

ebMS 2.0 Interoperability

Issues Report

Provides a list of issues resolved over the course of multiple ebMS 2.0 Interoperability Tests

December 19, 2008

Prepared & Facilitated by:
Drummond Group Inc.
www.drummondgroup.com

Table of Contents

Interoperability Issues.....	3
Resending Acknowledgments for Once-and-Only-Once Delivery.....	3
Use of SOAPAction header with the Apache Web Server.....	3
Use of ISO101026PADDING algorithm for XML Encryption.....	5
Empty responses must be HTTP 204 replies with an empty body.....	5
Addendum 4Q06.....	5
Synchronous Messages and the SyncReply element.....	6
SOAP Action HTTP Header in Sync Reply responses.....	6
The SOAPAction value must surrounded by double quotes.....	6
Format for Messages without payloads.....	6
SOAP Faults may be received as HTTP 200 or 500 Responses.....	6
CPPA and CPA Exchange.....	7
Role Element value.....	7
CID based start parameter in the MIME Content-Type header.....	7
MIME Multipart/Related header is case insensitive.....	7
Service and Action values.....	8
XMLDSIG Namespace declaration must be at Signature element level.....	8
XML Digital Signature KeyInfo.....	8
Payloads are not canonicalized during the digital signing process.....	8
Payloads are not canonicalized for XML-DSig and XML-Encryption.....	9
ConversationIDs must be unique over CPAId.....	9
Different Interpretations of the use of ConversationIDs.....	9
XMLSchema Instance declaration must be present.....	10
Timestamps may include fractions of seconds.....	10
References to original message MUST be in a signed Ack.....	10
MessageID and Content-ID MUST conform to MIME and include @.....	10
Error Messages should not be signed.....	10
SyncReply element may appear in a synchronous reply.....	10
Addendum 4Q06.....	11
Timestamp may differ in acknowledgments to duplicate requests.....	11
Timestamp may differ in retry messages.....	11
Content-type for XML Encryption.....	11
KeyName must be X509 Distinguished Name format.....	12
Sign first, Encrypt second.....	12
Compress first, Sign second.....	12
Compressed data will not be base64 encoded.....	12
Distinguished Names should be DNS or IP Address.....	12
Timing of Asynchronous replies versus http responses.....	13
Digital Signature Algorithm RSA vs. DSA for Signing.....	13
DSA Signature Certificate not fully support SSL Cipher Suites.....	13
Multiple inbound signed messages caused Apache XML Security Jar to Miscalculate the Hash.....	13
Common Name Must Support the Correct PrintableString Characters.....	14
About Drummond Group Inc.....	15

Interoperability Issues

During the course of interoperability tests, interoperability issues were discovered or questioned and then resolved through the debugging stage of the test. All products from a given test comply with the corresponding resolved issues. These issues are listed below to assist in resolving any supply-chain trading problem which may occur between products-with-version from this test and ebMS 2.0 products-with-version from outside the test, including backward versions of these test products.

Resending Acknowledgments for Once-and-Only-Once Delivery

When operating in Reliable Messaging mode with Once-and-Only-Once Delivery enabled, a Receiving MSH should persist (for a reasonable duration) any Ack returned to a Sending MSH. If the Receiving MSH receives a duplicate message, the persisted Ack should be resent (Section 6.5.5), but in the event the persisted Ack cannot be retrieved (or was not persisted), the Receiving MSH must regenerate and resend an “identical” Ack, which is defined as an Ack containing the same SOAP Header, Body, and Payload Container(s) (Section 6.5.6). For a regenerated Ack, the MessageID may be different than the original Ack returned.

In reviewing Section 6 of the ebMS v2 specification, it is clear that the specification intends for implementations to persist messages and acknowledgments that are exchanged using the Reliable Messaging module. This is for detecting duplicate messages and re-sending “lost” acknowledgments. Section 6.5.5 states that if an MSH receives a duplicate message, it should re-send the original acknowledgment from its persist store. However, 6.5.5 goes on to say that if the acknowledgment cannot be found in persistent storage, then the MSH should generate an **identical** acknowledgment. An identical message as defined by ebMS 2.0 is message containing the same SOAP Header, Body, and Payload Container(s). A **duplicate** message is defined by ebMS 2.0 as a message containing identical MessageID values. Therefore, regenerated acknowledgements can be *identical* and yet have a different MessageID. It should be re-iterated that it is *recommended* that implementations persist their sent messages and acknowledgments while operating in Reliable Messaging mode.

Use of SOAPAction header with the Apache Web Server

The presence of multiple SOAPAction headers in an ebMS message in combination with certain HTTP proxies can inadvertently lead to a non-conformant SOAPAction header format. The solution would be for ebMS message senders to ensure that messages contain one and only one SOAPAction header.

An ebMS v2.0 message should contain one (1) and only (1) SOAPAction header with the value and format as specified by a consensus item from the 4Q04 test round. This clarifying constraint is to ensure that the ABNF specification for the SOAPAction header (as defined in the SOAP 1.1 specification) is not violated by certain HTTP proxies (such as using the Apache Web Server in proxy mode).

During the 4Q07 Interoperability Test Event, it was discovered that if an ebMS message contained multiple SOAPAction headers

```
POST /ebxmlhandler HTTP/1.1
Connection: Keep-Alive
Host: 127.0.0.1
SOAPAction: "ebXML"
MIME-Version: 1.0
Content-Length: 2478
SOAPAction: "ebXML"
Content-Type: multipart/related; type="text/xml";boundary="----=MyMimeBoundary"
```

and an Apache Web Server was being used by the receiver as an HTTP proxy, the Apache Web Server was combining the multiple SOAPAction headers into a single SOAPAction header with a comma-separated list of values according to Section 4.2 of RFC2616. This resulted in the ebMS receiving handler seeing the Apache-modified SOAPAction headers as:

```
SOAPAction: "ebXML","ebXML"
```

The relevant text in Section 4.2 of RFC2616 says:

```
Multiple message-header fields with the same field-name MAY be present in a message if and only if the entire field-value for that header field is defined as a comma-separated list [i.e., #(values)]. It MUST be possible to combine the multiple header fields into one "field-name: field-value" pair, without changing the semantics of the message, by appending each subsequent field-value to the first, each separated by a comma. The order in which header fields with the same field-name are received is therefore significant to the interpretation of the combined field value, and thus a proxy MUST NOT change the order of these field values when a message is forwarded.
```

The first two sentences in the text above are important for interpretation. The HTTP specification does allow for multiple instances of a header to occur in an HTTP message, but only under the stipulation that the header values can be described as a list of comma-separated values and if that combination of values doesn't change the semantics of the message.

The SOAP 1.1 specification and more specifically to Section 6.1.1 for clarification of SOAPAction header field in SOAP messages, the specification provides the following ABNF syntax for SOAPAction:

```
soapaction    = "SOAPAction" ":" [ <"> URI-reference <"> ]
URI-reference = <as defined in RFC 2396>
```

The ABNF syntax does not include the variable repetition operator for the URI-reference element that would specify that list of comma-separated values are permissible for this field. And since the URI-reference element is enclosed in square brackets, this equivalent to

```
*1(<"> URI-reference <">)
```

According to ABNF syntax, the *1 syntax means there should be exactly 0 or 1 field-values in the form of a URI-reference for the SOAPAction header.

Use of ISO101026PADDING algorithm for XML Encryption

During the 4Q07 Interoperability Test Event, one participant was using the DESede/CBC/ISO101026Padding algorithm in the process of constructing and sending encrypted ebMS messages during the XML Encryption Optional Profile tests. The use of this algorithm caused decryption problems with a couple of participants that either did not support this algorithm or that their decrypt cipher was initialized with only the DESede/CBC/PKCS5Padding algorithm.

Historically, it is believed that participants in previous Interoperability Test Events had used the more common DESede/CBC/PKCS5Padding algorithm. In order to ensure interoperability, the participant that was using ISO101026Padding switched to PKCS5Padding.

However, for future Interoperability Test Events, participants should be prepared to accept and successfully decrypt ebMS messages that used the ISO101026Padding algorithm during the encryption process.

Empty responses must be HTTP 204 replies with an empty body

ebMS version 2 states under section B.2.3 that “A 2xx code MUST be returned when the HTTP Posted message is successfully received by the receiving HTTP entity.” and under section B.2.4 that in the case of asynchronous messaging “a HTTP response code in the range 2xx MUST be returned when the message is received successfully”.

In practice during the current and previous test rounds interoperability issues have been found when participants did not return a simple empty HTTP reply in these situations.

The consensus has been developed that in these situations an HTTP reply with code 204 must be returned and the body of the HTTP must be empty.

There has been strong agreement and proof of interoperability over this consensus and that it is in compliance with the ebMS 2.0 specification and details of RFC2616.

Addendum 4Q06

The argument of 204 .vs. 2xx left some unresolved questions around this consensus. Another conceptual view was necessary to qualify a revelation of this consensus.

DGI Internal review revealed this issue became known due to how HTTP 1.1 persistent connections could cause resources to become unknown or defunct. Consensus items “Timing of Asynchronous replies versus http responses” and “SyncReply without expected Response” also revealed how responses could cause stale resources.

In the absence of 204 the MSH (Message Service Handler) allows the connection to remain open to receive a response/message at any point in time. Response codes other than 204 may cause ebMS to expect an incoming response with a body (message), but it may or may not arrive leaving the connection open for a prolong period of time. This issue caused implementers to question how stagnant resources should be released or removed in a none-harmful manner.

The only viable solution was to issue a 204 response code. The response means the server fulfilled the request. The 204 response channeled the Initiator (request) to abolish unnecessary resources and close the connection (transaction) safely.

This consensus corrected the issue of persistent connections and resource management over an extended period.

Synchronous Messages and the SyncReply element

The majority of tests in the Basic profile are asynchronous, reflecting DGI's view that the market perceives asynchronous messaging as scalable and appropriate for B2B messaging. A minority of tests require synchronous message and will use the ebMS SyncReply element and will assume an ebMS SyncReplyMode of mshSignalsOnly.

SOAP Action HTTP Header in Sync Reply responses

Message Handlers should tolerate synchronous replies that contain a SOAPAction header and synchronous replies that do not contain a SOAPAction header. During discussions on this issue the key points made were that ebMS 2 is not clear on this issue and that while the intention of SOAP 1.1 HTTP Binding may be that SOAPAction is present only in HTTP requests and not in HTTP replies, SOAPAction is not specifically forbidden from being present in a reply.

The SOAPAction value must surrounded by double quotes

The SOAPAction value must be "ebXML" and include the double quotation marks.

For example,

```
SOAPAction: "ebXML"
```

As opposed to,

```
SOAPAction: ebXML
```

ebMS 2 does not directly address this specific issue, but underlying specifications do.

Format for Messages without payloads

Message Handlers should tolerate both messages which have no payload(s) sent as multipart/related, and messages which have no payload(s) sent as plain SOAP format (text/xml). ebMS 2 is not completely clear on this issue, but does provide examples of both.

SOAP Faults may be received as HTTP 200 or 500 Responses

Message Handlers should tolerate SOAP Faults sent as HTTP 500 responses or sent as HTTP 200 responses, or sent as separate asynchronous posts. It has been found that

many products in the field generate SOAP Faults in the form of HTTP 500 responses which causes some application server or web server products to ignore data in the HTTP body, resulting in a special case that needs to be understood and taken into account.

CPPA and CPA Exchange

Individual participants may optionally exchange CPAs (Collaborative Partner Agreements). The exchange of CPAs is not tested. Partner agreement information is detailed in English language descriptions within the Test Plan.

Over the history of the tests, a majority of participants have utilized CPA based systems and a minority of participants have not. In effect the test rounds are indirectly testing the ability of CPA based message handlers to interoperate with both other CPA based message handlers and to interoperate with message handlers not based on CPA.

Role Element value

If the Role element is present, as contained in either ToParty or FromParty elements, its value must be agreed upon by the two testing parties. ebMS version 2.0 is unclear on how to set this value. An addendum to ebXML CPPA version 2.0 specifies that the value for the Role element should be the same as the Role/@name value in a related CPA.

CID based start parameter in the MIME Content-Type header

The start parameter of the MIME Content-Type header may be a cid (content-id) style reference. It may contain a prefix of "cid:" which should be stripped to obtain the Content-ID value.

For example,

```
SOAPAction: "ebXML"
Content-Type: Multipart/Related; type="text/xml";
    boundary="MIMEBoundary"; start="cid:myContentIDHeader"
--MIMEBoundary
Content-Type: text/xml
Content-ID: <myContentIDHeader>
```

MIME Multipart/Related header is case insensitive

The ebMS version 2.0 specification requires a MIME Content-Type: Multipart/Related header to appear as an HTTP header. Consensus is that MIME specifications require and ebMS Message Handlers must tolerate mixed case in the phrase Multipart/Related.

For example, both of these phrases are valid

"Multipart/Related"

and

"multipart/related".

Service and Action values

Basic profile test message will use the same value for Service and different values for Action. This consensus is intended to allow ease of use for participants using CPA, and to assist in ease of interoperability between participants using CPA and participants not using CPA.

Service and Action values for Industry Profiles may use industry specific values.

XMLDSIG Namespace declaration must be at Signature element level

To simplify parsing logic and assist interoperability, the attribute that declares the XML Digital Signature namespace must be at the Signature element level, for example

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmlsig#">  
...  
</ds:Signature>
```

It is not unreasonable to include this namespace declaration at the SOAP envelope level, but there appears to be an overwhelming best practice to specify it at the Signature element level. During the 3Q03 ebMS test round at least one security toolkit was found not to process XML Digital Signature without its presence at this location.

The XML Digital Signature specification lists a non-normative DTD that describes the declaration at the Signature level. The XML Digital Signature normative XML Schema however, does not describe this format. All Digital Signature examples from the ebMS 2 specification have the declaration at signature level and an informal survey of examples from the web revealed every available example placing the declaration at the signature level.

XML Digital Signature KeyInfo

The KeyInfo element as defined by XMLDSIG is optional, if present it may be ignored. In other words, the assumption is that Digital Certificates used to verify a message signature will be exchanged out of band and will be known beforehand by the message receiver.

During previous test rounds, at least one security toolkit in use forced the use of the KeyInfo element if it was present. Because of this, message senders should either not format the KeyInfo element, or make sure that the Digital Certificate passed within the KeyInfo element is the correct Digital Certificate to be used by the receiver to verify Digital Signature.

Payloads are not canonicalized during the digital signing process

ebMS states that transformations applied to payloads are "implementation dependent." This consensus allows for interoperable validation of signature digests. Some past participants

have pointed out that C14 canonicalization removes comments, which could result in security or integrity issues.

Payloads are not canonicalized for XML-DSig and XML-Encryption

The Message Payload must be treated as a simple byte stream

When combining XML Encryption with Digital Signature for use in the XML Encryption Profile tests, payloads will be treated at all times during processing as simple bytestreams, even if the payload is XML being encrypted with XML Encryption. This follows the above consensus that Payloads are not canonicalized for Signature. This consensus has implications on the way security toolkits are configured and used. Some security toolkits will attempt to canonicalize XML data by default, and must be configured to treat an XML Payload as a simple byte stream.

During previous test rounds there have been interoperability problems found in the implementation of this consensus. When processing Encryption and Signature, some security toolkits will by default attempt to canonicalize or de-canonicalize data that it recognizes as XML with XML Encryption applied. To overcome these issues, some participants had to physically configure their security toolkits (at both sender and receiver implementations) to treat the payloads as simple binary byte streams. In other words, the data was not encoded via DOM or other similar mechanisms, but was treated at all times as a simple byte array.

One specific problem that was found was that if on the receiving side the payload is considered to be XML, the security toolkit will pass the XML data to the end application with the XML prelude declaration (i.e., "<?xml version='1.0' encoding='UTF-8'?>") missing, causing the application to view the payload as invalid XML.

ConversationIDs must be unique over CPAId

ConversationID must be unique for non-long running conversations under a specific CPAId. If the ConversationID is duplicated, the implementation MAY respond with an error.

Different Interpretations of the use of ConversationIDs

During previous test rounds, discussions revealed a difference of interpretation on the meaning and use of the ConversationID element.

The ebMS v2.0 specification requires that ConversationID be present in all messages, and requires that if you implement the optional MessageOrdering feature (not tested by DGI) that ConversationID must stay the same over all ordered messages.

End users and implementers should be aware that there are different interpretations of the meaning and use of ConversationIDs and if the use of ConversationID is important its usage should be clearly explained.

XMLSchema Instance declaration must be present

In past rounds, interoperability problems were encountered if a participant's implementation did not include an XMLSchema instance declaration as below:

```
<SOAP:Envelope  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  ...
```

Under section 2.2.2 of ebMS version 2, it is strongly recommended that the declaration be present. It has been found that the absence of the declaration will cause interoperability problems with some implementations, and for the purposes of the test round the declaration is required to be present, even though in the normative specification this is only a strong recommendation.

Timestamps may include fractions of seconds

In the fourth round, at least one implementation reported problems processing fractions of a second (milliseconds). Discussion among the testing group led to the consensus that XML Schema Datetime format allows for fractions of seconds and that ebMS implementations should support its use.

References to original message MUST be in a signed Ack

This consensus was gained by a majority of past participants and is supported by ebMS 2 which states "if you support signed acknowledgments, it is required that you include references to the original message digests." A further consensus was reached that signed acknowledgments should include these References, even if the original message itself was not signed.

MessageID and Content-ID MUST conform to MIME and include @

ebMS 2 clearly requires that @ be present in ebMS MessageIDs but it is less clear that @ is required in MIME Content-ID header values. The consensus is that these ID elements should be formatted in this fashion to comply with MIME specifications, but that receivers should act liberally, and not reject a message solely based on Content-ID or MessageID not containing an @.

Error Messages should not be signed

Error Messages should be sent without digital signature, to avoid the possibility that the Error is related to signing processes. Specifically, when a signed acknowledgment is requested, and an Error message is generated in reply, that Error message should not be signed.

SyncReply element may appear in a synchronous reply

ebMS 2 does not forbid the SyncReply element from appearing in a synchronous reply. General discussion has been that there may be some use case scenarios where this is

useful and a receiving Message Handler (the MSH receiving the response) should allow SyncReply element in an HTTP reply.

Addendum 4Q06

According to CPP/A V 2.0 lines:1760-1768 says, “the SyncReply element may appear in ebMS replies except synchronous Acknowledgements when the mode is ‘mshSignalsOnly’ “

The SyncReply element must not appear in synchronous Acknowledgements. Asynchronous acknowledgements are checked by the CPA. The CPAs should not request SyncReply on an asynchronous Acknowledgement – no need to ack an ack.

Timestamp may differ in acknowledgments to duplicate requests

ebMS 2 is not completely clear that acknowledgments to duplicate requests should be exact copies of the original acknowledgement message. This consensus was reached after finding that many vendor implementations allow the Timestamp value to differ in the acknowledgments.

Timestamp may differ in retry messages

ebMS 2 states that a message retry should be a resend of “the original message” It is not clear if ebMS 2 requires that the message Timestamps cannot change. Discussion around this issue is that MessageID is the primary element used to detect duplicate messages, and using real-time timestamps (as opposed to repeating an earlier timestamp) is useful for auditing and identifying message replay attacks.

Consensus is that the Timestamp element of a message sent as a retry may differ from the Timestamp element contained in the original message.

Content-type for XML Encryption

When implementing XML Encryption for the XML Encryption Profile tests, the Content-Type MIME header value for XML Attachments can be either text/xml or application/xml. In other words, the Content-Type should have these values, even if the payload is XML encrypted with the XMLEncryption standard. Products should gracefully handle a message that uses one of these two values. This consensus is based on input from test participants and recommendations from the CDC. For example both of these phrases are acceptable:

```
--MIMEBoundary
Content-Type: application/xml
or
--MIMEBoundary
Content-type: text/xml
```

During 3Q03 ebMS Test round, this topic was discussed among participants. An alternative suggestion was to use the value application/xenc+xml. Investigation into this MIME type revealed that it has been registered, but not yet accepted formally as an official MIME type.

KeyName must be X509 Distinguished Name format

For the purposes of the XML Encryption Profile, the value of the KeyName element must be a valid X509 Distinguished name. Specifically, this name is expected to be the Distinguished Name of the Certificate containing the public key used to encrypt the symmetric key passed in the message.

This is a best practice convention used during the test to enable the lookup of the related key, and is the method prescribed by the CDC PHIN architecture. In practice, in the field, systems may sometimes choose to use “alias” values other than Distinguished Name that are defined and agreed upon by the two parties.

Sign first, Encrypt second

For the purposes of the XML Encryption Profile, participants are required to apply Digital Signature first and XML Encryption of the XML Payload second. This consensus is per CDC recommendations and is also noted in ebMS 2.0 section 4.1.4.5.

Compress first, Sign second

For the purposes of the Automotive Retail profile, participants are required to apply compression first and Digital Signature second. During the current test round, there was discussion that there are cases where signing first and compressing second make sense.

This consensus is based on discussions with STAR members who worked directly on the STAR ebMS Transport guidelines, and it is intended that this order (compress first sign second) allows for implementations where a backend system may be responsible for the compression but the message handler will be responsible for the digital signature.

Compressed data will not be base64 encoded

During the fourth round, the testing group developed a consensus that compressed data will be represented as simple binary data and would not be encoded as base64. The reasoning behind this consensus 1) base64 encoding is not needed as http is considered to be a binary safe protocol 2) base64 encoding and decoding would add unnecessary processing overhead.

This consensus was vetted with the group that developed the STAR Transport guidelines.

Distinguished Names should be DNS or IP Address

Many products in the field recommend or require that the Distinguished Name of the digital certificate used for a B2B server or Web server be the DNS or IP Address of the server. To improve interoperability, DGI recommends that participants follow this convention where possible.

Timing of Asynchronous replies versus http responses

ebMS was originally intended as a transfer protocol neutral standard. ebMS provides bindings for SMTP and HTTP, but occasionally issues come up related to HTTP that are not addressed by the ebMS binding discussion, for example ebMS does not attempt to assign a meaning to an empty HTTP response. During early test rounds an issue was discovered with the timing of synchronous HTTP responses versus asynchronous ebMS replies. In other words, due to design or network issues it is theoretically possible to receive an asynchronous ebMS level reply before receiving a synchronous HTTP response. When designing for asynchronous messaging, DGI recommends that ebMS implementations do not depend on the timing of HTTP synchronous responses versus ebMS asynchronous replies.

Digital Signature Algorithm RSA vs. DSA for Signing

DGI recommended Signature Algorithm RSA for ebMS Interops, but DSA is recommended in the ebMS V2.0 Spec. DSA signed messages failed signature validation when received by IBM's XML Security Suite (XSS4J).

XMLDsig says DSA is required but accepts RSA Signature Algorithm.

The XSS4J Toolkit is no longer offered as a product by IBM. It has been rolled into the JSR 105/106 Security Services.

To ensure interoperability among the Test Group, every one used RSA. The Test Group believes DSA and RSA are valid in the marketplace, therefore both should be supported.

Future Interops will include Test Cases for both DSA and RSA Signature Algorithms.

DSA Signature Certificate not fully support SSL Cipher Suites

Some toolkits have difficulties validating Cipher Suites (during the SSL Handshake) when the certificate has sha1dsa as the Signature Algorithm.

The solution is to use SHA-1 RSA Signature Algorithm where most suites are supported.

Multiple inbound signed messages caused Apache XML Security Jar to Miscalculate the Hash

Sender (using JSR 105: XML Digital Signature APIs) sent multiple signed message to the Recipient (using Apache XML Security Jar 1.0.5). The first message was verified successfully by the Recipient, but subsequent signed messages were rejected (signature value verification failures).

To resolve this issue, the Recipient upgraded to Apache XML Security Jar 1.3 – all subsequent signed messages were verified successfully.

Common Name Must Support the Correct PrintableString Characters

Common Name with an underscore character is an invalid character within this field. The common name field must have valid 'PrintableString' characters.

About Drummond Group Inc.

Drummond Group Inc. (DGI) is an independent, privately held company that works with software vendors, vertical industries and the standards community to drive adoption of open standards by conducting interoperability and conformance testing, publishing related strategic research and developing vertical industry strategies. Founded in 1999, DGI represents best-of-breed in the industry on linking horizontal infrastructure technologies, standards and interoperability issues with the needs of vertical industries such as retail, grocery, health care, transportation, government and automotive. For more information, please visit www.drummondgroup.com or email: info@drummondgroup.com.