.X509.Signature.Algorithm** parameter MUST be set to *http://www.w3.org/2001/04/xmldsig-more#rsa-sha256* .

This eDelivery AS4 Common Profile anticipates an update to the OASIS AS4 specification to reference this newer version of the XML Signature specification.

The use of XML Signature in AS4 provides Non Repudiation of Origin (NRO) at Message Exchange level.

For encryption, WS-Security leverages the W3C XML Encryption recommendation used by WS-Security. The following AS4 processing mode parameters configure this feature:

- The **PMode[].Security. X509.Encryption.Encrypt** parameter MUST be set in accordance with section 5.1.6 and 5.1.7 of [AS4].
- The parameter **PMode[].Security.X509.Encryption.Algorithm** MUST be set to *http://www.w3.org/2009/xmlenc11#aes128-gcm* . This is the algorithm used as value for the **Algorithm** attribute of **xenc:EncryptionMethod** on **xenc:EncryptedData**.

As specified in section 5.1.6 of [AS4] and in https://issues.oasis-open.org/browse/EBXMLMSG-111, when XML Encryption is used, all and only payload MIME parts MUST be encrypted. The **eb:Messaging** header and any of its sub-elements MUST NOT be encrypted.

AS4 also references an older version of XML Encryption than the current one, [XMLENC] instead of [XMLENC1]. The AES 128 algorithm [AES] was already referenced in that earlier version. AES is fully consistent with current recommendations for “near term” future system use [ENISAAKSP]. However, the newer W3C XML Encryption specification recommends AES GCM strongly over any CBC block encryption algorithms.

In WS-Security, there are three mechanisms to reference a security token (see section 3.2 in [WSSX509]). The ebMS3 and AS4 specifications do not constrain this, neither do they provide a P-Mode parameter to select a specific option. For interoperability, implementations SHOULD therefore implement all three options. It is RECOMMENDED that implementations allow configuration of security token reference type, so that a compatible type can be selected for a communication partner. Note that as *BinarySecurityToken* is the most widely implemented option for security token references in AS4 implementations, implementations SHOULD implement this option.

Key Transport algorithms are public key encryption algorithms especially specified for encrypting and decrypting keys, such as symmetric keys used for encryption of message content. No parameter is defined to support configuration of key transport in [EBMS3]. Implementations MUST use the following algorithms on outbound messages and MUST accept them on inbound messages.:

- For encryption algorithm, *http://www.w3.org/2009/xmlenc11#rsa-oaep* . This is the algorithm used as value for the **Algorithm** attribute of **xenc:EncryptionMethod** on **xenc:EncryptedKey**.
- As mask generation function, *http://www.w3.org/2009/xmlenc11#mgf1sha256* . This is the algorithm used as value for the **Algorithm** attribute of **xenc:MGF** in **xenc:EncryptionMethod**.
- As digest generation function, *http://www.w3.org/2001/04/xmlenc#sha256* . This is the algorithm used as value for the **Algorithm** attribute on **ds:DigestMethod** in **xenc:EncryptionMethod**.

For backwards compatibility with implementations of versions of eDelivery AS4 prior to version 1.13, implementations MAY also accept, on incoming messages, the use of other key transport algorithm options specified in section 5.5 of [XMLENC1].

3.2.7. Configuration Management

AS4 implementations conformant with the eDelivery AS4 Common Profile SHOULD provide an Application Programming Interface (API) to manage (i.e. create, read, update and delete) AS4 configuration data, including Processing Mode parameter sets and X.509 certificates used for AS4 message exchanges. This is an abstract requirement only. Conformance to any specific API or data format is NOT REQUIRED.

3.2.8. Networking

Conformant implementations MUST support IPv4 and IPv6 networking.

3.3. Use of AS4 Additional Features

The OASIS AS4 specification defines a number of Additional Features. The eDelivery AS4 Common Profile REQUIRES support for the AS4 Compression and Reception Awareness and Duplication Detection features. None of the other AS4 Additional Features are used in the eDelivery AS4 Common Profile.

3.3.1. Compression

The OASIS AS4 specification defines Payload Compression as one of its additional features. Payload compression is a useful feature for many content types, including XML content, as it can speed up:

- Message layer security processing, as there is less data to be signed/validated and encrypted/decrypted.
- Transmission of data over networks, as the message is smaller.

To compress the payload(s) of a message payload, the GZIP [RFC1952] compression algorithm MUST be used. GZIP is the only compression type currently supported in OASIS AS4.

The **PartInfo** element in the message header that relates to a compressed payload part MUST have an **eb:Property** element with its **name** attribute set to the value *CompressionType*. The content type of a compressed payload part MUST be *application/gzip*.

```
1  <eb:Property name="CompressionType">application/gzip</eb:Property>
```

Presence of this part property is an indicator to the Receiving MSH that the Sending MSH has compressed a payload part. The receiving AS4 MSH MUST decompress any payload part(s) compressed by the Sending MSH before delivering the message.

When compression, signature and/or encryption are required, AS4 specifies that any attached payload(s) MUST be compressed prior to being signed and encrypted.

Packaging requirements:

- An **`eb:PartInfo/eb:PartProperties/eb:Property/@name`** attribute with value *MimeType* is REQUIRED to identify the MIME type of the payload before compression was applied.
- For XML payloads, an **`eb:PartInfo/eb:PartProperties/eb:Property/@name`** attribute with value *CharacterSet* is RECOMMENDED to identify the character set of the payload before compression was applied. The value of this property MUST conform to the values defined in section 4.3.3 of [XML10].

```
1  <eb:PartInfo href="cid:a0fe4348-bbde-461b-8edc-070d458a1fc5@example.com">
2       <eb:PartProperties>
3           <eb:Property name="MimeType">application/xml</eb:Property >
4           <eb:Property name="CharacterSet">utf-8</eb:Property >
5           <eb:Property name="CompressionType">application/gzip</eb:Property >
6       </eb:PartProperties>
7  </eb:PartInfo>
```

Use of compression is configured using the P-Mode parameter **PMode[].PayloadService.CompressionType**. This parameter is defined as follows in [AS4]:

- If the parameter is present with a value *application/gzip*: the AS4 Sending MSH SHOULD compress the attached payload(s) over this MEP segment. However, GZIP compression of payloads in data formats that provide native, built-in compression typically does not result in good compression ratios and is therefore NOT REQUIRED.
- If the parameter is absent (default), no compression is used over this MEP segment.

The parameter is REQUIRED with the value *application/gzip* in this eDelivery AS4 Common Profile.

In case an error occurs during decompression, the following error MUST be used: Code = EBMS:0303, Short Description = DecompressionFailure, Severity = Failure, Category = Communication.

The AS4 compression feature specifies the use of compression at the message layer, the structure and format of an AS4 message that uses AS4 compression, and the behavior of the Sending and Receiving MSH. Setting the AS4 compression P-Mode parameter enables the compression feature in the sending MSH, but use of compression is NOT REQUIRED for messages configured using the P-Mode as noted in https://issues.oasis-open.org/browse/EBXMLMSG-79. Whether or not compression is applied, for messages for which AS4 compression is enabled in its P-Mode, is therefore left to the implementation of the sending MSH.

A receiving MSH MUST NOT reject messages with payloads that are not compressed even though AS4 compression is specified in the P-Mode. However, the receiving MSH is REQUIRED to decompress any compressed payloads for messages for which the P-Mode specifies the use of AS4 compression, and for which the **CompressionType** part property is set to *application/gzip*.

As the **CompressionType** property is used by the AS4 Compression feature, which controls its presence or absence using a Processing Mode parameter, the property MUST NOT be set by the Producer and it MUST NOT be passed to the Consumer.

3.3.2. Reliable Messaging and Non-Repudiation of Receipt

For **Reliable Messaging** this profile specifies that AS4 non-repudiation receipts MUST be sent synchronously for each message type.

- The parameter **PMode[].Security.SendReceipt.NonRepudiation** MUST be set to the value *true*.
- The parameter **PMode[].Security.SendReceipt.ReplyPattern** MUST be set to the value *Response*.

The AS4 option to transmit receipts using asynchronous messages MUST NOT be used.

An AS4 receipt indicates that the message has been "successfully processed by the Receiving MSH (i.e. not just “received”)" [AS4].

In this eDelivery AS4 Common Profile the use of the AS4 Reception Awareness feature is REQUIRED. This feature provides a built-in Retry mechanism that can help overcome temporary network or other issues and detection of message duplicates.

- The parameter **PMode[].ReceptionAwareness** MUST be set to *true*.
- The parameter **PMode[].ReceptionAwareness.Retry** MUST be set to *true*.
- The parameter **PMode[].ReceptionAwareness.DuplicateDetection** MUST be set to *true*.

The **PMode[].ReceptionAwareness.Retry.Parameters** and related **PMode[].ReceptionAwareness.DuplicateDetection.Parameters** are sets of parameters configuring retries and duplicate detection. These parameters are not fully specified in [AS4] and are implementation-dependent. Implementations MUST support configuration of parameters for retries and duplicate detection.
Reception awareness errors generated by the Sender MUST be reported to the Submitting application:

- The parameter **PMode[].ErrorHandling.Report.MissingReceiptNotifyProducer** MUST be set to *true*.
- The parameter **PMode[].ErrorHandling.Report.SenderErrorsTo** MUST not be set. There is no support for reporting sender errors to a third party.

In the ebMS3 Core Specification [EBMS3], the EBMS:0202 error, DeliveryFailure is defined in the context of the ebMS3 Reliable Messaging module, which is distinct from the AS4 Reception Awareness feature. It states that the DeliveryFailure error can be generated either on the Sender or on the Receiver side.

- If no AS4 Receipt is returned by the Receiving MSH to the Sending MSH for a message, an EBMS:0301, MissingReceipt, Reception Awareness error MUST be generated for the message in the Sending MSH. This error MUST be reported to the Producer, because the parameter **PMode[].ErrorHandling.Report.MissingReceiptNotifyProducer** is set to *true*. This obviates the need for both generating and reporting a DeliveryFailure on the Sender Side.
- According to the OASIS AS4 specification, if a Receiving MSH cannot successfully complete processing of the message, it MUST NOT return a Receipt. The specification does not define a separate reception awareness error that is the (negative) counterpart to a (positive) AS4 receipt. In this situation, a Receiving eDelivery AS4 MSH SHOULD return the EBMS0202, DeliveryFailure error to the Sending MSH, which MUST report this error to the Producer.

Note that, if the Receiving MSH does not return a DeliveryFailure error, or if, for some reason, transmission of this error to the Sending MSH fails, the Sending MSH MUST still generate and report the Missing Receipt error.

In the ebMS3 Core Specification [EBMS3], reporting of Delivery Failures to the Producer is configured using the parameter **PMode[].Errorhandling.DeliveryFailuresNotifyProducer**. AS4 implementations may not support this parameter as it is linked to the ebMS3 Reliable Messaging module (see also https://issues.oasis-open.org/browse/EBXMLMSG-59), which is optional in AS4 and not used in this eDelivery profile. When using AS4 implementations that use this parameter for configuration of reporting by the Sending MSH of delivery failures to the Producer, this parameter MUST be set to *true.*

The WS-ReliableMessaging and WS-Reliability protocols, which are optional features in ebMS 3.0, MUST NOT be used in eDelivery AS4. Therefore, the **PMode[].Reliability** parameters MUST NOT be used.

3.4. Common Usage Profile

This section specifies how implementations that conform to the eDelivery AS4 ebHandler MUST be configured and deployed. This is similar to the concept of Usage Agreements in section 5 of [AS4] as it does not constrain how AS4 implementations are implemented, but rather how they are configured and used. The audience for this section are operators/administrators of AS4 implementations and B2B integration project teams. The structure of this chapter also partly mirrors the structure of [EBMS3], and furthermore covers some aspects outside core B2B messaging functionality.

3.4.1. Party Identification

This profile REQUIRES an MSH to not include more than one **eb:PartyId** element in the **eb:UserMessage/eb:PartyInfo/eb:From** and **eb:UserMessage/eb:PartyInfo/eb:To** elements.

For party types registered in ISO 6523:

- The ISO 6523 code MUST used.
- If a **type** attribute is used, it SHOULD be set to an ebCore Party ID type format value.
- If no **type** attribute is used, an ebCore Party Identifier identifier in URN format SHOULD be used.

The use of ebCore Party ID MUST follow the profiling specified in the eDelivery ebCore Party Id specification [eDelivery-EBCORE].

3.4.2. Correlation

AS4 provides multiple mechanisms to correlate messages within a particular flow.

1. **eb:UserMessage/eb:MessageInfo/eb:RefToMessageId** provides a way to express that a message is a response to a single specific previous message. Presence of an **eb:RefToMessageId** is required in response messages in Two Way message exchanges. By default, exchanges are considered One Way.
2. **eb:UserMessage/eb:CollaborationInfo/eb:ConversationId** provides a more general way to associate a message with an ongoing conversation, without requiring a message to be a response to a single specific previous message, but allowing update messages to existing conversations from both Sender and Receiver of the original message.

The ebMS3 and AS4 specifications do not constrain the use of the elements **eb:RefToMessageId** and **eb:ConversationId**, but the following profiling applies:

1. **eb:UserMessage/eb:MessageInfo/eb:RefToMessageId** is to be used to support message exchanges that are modeled as request-response interactions. In the response message, the value of the element MUST be set to the value of the **eb:UserMessage/eb:MessageInfo/eb:MessageId** element in the request message.
2. **eb:UserMessage/eb:CollaborationInfo/eb:ConversationId** MUST be included in any AS4 message (as it is a mandatory element). The value MUST be consistent with the data type specified in the ebMS3 XML schema, which is *xs:token*. In a Two Way exchange, the value of the **eb:ConversationId** in the response MUST be the same as the value of the **eb:ConversationId** in the request message.

3.4.3. Test Service

Deployments of the eDelivery AS4 Common Profile MUST deploy, for each combination of values for the three parameters **PMode.Initiator.Party**, **PMode.Responder.Party**, **PMode.Agreement** defined in some deployed P-Mode, a P-Mode that has these values for these three parameters and uses the test service and action (see discussion above in the ebHandler Feature Set overview). For these three parameters, for each configured leg, for parameters other than **PMode[].BusinessInfo.Service** and **PMode[].BusinessInfo.Action**, this test service configuration MUST have the same parameters as the non-test service. This includes the values including the **PMode[].Security.X509** parameters.This allows communication partners to perform a basic test of the communication configuration (including security configuration at network, transport and message layer, and reliability) in any environment, including the production environment.

Deployments of the eDelivery AS4 Common Profile MUST be configured such that messages for the test service are not delivered to any business application.

3.4.4. Service, Action and Role

The ebMS3 specification [EBMS3] defines the **eb:Service** element as: “This REQUIRED element occurs once. It is a string identifying the service that acts on the message and it is specified by the designer of the service.” The header is of XML schema type non-empty-string and its value configured using the **PMode[].BusinessInfo.Service** parameter, meaning communication partners are expected to define specific values for specific process modes, i.e. for various types of messages. The **eb:Service** element can have a **type** attribute to categorize services. It is not profiled in the eDelivery Common Usage Profile.

In [EBMS3], **eb:Action** is defined as: “This REQUIRED element occurs once. It is a string identifying the action the User message is intended to invoke on a particular service and it is specified by the designer of the service.” For each message exchange, [EBMS3] requires setting the values **PMode.Initiator.Role** and **PMode.Responder.Role**, which are used to set **eb:From/eb:Role** and **eb:To/eb:Role** values.

Users of the eDelivery AS4 Profile MUST agree on values for Services and, for each Service, the associated Actions. For each Service and Action, the From Role and To Role MUST be defined. This profile only provides high-level constraints on naming conventions on Service and Action.

- The value of **eb:Service** SHOULD identify a set of related business transactions or other message exchanges in the context of a business process or use case.
- The value of **eb:Action** SHOULD identify the different types of business transactions or other message exchanges in the context of an identified Service. This MAY be an identifier of a document type, if the exchange of a document of that type unambiguously identifies the purpose and requested action in the context of the Service.

3.4.5. Security

This profile is intended to support exchange of AS4 messages using either the public Internet or private networks. Each party is individually responsible to implement security measures to protect access to its IT infrastructure.

Parties SHOULD use firewalls to restrict incoming and/or outgoing message flows to specific IP addresses, or address ranges. Parties therefore:

- MUST use static IP addresses (or IP address ranges) for inbound and outbound AS4 HTTPS connections.
- MUST communicate all IP addresses (or IP address ranges) used for outgoing and incoming connections to their trading partners, also covering any passive nodes in active-passive clusters. Note that the IP address of the HTTPS endpoint at which a party accepts incoming pushed AS4 messages may differ from the IP address (or addresses) used by that party for outbound AS4 connections.
- MUST notify their partners about any IP address changes sufficiently in advance to allow firewall and other configuration changes to be applied.

The Transport Layer Security settings defined in the eDelivery AS4 ebHandler Feature Set section MAY be implemented in the AS4 communication server (as specified above in the section covering the ebHandler Feature Set) but TLS MAY also be offloaded to a separate infrastructure component (such as a firewall, proxy server or router). In that case, the recommendations on TLS version and cipher suites [OSSLTLS] specified section in 3.2.6 MUST be addressed by that component.

The TLS cipher suites recommended in section 3.2.6. are supported in recent versions of TLS toolkits and therefore are available for use. Support for these suites is RECOMMENDED. Whether or not less secure cipher suites (which are only recommended for legacy applications) are allowed is a local policy decision.

This profile does NOT REQUIRE the use of client authentication. Client authentication MAY be a requirement in the networking policy of individual parties that the AS4 deployment needs to meet.

The following parameters control configuration of security at the message layer:

- The **PMode[].Security.X509.Signature.Certificate** parameter MUST be set to a value matching the signing certificate of the Sending MSH.
- The **PMode[].Security.X509.Encryption.Certificate** parameter MUST be set to a value matching the encryption certificate of the Receiving MSH.

The eDelivery AS4 profile provides Non Repudiation of Origin and Receipt at the level of Message Exchange. Note that the use of non-repudiation information to resolve any disputes requires the communication partners to store data such as the exchanged AS4 messages, AS4 receipts corresponding to these messages, and possibly other verification data such as OCSP responses, for the period during which any such disputes may arise. The minimum retention period to be applied SHOULD be specified in a formal policy and be consistent with legal-regulatory requirements, business needs, and available storage/processing capacities. The AS4 message service handlers MUST be configured accordingly.

3.4.6. Message Payload and Flow Profile

In any exchange involving an AS4 **eb:UserMessage** that has a structured document payload and any number of associated payloads, the **eb:UserMessage** MUST reference the structured document payload part using the first **eb:PartInfo** element in the **eb:PayloadInfo** header. Subsequent **eb:PartInfo** elements MAY be used to reference additional structured or unstructured payload parts. The payload part referenced using the first **eb:PartInfo** element is the leading payload part for business processing. Any payload parts other than the leading

document payload part MUST NOT to be processed in isolation but only as adjuncts to the business document. Structured document, attachments and metadata MUST be submitted and delivered as a logical unit.

The format and schema of structured document payloads SHOULD be agreed among sender and receiver parties and SHOULD be XML or JSON, but other structured data types MAY be supported in specific business processes or contexts.

Note that in ebMS3, when payloads are carried in MIME parts, **eb:PartInfo** elements reference the corresponding MIME parts by using the Content-ID value of those parts in their **href** attribute. The order of referenced MIME payload parts in the MIME envelope MAY differ from the order of corresponding **eb:PartInfo** elements in the ebMS3 **eb:PayloadInfo** header.

For each business process, for each exchange involving exchange of XML documents, a specification SHOULD be agreed that defines the XML schema definition (XSD) that the XML document is expected to conform to. The mapping from AS4 headers including **eb:Service** and **eb:Action** to XSDs SHOULD be unique, allowing Receivers to validate XML documents using a specific XML schema. Note that there is no requirement that any XML schema validation is done by the AS4 MSH.

While the AS4 protocol has no inherent limitation on message size, users SHOULD take into consideration the practical limits that are imposed by some AS4 implementations (e.g. the use of certain data types in some security libraries imposes a maximum size of approximately 2 GB) or by deployment contexts of AS4 implementations (e.g. amount of memory or storage available to the AS4 server).

Any additional profiling of payloads and message flows is out of scope for this specification.

3.4.7. Environments

Message exchange solutions are part of the overall IT service life-cycle, in which different environments are operated (often in parallel) for development, test, pre-production (in some companies referred to as "acceptance environments" or "QA environments") and production. Development and test are typically internal environments in which trading partners are simulated using stubs. When exchanging messages (in either pre-production or production environments), parties MUST target the appropriate environment. In order to prevent a configuration error from causing non-production messages to be delivered to production environments or vice versa, parties SHOULD configure processing modes at message handlers so that messages from one type of environment cannot be accepted inadvertently by a different type of environment.

3.4.8. Networking

This eDelivery AS4 profile MAY be deployed on IPv4 and/or IPv6 networking.

3.5. P-Mode Parameters

The following table summarizes the P-Mode settings as defined in this eDelivery Common Profile.

Processing Mode Parameter	**Value in the eDelivery Common Profile**
PMode.ID	Not profiled
PMode.Agreement	Not profiled
PMode.MEP	http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/oneWay http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/twoWay
PMode.MEPBinding	http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/push http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/pushAndPush
PMode.Initiator.Party	Identifier (and possibly a *type* attribute value). For types registered in ISO 6523, the ISO 6523 code MUST be used and the ebCore Party ID type format SHOULD be used.
PMode.Initiator.Role	Not profiled

Processing Mode Parameter	Value in the eDelivery Common Profile
PMode.Initiator.Authorization.username	Not used
PMode.Initiator.Authorization.password	Not used
PMode.Responder.Party	Party identifier (and possibly an identifier *type* attribute value). For types registered in ISO 6523, the ISO 6523 code MUST be used and the ebCore Party ID type format SHOULD be used.
PMode.Responder.Authorization. username	Not used
PMode.Responder.Authorization. password	Not used
PMode[].Protocol.Address	Required, https URL of the receiver.
PMode[].Protocol.SOAPVersion	1.2
PMode[].BusinessInfo.Service	SHOULD identify a set of related business transactions or other message exchanges in the context of a business process or use case.
PMode[].BusinessInfo.Action	SHOULD identify the different types of business transactions or other message exchanges in the context of an identified Service. This MAY be an identifier of a document type, if the exchange of a document of that type unambiguously identifies the purpose and requested action in the context of the Service.
PMode[].BusinessInfo. Properties	Support required.
PMode[].BusinessInfo.MPC	Not profiled
PMode[].BusinessInfo.PayloadProfile	Not profiled
PMode[].Errorhandling.Report.SenderErrorsTo	Not used
PMode[].Errorhandling.Report.ReceiverErrorsTo	Not used
PMode[].Errorhandling.Report.AsResponse	True
PMode[].Errorhandling.Report.ProcessErrorNotifyConsumer	True (Recommended)
PMode[].ErrorHandling.Report.ProcessErrorNotifyProducer	True (Recommended)
PMode[].Errorhandling.DeliveryFailuresNotifyProducer	True (Recommended)
PMode[].ErrorHandling.Report.MissingReceiptNotifyProducer	True
PMode[].Reliability	Not used
PMode[].Security.WSSversion	1.1.1
PMode[].Security.X509.Sign	True
PMode[].Security. X509.Signature.Certificate	Signing Certificate of the Sender
PMode[].Security. X509.Signature.HashFunction	http://www.w3.org/2001/04/xmlenc#sha256
PMode[].Security.X509.Signature.Algorithm	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
PMode[].Security.X509.Encryption.Encrypt	True
PMode[].Security.X509.Encryption.Certificate	Encryption Certificate of the Receiver
PMode[].Security.X509.Encryption.Algorithm	http://www.w3.org/2009/xmlenc11#aes128-gcm
PMode[].Security.X509.Encryption.MinimalStrength	128

Processing Mode Parameter	Value in the eDelivery Common Profile
PMode[].Security.UsernameToken.username	Not used
PMode[].Security.UsernameToken.password	Not used
PMode[].Security.UsernameToken.Digest	Not used
PMode[].Security.UsernameToken.Nonce	Not used
PMode[].Security.UsernameToken.Created	Not used
PMode[].Security.PModeAuthorize	False
PMode[].Security.SendReceipt	True
PMode[].Security.SendReceipt.NonRepudiation	True
PMode[].Security.SendReceipt.ReplyPattern	Response
PMode[].PayloadService.CompressionType	application/gzip
PMode[].ReceptionAwareness	True
PMode[].ReceptionAwareness.Retry	True
PMode[].ReceptionAwareness.Retry.Parameters	Not profiled
PMode[].ReceptionAwareness.DuplicateDetection	True
PMode[].ReceptionAwareness.DetectDuplicates.Parameters	Not profiled
PMode[].BusinessInfo.subMPCext	Not used

4. Profile Enhancements

The eDelivery AS4 Common Profile MAY be enhanced with Profile Enhancements. This section specifies four such Profile Enhancements. The Profile Enhancements MUST be used in conjunction with the Common Profile. The four currently defined Profile Enhancements are mutually compatible, so they MAY be used in combination. However, this is NOT REQUIRED. Implementations MAY also limit support for one Enhancement to be used only in combination with another Enhancement. Future versions, or profiles, of this eDelivery AS4 Profile MAY specify additional Profile Enhancement modules.

4.1. Four Corner Topology

4.1.1. Introduction

The OASIS ebMS3 and AS4 specifications are specifications for point-to-point message exchange between two Message Service Handlers. However, eDelivery AS4 is also used in situations where messages are exchanged by Access Points on behalf of other parties. In this so-called four corner topology, from an end-to-end perspective, there are four rather than two parties involved in the message exchange. Two parties are the original sender and final recipient parties. The other two parties are Access Points that route messages from the original sender to the final recipient and reverse route response messages. The four parties are conventionally referred to using *Cn* labels, where *C* stands for "corner" and the *n* is one of the digits 1 to 4:

- *C1* is the original sender party.
- *C2* is an Access Point that sends messages on behalf of *C1*.
- *C3* is an Access Point that receives messages on behalf of *C4.*
- *C4* is the final recipient party.

The Four Corner Topology Profile Enhancement supports the use of eDelivery AS4 as a common solution to interconnect parties using Access Points. The Four Corner topology is typically used in situations where Access Points interconnect heterogeneous messaging domains, following the Messaging Bridge pattern:

- The C2 is a Receiver and Consumer for the C1-C2 message exchange, and a Producer and Sender for the C2-C3 message exchange.
- The C3 is a Receiver and Consumer for the C2-C3 message exchange, and a Producer and Sender for the C3-C4 message exchange.

However, this Profile Enhancement MAY also be used to identify distinct Producer and Consumer components, not involving the Messaging Bridge pattern. As the Common Profile applies to the C2-C3 exchange, the exchange involves an AS4 push from C2 as Initiator to C3 as Responder. The exchanges between C1-C2 and C3-C4 are NOT REQUIRED to use AS4 and are out of scope for this specification. Depending on the integration (e.g. store-and-forward, mailbox etc.), the transfer from C3 to C4 may be initiated by C3 or C4.

4.1.2. Addressing and Party Identification

This Enhancement to the Common Profile specifies the use of eDelivery AS4 in four corner message exchanges. It defines conventions for the use of ebMS3 message headers and configuration of the corresponding processing mode parameters.

For Message Packaging this Profile Enhancement constrains values for several elements in the AS4 message header and the overall message structure. In scenarios where AS4 is used for point-to-point communication between end entities, the **eb:From** and **eb:To** headers in the **eb:UserMessage/eb:PartyInfo** identify the Sender and Receiver respectively. In a four-corner-model, the Sender and Receiver of AS4 messages are the inner corner Access Points (C2, C3), not the outer corner parties (C1, C4). To facilitate the use of unmodified AS4 messaging implementations and to simplify configuration of AS4 message service handlers, **eb:From/eb:PartyId** and **eb:To/eb:PartyId** MUST identify the inner corner Access Points.

To be able to route a received message, the receiving Access Point (C3) needs to be able to determine the final recipient (C4). This information is generally available in a structured payload. However, using information from a structured payload assumes an understanding of the schema on which the payload is based. In order to allow Access Points to process payloads of any type, it is desirable to adopt a mechanism that is independent of particular schemas. Furthermore, in some situations there MAY be a requirement to route unstructured or encrypted data. This Profile Enhancement therefore uses the ebMS3 property mechanism to identify C1 and C4. The property mechanism allows the use of arbitrary property-value pairs in an AS4 message and is independent of payload format or structure.

When used in a Four Corner typology:

- A property named **originalSender** MUST be added to the message that identifies the original sender (C1) party.
- A property named **finalRecipient** MUST be added to the message that identifies the final recipient (C4) Party

As with the **eb:From/eb:PartyId** and **eb:To/eb:PartyId** headers, the **type** attribute MAY be used with these two properties to categorize party identifier types. As identifier system and format for addressing, the ISO 6523 code MUST be used for party types registered in ISO 6523 and the ebCore Party ID format SHOULD be used as specified in [eDelivery-EBCORE].

This profile defines an additional, optional third property:

- A property named **trackingIdentifier** MAY be added to the message to include an identifier (in arbitrary string format) that allows end-to-end tracking of messages in a four-corner exchange. Its value MAY be set to the value of an identifier for the message from C1 to C2 that the AS4 message relates to. This allows tracking and tracing of messages.

A key advantage of the use of ebMS3 message properties is that no constraints are imposed on message payload. It is possible to transport, route and forward any payload, or combination of payloads, even if unstructured, binary or encrypted.

When using the eDelivery AS4 Four Corner Topology Profile Enhancement:

- Values for the four corner properties MUST be set by the Producer in the submission to the C2 AS4 MSH.
- The use of the four corner properties MUST be consistent with the P-Mode configuration of the C2 AS4 MSH and the C3 AS4 MSH.
- The receiver C3 AS4 MSH MUST include information about these properties and their values in message delivery and MUST use the party identified in the **finalRecipient** property as C4.

4.1.3. P-Mode Parameters

Processing Mode Parameter	Value in the eDelivery Four Corner Topology Enhancement
PMode[].BusinessInfo.Properties	Support required. In four corner exchanges, mandatory inclusion of **originalSender** and **finalRecipient** and optional inclusion of **trackingIdentifier.**

These parameters are P-Mode parameters that can be set for individual P-Modes. A conformant MSH MAY be engaged in Four Corner Topology exchanges for some partners, or for some services, and not for others.

4.1.4. Relation to SOAP and ebMS3 intermediary concepts

In a Four Corner topology, when used as Messaging Bridge, the scope of AS4 as interconnect messaging protocol is limited to the interaction between the inner two corners (C2 and C3). This means that, in terms of the ebMS3 Messaging Model:

- The Message Producer is a separate messaging component that receives messages from an original sender (C1), using a separate messaging infrastructure, and re-*submits* the message to the C2 Sender AS4 MSH.
- The Message Consumer is a separate messaging component to which the C3 Receiver AS4 MSH *delivers* the message content and metadata. That Consumer component is responsible for forwarding the message content and metadata to the final recipient (C4), using another separate messaging infrastructure.

This exchange is not to be confused with the SOAP processing model as defined in SOAP 1.2, second edition [SOAP12]. SOAP intermediaries operate in a chain of SOAP processing nodes. In a Four Corner Typology, The C1-C2 and C3-C4 connections MAY be SOAP-based, but this is NOT REQUIRED.

The ebMS3 Part 2 “Multihop” module [EBMS3P2] uses the SOAP processing model to define a multihop messaging protocol based on ebMS3 SOAP intermediaries that forward ebMS3 SOAP messages. In that model there is no submission or delivery to the intermediary MSH and end-to-end security and reliability can be provided. The eDelivery AS4 Four Corner Typology Enhancement REQUIRES use of AS4 between inner nodes only and is not based on the ebMS3 multihop advanced feature. AS4 is NOT REQUIRED in the communication from and to the outer corners.

4.1.5. Note on Non-Repudiation

The use of XML Signature in WS-Security specified in the Common Profile provides Non Repudiation of Origin (NRO) at Message Exchange level. When using the Four Corner Topology enhancement, the origin is understood to be the AS4 sender C2. Any end-to-end Non Repudiation of Origin (NRO), providing content commitment for C1, is out of scope for this Profile Enhancement. This is a difference to the ebMS3 Part 2 “Multihop” module that does provide end-to-end NRO.

The use of signed AS4 Non Repudiation Receipts as specified in the Common Profile provides Non Repudiation of Receipt (NRR) at Message Exchange level. When using the Four Corner Topology enhancement, the receiver is understood to be the AS4 receiver C3. Any end-to-end Non Repudiation of Receipt (NRR), providing content commitment for C4, is out of scope for this Profile Enhancement. This is a difference to the ebMS3 Part 2 “Multihop” module that does provide end-to-end NRR.

For more information on this topic, see the Security Controls Guidance document [SECCONTROLS].

4.2. Standard Business Document Header (SBDH)

4.2.1. Introduction

The eDelivery AS4 Common Profile MAY be used in conjunction with the UN/CEFACT Standard Business Document Header [SBDH]. SBDH is a standard XML format that encodes common message metadata, such as identification of sender and receiver, the type of the payload and the business scope, business process, business transaction, agreement, and business quality-of-service. SBDH is widely adopted in various domains.

When used in conjunction with AS4, the SBDH can be used in two ways:

- As a standalone XML document, packaged as an AS4 message payload part, separate from and not integrated in other business content parts.
- As an integrated part of an XML business payload part.

4.2.2. Using a standalone SBDH instance

When used as a standalone XML document, the SBDH serves as a specialized XML payload, which can be seen as a kind of internal header. Other content parts do not need to be modified (e.g. base64 encoded) to be used with a standalone SBDH part, and are exchanged as additional separate message payload parts.

The standalone SBDH is most useful in conjunction with non-XML documents. In this case, the SBDH and the payload have to be in separate MIME parts because they have different content types: SBDH is XML and the payload non-XML. An example use of a non-XML business payload is the ETSI ASiC container specification [ASiC]. ASiC is a non-XML file format, which has the MIME content type *application/vnd.etsi.asic-e+zip*.

Like any AS4 payload part, the SBDH payload part and the additional message parts MUST be referenced from the ebMS **eb:UserMessage** header. The **eb:UserMessage** header uses the Content ID URI [RFC2392] in its **eb:PayloadInfo** section to reference

other MIME parts in the same message as the referring MIME part. The SBDH references the additional message parts using the **sh:Manifest** element.

4.2.3. Identifiers and Cross-References in a MIME Envelope with ebMS3 Header and Standalone SDBH

The following table shows the identifiers of various parts of an AS4 messaging containing a standalone SDBH and one or more additional payloads. The MIME parts are all identified using a Content-ID header. The enveloping HTTP/MIME structure references the SOAP root MIME part using its *start* parameter. The ebMS3 header in the SOAP root part references the standalone SBDH header document payload and all other payloads from its **eb:PayloadInfo** structure. In this situation, the SBDH MUST be referenced using the first **eb:PartInfo** part reference and is considered the initial leading document. Any other payload parts MUST be referenced from the SBDH, in addition to being referenced from the AS4 header.

Note that, according to the WS-I Attachments Profile [AP], semantics should be neither given to, nor implied by the ordering of MIME parts in a message. The logical relations are established using part content identifiers and references using those identifiers only.

HTTP/MIME Envelope	AS4 header	SBDH header
Reference to mandatory SOAP root MIME part Value of the "start" parameter in the Content-Type header *Identification of the mandatory SOAP root MIME part* Content ID of the containing SOAP root MIME part	X	X
Identification of the mandatory SBDH header MIME Part Content ID of the containing MIME part	*Reference to mandatory SBDH header MIME Part* //eb:Messaging/eb:UserMessage[1]/ eb:PayloadInfo/eb:PartInfo[1]/ @href	X
Identification of the mandatory business payload part: Content ID of the containing MIME part	*Reference to mandatory business payload part:* //eb:Messaging/eb:UserMessage[1]/ eb:PayloadInfo/eb:PartInfo[2]/ @href	*Reference to mandatory business payload part:* sh:StandardBusinessDocumentHeader/ sh:Manifest/ sh:ManifestItem[1]/ sh:UniformResourceIdentifier
Identification of optional additional payload parts: Content ID of the containing MIME part	*Reference to any optional additional payload parts:* //eb:Messaging/eb:UserMessage[1]/ eb:PayloadInfo/eb:PartInfo[*3, 4 etc.*]/ @href	*Reference to any optional additional payload parts:* sh:StandardBusinessDocumentHeader/ sh:Manifest/ sh:ManifestItem[2, *3 etc.*]/ sh:UniformResourceIdentifier

Note: for readability and formatting (line wrapping) reasons, the XPath expressions in this table have been edited to include spaces.

4.2.4. Standalone SBDH Example

The following example shows a sample standalone SBDH document. It uses an **sh:Manifest** block for referencing non-XML documents or files.

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<sh:StandardBusinessDocumentHeader
    xmlns:sh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader">
    <sh:HeaderVersion>1.0</sh:HeaderVersion>
    <sh:Sender>
        <sh:Identifier Authority="urn:oasis:names:tc:ebcore:partyid-type:iso6523:0002"
            >123456789</sh:Identifier>
```

```
        <sh:ContactInformation>
            <sh:Contact>John Doe</sh:Contact>
            <sh:EmailAddress>John_Doe@purchasing.XYZretailer.com</sh:EmailAddress>
            <sh:FaxNumber>+1-212-555-1213</sh:FaxNumber>
            <sh:TelephoneNumber>+1-212-555-2122</sh:TelephoneNumber>
            <sh:ContactTypeIdentifier>Buyer</sh:ContactTypeIdentifier>
        </sh:ContactInformation>
    </sh:Sender>
    <sh:Receiver>
        <sh:Identifier Authority="urn:oasis:names:tc:ebcore:partyid-type:iso6523:0106"
            >192837465</sh:Identifier>
    </sh:Receiver>
    <sh:DocumentIdentification>
        <sh:Standard>urn:oasis:names:specification:ubl:schema:xsd:OrderResponse-2</sh:Standard>
        <sh:TypeVersion>2.0</sh:TypeVersion>
        <sh:InstanceIdentifier>100002</sh:InstanceIdentifier>
        <sh:Type>OrderResponse</sh:Type>
        <sh:CreationDateAndTime>2011-08-22T11:31:52Z</sh:CreationDateAndTime>
    </sh:DocumentIdentification>
    <sh:Manifest>
        <sh:NumberOfItems>1</sh:NumberOfItems>
        <sh:ManifestItem>
            <sh:MimeTypeQualifierCode>application/vnd.etsi.asic-e+zip</sh:MimeTypeQualifierCode>
            <sh:UniformResourceIdentifier>bfefb5d1-2f9d-4c7d-9c91-ef29c6dae43b@XYZretailer.com</sh:UniformResourceId
        </sh:ManifestItem>
    </sh:Manifest>
    <sh:BusinessScope>
        <sh:Scope>
            <sh:Type>BusinessProcess</sh:Type>
            <sh:InstanceIdentifier>ecae53d4-7473-45a6-ad70-61970dd7c4b0</sh:InstanceIdentifier>
            <sh:Identifier>cpa:123456789:192837465</sh:Identifier>
            <sh:BusinessService>
                <sh:BusinessServiceName>urn:www.cenbii.eu:profile:BII06:ver1.0</sh:BusinessServiceName>
                <sh:ServiceTransaction TypeOfServiceTransaction="RequestingServiceTransaction"
                    IsAuthenticationRequired="true" IsNonRepudiationRequired="true"
                    IsNonRepudiationOfReceiptRequired="true" IsIntelligibleCheckRequired="true"
                    IsApplicationErrorResponseRequested="true" TimeToAcknowledgeReceipt="P12H"
                    TimeToAcknowledgeAcceptance="P2D" TimeToPerform="P5D" Recurrence="3"/>
            </sh:BusinessService>
        </sh:Scope>
    </sh:BusinessScope>
</sh:StandardBusinessDocumentHeader>
```

4.2.5. AS4 and XML Payloads using SBDH Headers

As an alternative to standalone use of SBDH, the SBDH can also be an integral part of an XML business document. When used with the eDelivery Common Profile, this business document, including its SBDH part, will be packaged in a single MIME part. There are many reasons why the implementer would choose an integrated packaging approach (SBDH as part of XML document) or a non-integrated approach (standalone SBDH). The following arguments favor the integrated approach:

- If SDBH is an integral part of the XML instance document, the document can be parsed at a high level and routing and processing decisions can easily be made without taking into account other information.
- If the SBDH is contained in a separate body part, once the message is delivered to the backend application, the linkage between the two body parts can be lost and the routing / processing functionality becomes more complex.

An SBDH that is part of an XML business document that does not reference any attachments does not have a **sh:Manifest** element.

4.2.6. Using the SBDH in a Four Corner Topology

The SBDH Profile Enhancement MAY be used in conjunction with the Four Corner Topology Profile Enhancement. If, in a Four Corner context, the SBDH is used as a container for information relating to the "end to end" exchange, then the **sh:Sender** at SBDH level corresponds to the C1 original sender and the **sh:Receiver** corresponds to the C4 final recipient. In this situation, the values of the AS4 message properties specified in the Four Corner Topology Profile Enhancement can be populated and set in submission to the sending AS4 MSH, using values in the SBDH content. The consumer SHOULD validate that, for any message delivered to it, the values of the AS4 Four Corner message properties match the values of the corresponding SBDH values.

AS4 Header	SBDH

AS4 Header	SBDH
//eb:Messaging[1]/ eb:UserMessage[1]/ eb:MessageProperties/ eb:Property [@name="originalSender"]/ text()	//sh:StandardBusinessDocumentHeader/ sh:Sender/ sh:Identifier/ text()
//eb:Messaging[1]/ eb:UserMessage[1]/ eb:MessageProperties/ eb:Property [@name="originalSender"]/ @type	//sh:StandardBusinessDocumentHeader/ sh:Sender/ sh:Identifier/ @Authority
//eb:Messaging[1]/ eb:UserMessage[1]/ eb:MessageProperties/ eb:Property [@name="finalRecipient"]/text()	//sh:StandardBusinessDocumentHeader/ sh:Receiver/ sh:Identifier/ text()
//eb:Messaging[1]/ eb:UserMessage[1]/ eb:MessageProperties/ eb:Property [@name="finalRecipient"]/@type	//sh:StandardBusinessDocumentHeader/ sh:Receiver/ sh:Identifier/@Authority

This mapping applies both in situations where the SBDH is a separate payload and in situations where it is an integral part of the leading business document.

If the exchange from the original sender (C1) to the sending Access Point (C2) and/or the exchange from the receiving Access Point (C3) to the final recipient (C4) is also based on AS4 or other messaging protocols that support exchanging multiple payloads using *multipart/related* MIME containers (such as ebMS2 or AS2), then all message payloads including the SBDH part and any cross-references between them based on Content-ID headers can be forwarded without loss of information.

Note that it is NOT REQUIRED to use the SBDH Profile Enhancement in conjunction with the Four Corner Topology Profile Enhancement. It MAY also be used in point-to-point exchanges.

4.2.7. AS4 Header and SBDH Comparison

The AS4 header is the ebMS3 **eb:Messaging** header which is part of the ebMS3 SOAP message. It is not a message payload and it is processed by the AS4 Message Service Handler. The format and content of the AS4 user message header are similar to the header structure defined in the earlier ebMS 2.0 standard. The header allows:

- Routing or delivering messages to specific back-end applications using delivery criteria;
- Monitoring business activity with specific partners, services, or business processes;
- Tracking messages based on AS4 headers only, and in a payload-agnostic fashion.

The following table provides a comparison between the AS4 Messaging Header and the SBDH, showing their functional similarity:

AS4 Header	SBDH
The AS4 **eb:PartyInfo** group contains information about the **eb:From** and **eb:To** parties. When used with the Four Corner Enhancement, they identify corners 2 and 3.	The corresponding SBDH elements are the **sh:Sender** and **sh:Receiver** elements. When used with the Four Corner Enhancement, they identify corners 1 and 4.
The AS4 **eb:CollaborationInfo** group contains an optional **eb:AgreementRef** and mandatory **eb:Service**, **eb:Action** and **eb:ConversationId** elements.	The optional **sh:BusinessScope** group in the SBDH and the related **sh:BusinessScope** schema provide the elements **sh:BusinessServiceName** and **sh:ServiceTransaction** that have a similar purpose.
The optional **eb:MessageProperties** group contains a series of arbitrary name/value properties.	SBDH has a similar extensibility mechanism based on XML schema type substitution.
The AS4 **eb:PayloadInfo** group contains information about all message payload parts. The payloads themselves are stored in separate MIME parts in the AS4 MIME message and referenced via the *href* attribute.	In SBDH, the **sh:Manifest** group is used for (non-XML) attachments. The SBDH itself can be used standalone or as part of a standard business document, i.e. an XML payload. Attachments can be in separate MIME parts as is the case in AS4.

As the AS4 Header provides most SBDH features, the SBDH in many cases is not be needed.

4.3. Dynamic Receiver

4.3.1. Introduction

In traditional B2B data exchange, parties interact with a known set of counterparties. Messaging implementations allow users to configure partners and message exchange parameters for these partners, such as signing and encryption certificates and endpoint URIs. Users update these configurations as partners are added or removed or any configuration parameter values change.

This optional Profile Enhancement relaxes this requirement for incoming messages and allows parties to configure their AS4 MSH to receive user messages from Sender parties that have not been registered in the Receiving MSH and for which the party identifier and signing certificate have not been pre-shared between Sender and Receiver. For party authentication and authorization, it specifies an X.509 PKI-based mechanism.

4.3.2. Unregistered Sender Party

When using a Push transport channel binding (as is always the case in the current version of eDelivery AS4 Common Profile), in a One Way exchange the Initiator Party is the Sender Party and the Responder Party is the Receiver Party. In ebMS3, the **PMode.Initiator.Party** parameter is stated to be optional if the **PMode.Responder.Party** parameter is specified [EBMS3]. The **PMode.Responder.Party** parameter is always configured for a P-Mode for One Way incoming user messages deployed at the Receiver party MSH.

In the Dynamic Receiver Profile Enhancement, for a One Way exchange the **PMode.Initiator.Party** parameter MUST NOT be set in the Receiver MSH for P-Modes that are to be used to dynamically process messages from unregistered Senders. This mechanism is needed to allow a Receiving MSH to handle One Way messages from multiple Sender Parties using a single P-Mode or P-Mode template. Implementations MAY have additional mechanisms, out of scope for this specification, to configure that a P-Mode is to be used, or is not be used, dynamically with unregistered parties.

Section 4.3.5 extends this feature to Two Way exchanges.

4.3.3. X.509 PKI-based Authentication and Authorization

In ebMS3, it is stated that the **PMode[].Security.X509.Signature.Certificate** "*identifies the public certificate to use when verifying signed data*". In the case of unregistered senders, the leaf signing certificate of the Sender Party may not be available to the Receiver Party before the first message that uses the certificate is transmitted to it. Therefore, this Profile Enhancement requires the Receiver MSH to instead configure:

- One or more Certification Authority root certificates that are considered trust anchors for signed messages exchanged under control of the P-Mode.
- Zero or more Certificate Policies that apply to the signing certificate.

As the receiver MAY not have pre-configured the signing leaf certificate, a *BinarySecurityToken* token reference MUST be used to reference the signing certificate.

A Receiving MSH that implements this Profile Enhancement, when processing a message from an unregistered Sender Party under control of a Dynamic Receiver P-Mode, MUST verify the following constraints:

- The presented leaf signing certificate MUST chain to one of the specified trust anchors.
- Any policy and issuing Certification Authorities involved in issuing the presented signing certificate MUST implement the specified certificate policies, if any are specified.
- Subject identity fields in the presented certificate MUST match the presented values in the **eb:Messaging/eb:UserMessage/eb:PartyInfo/eb:From/eb:PartyId** structure in the AS4 message. Specific constraints are out of scope for this specification but SHOULD be set by communities using this Profile Enhancement. For example, the Common Name or Organization Identifier field MAY be required to have the same value as the **eb:PartyId** element.

Authentication of the message fails if any of these constraints are not met, in which case the Receiving MSH MUST NOT accept the AS4 message.

Note that these requirements may not (or not just) be Usage Profile conventions that users deploying AS4 implementations are expected to observe, but may require either specific built-in functionality in the AS4 MSH, or some separate mechanism that provides just-in-time updates of the AS4 configuration.

Also note that this Profile Enhancement does not preclude implementations from providing additional authorization checks (for example, using white-listing or black-listing at party or service level) to control which parties can and cannot send messages to them. However, any such mechanisms are out of scope for this specification.

4.3.4. P-Mode Parameters

For One Way exchanges, the following P-Mode parameters are affected by this Profile Enhancement:

Processing Mode Parameter	Value in the eDelivery AS4 Dynamic Receiver Enhancement

Processing Mode Parameter	Value in the eDelivery AS4 Dynamic Receiver Enhancement
PMode.Initiator.Party	No fixed value is specified for party identifier and identifier type in the Receiver MSH. The Receiver MSH is configured to accept messages with arbitrary values in the **eb:Messaging/eb:UserMessage/eb:PartyInfo/eb:From/eb:PartyId**, provided they match corresponding subject fields in the presented certificate.
PMode[].Security.X509.Signature.Certificate	No requirement to specify a specific end entity certificate required in the Receiver MSH. Requirement instead to configure trust anchors and optionally certificate policies.

These parameters are P-Mode parameters that can be set for individual P-Modes. A conformant deployed MSH MAY be a Dynamic Receiver for some exchanges and not for others.

4.3.5. Dynamic Receiver in Two Way Exchanges

A Two Way exchange consists of two legs (a request leg and a response leg). The party that is the Sender in the first (request) leg is the Receiver in the second (response) leg and vice versa. For Two Way exchanges, the Receiver MSH that implements the Dynamic Receiver Profile Enhancement and that is the Responding Party processes the first leg as it processes a One Way message.

The processing by an MSH that is the Receiver MSH for messages received on the second leg of Two Way exchange is similar to processing One Way exchanges, except that in the second leg of a Two Way exchange the Receiving MSH is the Initiator and the Sender Party corresponds to the **PMode.Responder.Party** parameter. The **PMode.Responder.Party** parameter is optional if the **PMode.Initiator.Party** parameter is specified [EBMS3]. The affected P-Mode Parameters are therefore as specified in section 4.3.4, except that the **PMode.Responder.Party** is affected instead of the **PMode.Initiator.Party**.

Note that when an MSH receives a second leg response message from another MSH, it will (by the definition of Two Way exchanges) have previously sent a message to that other MSH. It MAY have used the Dynamic Sender Profile Enhancement and accessed a discovery infrastructure to send the first leg request. So by the time that previously unknown Responder MSH returns a response message, it may have become a registered Sender.

4.3.6. Relation to other Profile Enhancements

The separate Dynamic Sender Profile Enhancement is similar to this Dynamic Receiver Profile, but is concerned with outgoing messages, whereas this Dynamic Receiver Enhancement is concerned with incoming messages. The two Profile Enhancements are complementary and MAY be used in a single AS4 deployment. However, this is NOT REQUIRED.

This profile MAY be used in conjunction with the Four Corner Topology Profile Enhancement by an Access Point acting, as a C3, as a receiver of AS4 messages. However, this Profile Enhancement MAY also be used in point-to-point exchanges.

This Profile Enhancement supports exchanges with or without SBDH.

4.3.7. Sample Use Case

As an example, the eDelivery AS4 Dynamic Receiver Profile Enhancement could be used in a star topology, in which one party acts as a central hub to receive data from a large (and possibly dynamic) number of counterparties, which do not communicate directly with each other. In this case, all counterparties could have a single, fixed configuration for communication with the hub, not using any of the four Profile Enhancements defined in this specification. The hub could also have a single configuration, taking advantage of the Dynamic Receiver Profile Enhancement to simplify the configuration of its AS4 MSH, while still accepting messages from multiple counterparties.

4.3.8. Limitations

While this Profile Enhancement addresses the requirements of some user communities, it has some important limitations:

- The requirement to adopt a dedicated PKI may not be acceptable for a particular community.
- The initial and ongoing operational costs of a dedicated PKI may be prohibitive.

- Many practical eDelivery scenarios require some ahead-of-time sharing of party identifiers, network security configuration data, IP addresses or address ranges. Whatever secure exchange mechanism is used to pre-share and agree on these parameters could also be used to share certificates, obviating the need for dynamic handling of signing certificates.
- The approach is not compatible with a potential future Pull Profile Enhancement, as it assumes that a Receiving MSH is always a Responding MSH.
- The license structure of commercial messaging software is very often linked to the number of partners a customer wants to exchange messages with, with higher license fees for larger (or unlimited) numbers of partners. In this Profile Enhancement, the number of communication partners is not known.

4.4. Dynamic Sender

4.4.1. Introduction

In traditional B2B data exchange, parties interact with a known set of counterparties. Messaging implementations allow users to configure partners and message exchange parameters for these partners, such as signing and encryption certificates and endpoint URIs. Users can update these configurations, as and when partners are added or removed or their configurations change.

This optional Profile Enhancement relaxes this requirement for outgoing messages and allows parties to configure their AS4 MSH to send user messages to Receiver parties that have not been pre-configured. For these parties, the party identifier and AS4 protocol parameters (such as the party's encryption certificate and the address of its AS4 server endpoint) are not registered in the Sending MSH. The Profile Enhancement provides a mechanism by which P-Modes can be created dynamically and deployed on an ad hoc basis, by instantiating templates using additional parameters supplied by the Producer and data retrieved using queries on a discovery infrastructure.

This Profile Enhancement supports two modes of operation:

- When used in combination with the Four Corner Topology Profile Enhancement, in a One Way exchange the **PMode.Responder.Party** (which corresponds to the C3) is not known but the *finalRecipient* ebMS3 **Property** (which corresponds to the C4) is provided by the Producer. In this case, the **PMode.Responder.Party** is determined dynamically, in addition to the Receiver MSH specific AS4 protocol parameters.
- When used for Point-to-Point communication, in a One Way exchange the **PMode.Responder.Party** is set directly by the Producer. Additional Receiver specific AS4 protocol parameters are determined dynamically for this party.

Note that a conformant deployed MSH MAY be a Dynamic Sender for some services and not for others.

4.4.2. Dynamic P-Modes and Discovery

This Profile Enhancement defines an ability of a Sending AS4 MSH, or a communication system that includes an AS4 MSH and exposes a similar **Submit** operation to Producers, to dynamically instantiate a P-Mode for a submitted outbound message using a *P-Mode template*, a number of input parameters and information obtained from a discovery infrastructure. The term *P-Mode template* is used informally to refer to an abstract incomplete AS4 Processing Mode that needs to be extended with some additional parameters to be ready for actual data exchange.

- One or more P-Mode templates MUST be deployed in the AS4 MSH for Dynamic Sender use that have preset values for the **PMode.Initiator.Party**, **PMode.Initiator.Role**, **PMode.Responder.Role**, **PMode[].BusinessInfo.Service** and/or **PMode[].BusinessInfo.Action** parameters. These templates MUST NOT have values for **PMode.Responder.Party**, **PMode[].Security.X509.Encryption.Certificate**, and **PMode[].Protocol.Address**. The **PMode[].Security.X509.Encryption.Certificate** and **PMode[][s].Security.X509.Signature.Certificate** MAY be used to specify constraints on trust anchors or certificate policies but MUST NOT specify any specific leaf certificate.
- If a message is submitted and matched to a P-Mode template that uses the Dynamic Sender Profile Enhancement, a request is issued to the discovery infrastructure to obtain additional information to fully instantiate the P-Mode. This request takes as input parameters:
  - The parameters specified in the P-Mode template. (This is needed because different configurations MAY be discovered depending on the supplied values, for example for different services or actions).
  - If the P-Mode template relates to a point-to-point AS4 exchange, the expected **PMode.Responder.Party** value.
  - If the P-Mode template relates to a Four Corner Topology configuration, **Party** identification for the **finalRecipient**. This will be used as value for the **finalRecipient** ebMS3 **Property** and, in addition, will be used to determine the **PMode.Responder.Party**.
- A successful response from the discovery infrastructure has matching values for the input parameters, and provides values for additional parameters and leaf certificates for the Receiver.
- If the discovery infrastructure successfully returns:

- The **Submit** operation succeeds.
- The P-Mode MUST be deployed in the AS4 MSH and the submitted AS4 message MUST be sent using this P-Mode.
- The deployed P-Mode MUST be used to process receipt or error signals from the Responder MSH.

- If the request to the discovery infrastructure is unsuccessful:
  - The **Submit** operation fails.

Note that, like the concept of Processing Modes in ebMS3, this specification uses the term P-Mode *template* as an abstract concept only. It does not intend to constrain any particular implementation options for AS4 implementations.

The Dynamic Sender Profile Enhancement MAY be implemented in a layer between Producer and the AS4 MSH. This is possible as the eDelivery AS4 Common Profile, section 3.2.7, REQUIRES implementations to provide an API to create and deploy P-Modes. This API can be used to deploy a P-Mode that has been constructed using information obtained from the discovery infrastructure. This allows any AS4 implementation that supports the eDelivery AS4 Common Profile to implement the Profile Enhancement. Alternatively, the functionality of the Profile Enhancement MAY be implemented internally in the AS4 MSH directly.

4.4.3. Discovery Infrastructure Requirements

For the eDelivery AS4 Common Profile, the Discovery Infrastructure MUST return information for the following parameters, for a One Way Exchange.

Processing Mode Parameter	Value in the eDelivery AS4 Dynamic Sender Enhancement
PMode.Responder.Party	When used in combination with the Four Corner Topology Profile, this identifies the Party offering the C3 AS4 MSH, determined by using the C4 party identifier as search key.
PMode[].Protocol.Address	The HTTPS URL of the Receiving AS4 MSH.
PMode[].Security.X509.Encryption.Certificate	This is the discovered leaf certificate that Sender MUST use to encrypt the AS4 message payloads. The P-Mode template MAY configure trust anchors and certificate policies that the discovered certificate MUST meet.
PMode[][s].Security.X509.Signature.Certificate	The discovered X.509 certificate which the Responding MSH SHOULD use to sign AS4 receipts or errors. The P-Mode template MSH MAY configure trust anchors and certificate policies that the discovered certificate MUST meet.

This Profile Enhancement is independent of any specific discovery infrastructure, as long as the infrastructure is able to meet the specified functional requirements.

4.4.4. Dynamic Sender in Two Way Exchanges

In a Two Way exchange, the Dynamic Sender Profile Enhancement applies to the first leg as specified in the preceding sections. For the second leg, which in the eDelivery Common Profile is a separate Push exchange, the reverse process is very similar: the Consumer submits a response message to the Responder MSH (or to any component that includes the Responder MSH responsible for Dynamic Sender processing), which among others:

- includes a **eb:RefToMessageId** set to the **eb:MessageId** of the first leg request message;
- reverses the **eb:From** and **eb:To** party identifiers and roles;
- sets the **eb:Service** and **eb:Action** for the response;
- in a Four Corner Exchange, reverses the **eb:originalSender** and **eb:finalRecipient** parameters.

The discovery infrastructure is used to determine the **PMode[2].Protocol.Address**, **PMode[2].Security.X509.Encryption.Certificate** and **PMode[2][s].Security.X509.Signature.Certificate** parameters.

4.4.5. Relation to other Profile Enhancements

The separate Dynamic Receiver Profile Enhancement is similar to this Dynamic Sender Profile Enhancement, but is concerned with incoming messages, whereas this Dynamic Sender Profile Enhancement is concerned with outgoing messages. The two Profile Enhancements are complementary and MAY be used in a single AS4 deployment.

If the discovery infrastructure does not support discovery of the X.509 certificate that the Responder MSH will use to sign AS4 receipts or errors, the Initiating MSH SHOULD implement the Dynamic Receiver Profile Enhancement to validate the AS4 signal and the signing certificate used.

This profile MAY be used in conjunction with the Four Corner Topology Profile Enhancement by an Access Point acting, as a C2, as a sender of AS4 messages for discovery of the C3 for a C4 final recipient. However, this Profile Enhancement MAY also be used in point-to-point exchanges.

This Profile Enhancement supports exchanges with or without SBDH as specified in the Standard Business Document Header (SBDH) Profile Enhancement.

4.4.6. Sample Use Case

The Dynamic Sender Profile Enhancement could be implemented by a service provider in an open network of service providers that offers a discovery infrastructure. Customers of the service provider can submit messages that are to be sent to any party for which the receiver service provider and a Receiving MSH AS4 communication configuration can be discovered.

4.4.7. Limitations

While this Profile Enhancement addresses the requirements of some user communities, it has some important limitations:

- Many practical eDelivery scenarios require some ahead-of-time sharing of party identifiers, network security configuration data, IP addresses or address ranges. Whatever secure exchange mechanism is used to pre-share and agree on these parameters could also be used to share ahead-of-time the parameters that the Dynamic Sender enhancement provides just-in-time.
- The approach is not compatible with a potential future Pull Profile Enhancement, as it assumes that a Sending MSH is always an Initiating MSH.
- The license structure of commercial messaging software is very often linked to the number of partners a customer wants to exchange messages with, with higher license fees for larger (or unlimited) numbers of partners. In this Profile Enhancement, the number of communication partners is not known.

5. Example

The following (non-normative, edited) example contains an AS4 message from a Seller to a Buyer in an e-procurement scenario involving the exchange of an XML order response document. It implements the Common Profile and the Four Corner Typology Profile Enhancement. In the example, both the C1 original sender, the C4 final recipient and intermediate C2 and C3 parties identified are using GS1 GLN numbers encoded as ebCore Party Identifiers.

Utvid kilde

6. Conformance

This section defines six Conformance Clauses.

6.1. eDelivery AS4 Common Profile Conformance Clause

In order to confom to the eDelivery AS4 Common Profile, an implementation MUST conform to all normative statements and requirements in section 3. In particular, it MUST:

- Support the extended subset of the feature of the AS4 ebHandler Profile specified in section 3.2.
- Support the AS4 Additional Features Payload Compression and Reception Awareness and Duplicate Elimination, as specified in section 3.3.
- Support the Usage Rules defined in section 5.1 of [AS4] and the Common Usage Profile in section 3.4.
- Support the P-Mode parameters as specified in section 3.5.
- Support the WS-I conformance requirements as specified in section 2.1 of [AS4].

This Conformance Clause is adapted from the AS4 ebHandler Conformance Clause, specified in section 6.1 of [AS4].

6.2. eDelivery AS4 Four Corner Topology Conformance Clause

In order to conform to the eDelivery AS4 Four Corner Conformance Clause, an implementation MUST:

- Conform to the eDelivery AS4 Common Profile Conformance Clause in section 6.1.
- Conform to all normative statements and requirements in section 4.1.

6.3. eDelivery AS4 SBDH Conformance Clause

In order to conform to the eDelivery AS4 Four Corner Conformance Clause, an implementation MUST:

- Conform to the eDelivery AS4 Common Profile Conformance Clause in section 6.1.
- Conform to all normative statements and requirements in section 4.2.

6.4. eDelivery AS4 Dynamic Receiver Conformance Clause

In order to conform to the eDelivery AS4 Dynamic Receiver Conformance Clause, an implementation MUST:

- Conform to the eDelivery AS4 Common Profile Conformance Clause in section 6.1.
- Conform to all normative statements and requirements in section 4.3.

6.5. eDelivery AS4 Dynamic Sender in Point-to-Point Exchanges Conformance Clause

In order to conform to the eDelivery AS4 Dynamic Sender Conformance Clause, an implementation MUST:

- Conform to the eDelivery AS4 Common Profile Conformance Clause in section 6.1.
- Conform to all normative statements and requirements in section 4.4 related to Point-to-Point exchanges.

6.6. eDelivery AS4 Dynamic Sender in Four Corner Exchanges Conformance Clause

In order to conform to the eDelivery AS4 Dynamic Sender Conformance Clause, an implementation MUST:

- Conform to the eDelivery AS4 Common Profile Conformance Clause in section 6.1.
- Conform to all normative statements and requirements in section 4.1.
- Cconform to all normative statements and requirements in section 4.4 related to Four Corner exchanges.

7. Ownership

The AS4 Profile of ebMS 3.0 Version 1.0 technical specification is Copyright © OASIS Open 2013. All Rights Reserved. The AS4 Profile of ebMS 3.0 Version 1.0 technical specification is created by the OASIS ebXML Messaging Services Technical Committee which operates under the RF on Limited Terms Mode of the OASIS IPR Policy.

AS4 is based on the OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features OASIS Standard, which is *Copyright © OASIS ® 1993–2007. All Rights Reserved.* The ebMS 3.0 Standard uses the SOAP protocol. The IPR declaration of SOAP submitters to W3C is available from http://www.w3.org/Submission/2000/05/.

Parts of the implementation guidelines in this specification are derived, with permission, from parts of the ENTSOG AS4 Profile for TSOs. ENTSOG can be contacted at http://www.entsog.eu/publications/data-exchange.

Parts of the implementation guidelines in this specification, in particular the Four Corner Enhancement, are derived, with permission, from e-CODEX specifications [ECODEXD5.11] . E-CODEX can be contacted as follows:

Ministry of Justice NRW, Martin-Luther-Platz 40 - 40212 Düsseldorf (GERMANY).

All other content of this specification, up to version 1.11, was created in the former EU e-SENS project. The e-SENS project completed in March 2017. The EU e-SENS project formally transferred ownership of its specifications to the European Commission, which accepted it for further maintenance in the context of the CEF Programme.

The Dynamic Receiver and Dynamic Sender Profile Enhancements introduced in section 1.13 are derived from the e-SENS SMP implementation guidelines which are derived from PEPPOL specifications.

Information on governance and procedures for eDelivery is available from Governance and Procedures.

8. References

[AES] Advanced Encryption Standard. FIPS 197. NIST, November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[AP] WS-I Attachments Profile Version 1.0. Final Material. 2006-04-20. http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html

[ASiC] ETSI TS 102 918 V1.1.1 (2011-04) Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC) http://www.etsi.org/deliver/etsi_ts/102900_102999/102918/01.01.01_60/ts_102918v010101p.pdf

[AS4] AS4 Profile of ebMS 3.0 Version 1.0. OASIS Standard, 23 January 2013. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/profiles/AS4-profile/v1.0/

[BP20] Basic Profile Version 2.0. OASIS Committee Specification. http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/BasicProfile-v2.0.pdf

[BSITLS] Mindeststandard des BSI nach § 8 Abs. 1 Satz 1 BSIG für den Einsatz des SSL/TLS-Protokolls in der Bundesverwaltung. Bundesamt für Sicherheit in der Informationstechnik (BSI). Bonn, 08 Oktober 2013. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Mindeststandards/Mindeststandard_BSI_TLS_1_2_Version_1_0.pdf

[eDelivery-EBCORE] eDelivery ebCore Party Id Specification. eDelivery ebCore Party Id

[eDelivery-SMP] eDelivery SMP Specification. https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eDelivery+SMP

[EBCOREP] OASIS ebCore Party Id Type Technical Specification Version 1.0. OASIS Committee Specification, 28 September 2010, http://docs.oasis-open.org/ebcore/PartyIdType/v1.0/PartyIdType-1.0.odt

[EBMS3] OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features. OASIS Standard. 1 October 2007. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/os/

[EBMS3P2] OASIS ebXML Messaging Services Version 3.0: Part 2, Advanced Features. May 2011. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/part2/201004/ebms-v3-part2.pdf.

[EBERRATA] OASIS ebXML Messaging TC Issue Tracker https://tools.oasis-open.org/issues/browse/EBXMLMSG.

[ECODEXD5.11] e-CODEX D5.11: Concept of Implementation. D5.11 Concept of Implementation v2

[ENISAAKSP] Algorithms, Key Sizes and Parameters Report 2013 recommendations version 1.0 – October 2013. ENISA. http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report

[ENTSOGAS4] ENTSOG AS4 Usage Profile. https://entsog.eu/publications/data-exchange#AS4-USAGE-PROFILE

[GLN] GS1 Global Location Number (GLN). http://www.gs1.org/barcodes/technical/idkeys/gln

[OSSLTLS] OpenSSL TLS 1.2 Cipher Suites. http://www.openssl.org/docs/apps/ciphers.html#TLS_v1_2_cipher_suites

[RFC1952] *GZIP file format specification version 4.3*. IETF RFC. May 1996, http://tools.ietf.org/html/rfc1952

[RFC2119] A. Ramos. Key words for use in RFCs to Indicate Requirement Levels. IETF RFC 2119. January 1998. http://www.ietf.org/rfc/rfc2119.txt

[RFC2392] Content-ID and Message-ID Uniform Resource Locators http://www.faqs.org/rfcs/rfc2392.html

[RFC5246] T. Dierks et al. The Transport Layer Security (TLS) Protocol Version 1.2. IETF RFC 5246. August 2008. http://tools.ietf.org/html/rfc5246

[RFC6176] S. Turner et al.Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176. March 2011. http://tools.ietf.org/html/rfc6176

[SBDH] UN/CEFACT ATG, Standard Business Document Header (SBDH). https://www.gs1.org/standard-business-document-header-sbdh

[SECCONTROLS] Security Controls Guidance. https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Security+Controls+guidance

[SOAP12] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C Recommendation. April 2007. http://www.w3.org/TR/soap12-part1/

[TLSSP] Transport Layer Security (TLS) Parameters. Last Updated 2017-08-14. http://www.iana.org/assignments/tls-parameters/tls-parameters.xml#tls-parameters-4

[WSSSMS] OASIS Web Services Security: SOAP Message Security Version 1.1.1. OASIS Standard, May 2012. http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SOAPMessageSecurity-v1.1.1.doc

[WSSSWA] OASIS Web Services Security: Web Services Security SOAP Message with Attachments (SwA) Profile Version 1.1.1. OASIS Standard, May 2012. http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SwAProfile-v1.1.1.doc

[WSSX509] OASIS Web Services Security: Web Services Security X.509 Certificate Token Profile Version 1.1.1. OASIS Standard, May 2012. http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-x509TokenProfile-v1.1.1.doc

[XMLDSIG] XML Signature Syntax and Processing (Second Edition). W3C Recommendation 10 June 2008. https://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/

[XML10] *Extensible Markup Language (XML) 1.0.* W3C Recommendation 26 November 2008, http://www.w3.org/TR/REC-xml/

[XMLENC] XML Encryption Syntax and Processing. W3C Recommendation 10 December 2002. http://www.w3.org/TR/xmlenc-core/

[XMLENC1] XML Encryption Syntax and Processing Version 1.1. W3C Recommendation 11 April 2013. http://www.w3.org/TR/xmlenc-core1/

9. Contributors

The CEF eDelivery team is in charge of maintaining the current version of the specifications.

Details about contributors of previous versions of this specification can be consulted on the e-SENS website.

10. History

Ver.	Date	Changes made	Modified By
1.12	18.okt.2017	Minor update as part of the migration of the specification to CEF Digital, not requiring external review. Layout changed to match CEF Digital template Content copied from e-SENS architecture, with the following editorial changes and bibliographic updates: • Set the version number to 1.12. • Switched from e-SENS specification metadata to the CEF eDelivery specification metadata: Publication Date, Obsoletes, Obsoleted by, • Removed the Standardization and Sustainability Assessment section. • Updated the Contributor section, historical contribution information is left to the last referenced e-SENS project version. • Updated the Ownership section to include the handover of the ownership of content from the e-SENS project to CEF Digital. • Started with a clean history table (the e-SENS history is still available in the 1.11 version). • Updated internal links to SMP to point to the CEF versions. • Removed hyperlinks from some URIs defined in ebMS3 that are identifiers only and not hyperlinks and caused HTTP 404 errors. • Updated the BDXL and SMP bibliographic references to reflect their current OASIS Standard status and to use their recommended citation formats. • Removed e-SENS bibliographic references to deliverables D6.1 and D3.2, which were not referenced. • Updated the reference for Domibus and update text to reflect its transfer from the e-CODEX project to the European Commission. • Updated the reference for the ENTSOG AS4 Usage Profile. • Updated the bibliographic reference to the e-SENS e-Confirmation pilot to point to the public e-SENS site. • Removed the bibliographic reference of the e-SENS e-Invoicing pilot, as it was not referenced. • Updated the reference for e-CODEX use of message properties for end entity addressing to the public D5.11. • Added the URL to XML Signature 1.1, 2nd edition. • Renamed the section "Specifications" to "Base Specification". • Changed "PR" to "e-SENS".	Gianmarco Piva, Pim van der Eijk
1.13	30.mai.2018	Restructuring and modularization Introduction and General: • Version updated to 1.13	Pim van der Eijk

Ver.	Date	Changes made	Modified By
		• Moved the introductory description of ebMS3/AS4 features and benefits to the "Base Specifications" section. • Renamed to eDelivery AS4 • Separate Common Profile and Profile Enhancements. • In the context of AS4, no longer using the term Gateway, more consistent terminology. • Start using RFC 2119 upper case consistently. • New "Conformance" section. Similar to OASIS specifications, listing clauses that implementations can claim conformance to. • Using "party" instead of "organization". • More consistent formatting, bold for elements, italic for attributes and properties. • Use the notation PMode[] instead of PMode[1] to also cover Two Way exchanges Common Profile • In Common Profile, separate ebHandler feature review from AS4 additional features. • In feature set, requirement that order of payload parts can be controlled in submission and is provided in delivery. This is needed to distinguish the primary business document payload from other payloads. • In feature set, added a note that CompressionType is used by AS4 Compression, so it's not to be used in the Submit/Delivery interfaces. Also made explicit that compression is used in the Common Profile. • In feature set, required product support for IPv6 in new subsection networking. • In feature set, defined requirement to be able to define a P-Mode for the test service • In feature set, note that part properties are mandatory. • In new additional Features subsection, added subsections for Compression and Reliable Messaging / Reception Awareness. Explained that these are the only two additional features used. • In common usage profile, requirement to have only one **From**|**To**/**PartyId** in separate usage profiling subsection. • In common usage profile, added reference to ebCore Party ID and ISO 6523 and the eDelivery profiling of this. • In common usage profile, note that deployments can choose IPv4 or IPv6. • In common usage profile, require that there is a test service for every communication pair and agreement • Better describe the test service, which impacts both implementations (products) and deployments (by users) of AS4, and how it can be used to check correct configuration of security including certificates. • In payload profile profiling, note JSON as structured format like XML. • Better explain the concept of leading business document. • Explain that the concept of "first" is linked to the **PartInfo** sequence, which is not guaranteed to be the same order as the MIME part order. Profile Enhancements, Four Corner Topology • Four Corner Topology, more explanation on submit/deliver in Four Corner exchanges, and difference to SOAP and ebMS3 intermediaries.	

Ver.	Date	Changes made	Modified By
		• Listed the P-Mode properties in a separate table. • Explanation that AS4 level Non-Repudiation does not cover all four corners. Profile Enhancements, SBDH • Reworked and restructured. • Better explained the integrated and non-integrated uses. • New subsection and table explaining identifiers and cross-references when using a standalone SBDH. • New subsection and table explaining mapping between AS4 four corner attributes and SBDH elements. • Adapted the SBDH example to have a **Manifest** element referencing the eDocument. • Better explained the (optional) link to the Four Corner Enhancement. Profile Enhancements, Dynamic Receiver • New section, covering the PKI assumptions of dynamic connections separately from the discovery aspects. • some content generalized from previous SMP profiling. • Make the configuration of trust anchors explicit. • Added note on certificate policies. • Separate from Dynamic Receiver. • Covered the Two Way MEP. Profile Enhancements, Dynamic Sender • New section, some content generalized from previous SMP profiling • Support point-to-point in addition to Four Corner topologies. • Define functional requirements that a Discovery Infrastructure must meet, without requiring any specific technology. • Separate from Dynamic Receiver. • Covered the Two Way MEP. Example: • Updated to fix some incorrect use of WS-Security. Bibliographic: • Updated [TLSSP] reference. • Added [BP20] reference. • Added [AP] reference. • Added [ASiC] reference. • Removed the [BDXL] and [SMP] references. Additions from comparison to ENTSOG: • In Common Profile feature set, clarification on use of UTF-8 and Unicode BOM for BP20 added. • In Common Profile feature set, requirement to be able to add Part Properties added. This is needed to support users that want to specify part properties. • In Common Profile feature set, added requirement to be able to set the **AgreementRef** using the **PMode.Agreement** parameter. • In Common Profile feature set, added requirement to be able to manage PModes and Certificate configuration in an MSH using an API, in new	

Ver.	Date	Changes made	Modified By
		"Configuration Management" section. • In Common Profile feature set, added requirement that all ebMS3 headers linked to PMode parameters are used to determine the PMode to use. Updates after CEF team meeting 21.11.2017: • Expanded description and definition of point-to-point and four corner exchanges. • Added explanation that Producer and Sender (Receiver and Consumer) are not the same as C1 and C2 (C3/C4) corners and are not intermediaries. • Described multi-tenancy concept. • The PMode[].ErrorHandling.Report.MissingReceiptNotifyProducer was listed in 3.2.5 but not in section 3.5. • The PMode[].ErrorHandling.Report.ProcessErrorNotifyProducer was missing altogether. • Added more explanation and discussion on generation and reporting of DeliveryFailure errors. • Some minor editorial updates. • Separated the Dynamic Sender functionality in two conformance clauses, one for point to point exchanges, one for four corner exchanges. • Note on BinarySecurityToken reference recommended and mandatory for Dynamic Receiver. Updated in December after CEF team meeting 15.12.2017: • Renamed from e-SENS AS4 to eDelivery AS4 • Added PEPPOL to Ownership section for Dynamic Sender / Dynamic Receiver. • Added explanation of RFC2119 keywords. • Updated SBDH URL due to GS1 site change • More consistent terminology: document, data, payload • Remove reference to e-Interaction and sample use case. • Added reminder to 7.6 of ebMS3 Core that encryption follows signing. • More consistent terminology: conformant instead of compliant, conformant, conforming. • More consistent terminology: implementation instead of product. • Consistent upper/lower case for various terms. • Correct use of P-Mode and PMode as per ebMS3 Core. • Updated sample message to use BST token reference for signature. • Added link to security controls document. • Explanation of P-Mode notation. • New section 2.3, Notation. • Section 3.4.6, restructured to allow any combination of payloads. If there is a leading payload, it goes first. • More consistent use of italic and bold font • Other editorial. • New sections 4.3.8 and 4.4.7 Mid January updates:	

Ver.	Date	Changes made	Modified By
		• Made the three key transport related algorithms mandatory instead of recommended. The reason is that interoperability issues may occur if they are just recommeded. We also verified that all currently conformant implementations support the recommendations, so no existing implementation is affected. • Made support for TLS 1.2 mandatory and removed an ambiguity about SSL 3.0. • In section 3.3.1, explain that compression also optimizes security processing (less data to sign/encrypt). • In XPath expressions or expression fragments, made sure to use namespace prefixes consistently. • Added a note that WS-ReliableMessaging and WS-Reliability are not used. • Fixed XPaths in 4.2.3. • UUID are recommended for message identifiers, but must still be RFC 2822 compliant, so they can be used in the id-left. • In 3.4.5, configuration is required to follow non-repudiation policy. • Fixed bibliographic reference to eDelivery SMP and eDelivery ebCore Party Id. • Fixed a broken link. • Made the Configuration Management API recommended instead of mandatory, as we don't know if all currently conformant implementations have such an API. (Note that an API is required for some enhancement and for ENTSOG). • To address https://issues.oasis-open.org/browse/EBXMLMSG-111, added a note that AS4 encryption applies to payload parts, not to the **eb:Messaging** header or any of its sub-elements. • Changed status to eDelivery Community Draft 23 April updates, comments from Public Review and from the eDelivery team: • Added a sentence "The AS4 option to transmit errors using asynchronous messages MUST NOT be used." • Added a sentence "The AS4 option to transmit receipts using asynchronous messages MUST NOT be used." • Changed "Parties MAY use firewalls " to "Parties SHOULD use firewalls." • Added a sentence "Therefore, the **PMode[].Reliability** parameters MUST NOT be used.", to indicate that these do not apply to AS4 reception awareness. • Clarified that the requirement to be able to set (references to) message and conversation identifiers in submission and to preserve them in delivery only applies to user messages. • In section 3.2.6, added that other key transport algorithms may be accepted on inbound messages for backward compatibility reasons. • Minor layout and formating improvements. • Fixed a broken link • **eb:RefToMessageId** is only delivered if present in the message. 7 May 2018: • Fixed a typo.	

Ver.	Date	Changes made	Modified By
		30 May 2018: • Status set to eDelivery Specification	

Details about previous versions of the specifications can be consulted on the e-SENS website.