Individual Submission
Internet-Draft
Intended status: Informational

Expires: June 18, 2012

K. Meadors
Drummond Group Inc.
D. Moberg
Axway, Inc.

December 22, 2011

Certificate Exchange Messaging for EDIINT
draft-meadors-certificate-exchange-14.txt

Abstract

The EDIINT AS1, AS2 and AS3 message formats do not currently contain any neutral provisions for transporting and exchanging trading partner profiles or digital certificates. EDIINT Certificate Exchange Messaging provides the format and means to effectively exchange certificates for use within trading partner relationships. The messaging consists of two types of messages, Request and Response, which allow trading partners to communicate certificates, their intended usage and their acceptance through XML. Certificates can be specified for use in digital signatures, data encryption or SSL/TLS over HTTP (HTTPS).

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 18, 2012.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction....3
1.1 Overview....3
1.2 Terminology and Key Word Convention....4
1.3 Certificate Lifecycle....5
1.4 Certificate Exchange Process....6
2. Message Processing....7
2.1 Message Structure....7
2.2 EDIINT Features Header....9
2.3 Certificate Exchanging....9
2.4 Certificate Implementation....10
2.5 CEM Response....12
3. CEM XML Schema Description....13
3.1 EDIINTCertificateExchangeRequest element....13
3.2 EDIINTCertificateExchangeResponse element....17
4. Use Case Scenario....18
5. Profile Exchange Messaging....20
6. Implementation Considerations....21
7. Future Considerations for CEM I-D....21
8. Security Considerations....21
9. IANA Considerations....22
10. References....22
10.1 Normative References....22
10.2 Informative References....23
11. Acknowledgments....23
Author's Addresses....23
Appendix....24
A.1 EDIINT Certificate Exchange XML Schema....24
A.2 Example of EDIINT Certificate Exchange Request XML....27
A.3 Example of EDIINT Certificate Exchange Response XML....28
Changes from Previous Versions....28
B.1 Updates from Version 00....28
B.2 Updates from Version 01....29
B.3 Updates from Version 02....29
B.4 Updates from Version 03....29
B.5 Updates from Version 04....29
B.6 Updates from Version 05....29
B.7 Updates from Version 06/07/08/09/10....29

1. Introduction

1.1 Overview

The growth and acceptance of EDIINT protocols, AS1, AS2 and AS3, in numerous supply-chains was due in part to the security feature which was provided. The security is not possible without the digital certificates which enable it. To maintain the level of security necessary to transmit business documentation, existing certificates must occasionally be replaced and exchanged with newer ones. The exchanging of digital certificates is unavoidable given how certificates can expire or become compromised. Complicating this is supply-chains which cannot afford to shutdown their business transactions while trading partners laboriously upload new certificates. Certificate exchange must be accomplished in a reliable and seamless format so as not to affect ongoing business transactions.

This document describes how EDIINT products may exchange public-key certificates. Since EDIINT is built upon the security provided by public-private key pairs, it is vital that implementers are able to update their trading partners with new certificates as their old certificates expire, become outdated or insecure. Certificate Exchange Messaging (CEM) described here utilizes XML data to exchange the certificate and provide information on its intended usage and acceptance within the trading partner relationship. There are two types of CEM messages. The CEM Request which presents the new certificate to be introduced into the trading partner relationship and the CEM Response which is the recipient's response to the CEM Request. CE messages can be exchanged through AS1 [AS1], AS2 [AS2] or AS3 [AS3] message transports. However, it is possible to leverage CE messaging through other transport standards besides EDIINT.

1.2 Terminology and Key Word Convention

[RFC4949] provides a glossary of Internet security terms, and several of their definitions are listed here verbatim. However, some definitions required for this document were undefined by [RFC4949] or rewritten to better explain their specific use within CEM.

Certificate - A digital certificate contains the owner's (End Entity's) name, the issuer's name, a serial number, expiration date, and a copy of the owner's Public Key. The Public Key is used for Encrypting messages and Verifying Signatures (verifying a signature

is also called Authentication).

Certificate Revocation List (CRL) - A data structure that enumerates digital certificates that have been invalidated by their issuer prior to when they were scheduled to expire. [RFC4949]

Certification Authority (CA) - An entity that issues digital certificates (especially X.509 certificates) and vouches for the binding between the data items in a certificate. [RFC4949]

CA Certificate - A certificate issued by a trusted certification authority. CA certificates are not used to encrypt data but to sign other certificates. CA certificates are signed by themselves, but are not considered self-signed certificates for the purpose of this document.

Certification Hierarchy - In this structure, one CA is the top CA, the highest level of the hierarchy. The top CA may issue public-key certificates to one or more additional CAs that form the second highest level. Each of these CAs may issue certificates to more CAs at the third highest level, and so on. The CAs at the second-lowest of the hierarchy issue certificates only to non-CA entities, called "end entities" that form the lowest level. Thus, all certification paths begin at the top CA and descend through zero or more levels of other CAs. All certificate users base path validations on the top CA's public key. [RFC4949]

CEM Request - The EDIINT Certificate Exchange Messaging (CEM) Request is one of two possible CEM messages. It presents a certificate to be introduced into the trading partner relationship along with relevant information on how it is to be implemented.

CEM Response - The EDIINT Certificate Exchange Messaging (CEM) Response is one of two possible CEM messages. It is the response to the CEM Request indicating whether or not the end entity certificate present in the CEM Request was accepted.

End Entity - A system entity that is the subject of a public-key certificate and that is using, or is permitted and able to use, the matching private key only for a purpose or purposes other than signing a digital certificate; i.e., an entity that is not a CA. [RFC4949]

End Entity Certificate - A certificate which is used to encrypt data or authenticate a signature. (The private key associated with the certificate is used to decrypt data or sign data). The certificate may be self-signed or issued by a trusted certificate.

Intermediary Certificate - A certificate issued by a CA certificate which itself issues another certificate (either intermediary or end entity). Intermediary certificates are not used to encrypt data but

to sign other certificates.

Public Key - The publicly-disclosable component of a pair of cryptographic keys used for asymmetric cryptography. [RFC4949]

Public Key Certificate - A digital certificate that binds a system entity's identity to a public key value, and possibly to additional data items. [RFC4949]

Self-signed Certificate - A certificate which is issued by itself (both issuer and subject are the same) and is an End Entity certificate.

1.3 Certificate Lifecycle

A certificate has five states.

1. Pending - Upon receiving a certificate from a trading partner, the certificate is marked as Pending until a decision can be made to trust it or if its validity period has not yet begun.
2. Rejected - If a Pending certificate is not trusted, it is considered Rejected.
3. Accepted - Once a Pending certificate has been trusted, it is considered Accepted. An Accepted certificate may be used in secure transactions.
4. Expired - A certificate which is no longer valid because its expiration date has passed. Expired certificates SHOULD be kept in a certificate storehouse for decrypting and validating past transactions.
5. Revoked - A certificate which has been explicitly revoked by its owner or the certificate authority.

1.4 Certificate Exchange Process

This section describes a process whereby a company can distribute certificates to its partners, and the company and its partners can put the certificates into use. Later sections describe the specific CEM protocol, which is an implementation of this process.

The exchange process can be used even when CEM is not useable, for example, when the transport protocols or installed software systems do not support CEM. It is RECOMMENDED that this process be followed in distributing certificates.

The company that owns the certificates initiates the process. For a certificate that is to be used (by the partners) to encrypt messages, the initiator first prepares his system to decrypt messages that are encrypted with this certificate. The initiator must also be able to

decrypt messages using the old certificate. The initiator company

distributes the new certificates by some means. The distribution MUST describe the purposes of the certificates and MAY contain a respond by date, the date when the distributor expects to put the certificates in use. The respond by date SHOULD be present for certificates that are used to sign messages or to authenticate TLS/SSL connections.

When a partner receives a certificate, the partner should authenticate the distribution message by some means. (CEM provides for automatic authentication. Partners can use manual methods, either with or without CEM.)

When a partner receives a certificate for use in encrypting messages and has authenticated the certificate, the partner SHOULD begin using that certificate to encrypt messages. The initiator MUST be prepared to receive messages encrypted with either the old or new certificate.

When a partner receives a certificate for use in digitally signing messages or for TLS/SSL authentication and has authenticated the certificate, the partner MUST prepare his system to accept messages that are signed or authenticated with the new certificate. The partner MUST also accept messages signed or authenticated with the old certificate.

The partner MAY return a response to the initiator, indicating that the partner has accepted the new certificate and put it in use. The initiator can use these responses to track which partners are ready to use the new certificate.

When the partner has sent a response indicating acceptance of the new certificate, or when the respond by date has passed, the initiator can begin using the new certificate to digitally sign messages or authenticate TLS/SSL messages. The initiator MUST NOT sign or authenticate messages with the new certificate until the partner has accepted it or until the respond by date has passed. The initiator MAY wait until the respond by date or until all partners have accepted. The partners MUST accept messages signed or authenticated with either the old or new certificate.

When the process is fully automated, it is not necessary to have a specific time when both the initiator and partners switch to the new certificate.

The initiator MUST be able to decrypt messages with both the old and new certificates as soon as the new certificates are distributed. The partners MUST be able to accept messages signed or TLS/SSL authenticated with either the old or new certificates after they have

accepted the new certificate. The initiator SHOULD allow a reasonable time after distributing a new signing or authenticating certificate before putting it in use, so that partners have time to authenticate the new certificate and prepare their systems for it.

For a certificate used to digitally sign messages or authenticate TLS/SSL messages, there must be some way for the initiator to know when partners are ready to receive the certificate. For example, this may be a response from the partners, an explicit respond by date in the initial distribution, an implied respond by date based on partner agreements, or the expiration date of the old certificate. For a certificate used to encrypt messages, the respond by date and responses are less important, but responses may be useful to track partners' acceptances.

2. Message Processing

2.1 Message Structure

CEM messages use the underlying EDIINT transport, such as AS2, to communicate information on the certificate, its intended use and its acceptance. Both digital certificates and the XML data describing their intended use are stored within a multipart/related MIME envelope [RFC2387]. For the CEM Request message, the certificates are stored in certificate chains through SMIME, certs-only MIME envelope, and processing information is XML data which is identified through the MIME content-type of application/ediint-cert-exchange+xml. The format for a CEM Request message is as follows:

```
Various EDIINT headers
Disposition-Notification-To: http://10.1.1.1:80/exchange/as2-company
Content-Type: multipart/signed; micalg=sha1;
protocol="application/pkcs7-signature";
  boundary="--OUTER-BOUNDARY"

----OUTER-BOUNDARY
Content-Type: multipart/related; type="application/ediint-cert-
exchange+xml"; boundary="--INNER-BOUNDARY"

----INNER-BOUNDARY
Content-Type: application/ediint-cert-exchange+xml
Content-ID: <20040101-1.alpha@example.org>
```

```
[CEM XML data]
----INNER-BOUNDARY
Content-Type: application/pkcs7-mime; smime-type=certs-only
Content-ID: <20040101-2.alpha@example.org>

[digital certificate]
----INNER-BOUNDARY--

----OUTER-BOUNDARY
Content-Type: application/pkcs7-signature

[Digital Signature]
----OUTER-BOUNDARY--
```

One and only one MIME type of application/ediint-cert-exchange+xml MUST be present in the multipart/related structure, and it MUST be the root element. Multiple certs-only media types may be included, but at least one MUST be present. A unique content-id header MUST be present within each of the multipart structures.

For the CEM Response message, a multipart/related MIME structure is also used. However, no certificates are present in a CEM Response, and the multipart/related structure only contains one MIME type of application/ediint-cert-exchange+xml. The format for a CEM Request message is as follows:

```
Various EDIINT headers
Disposition-Notification-To: http://10.1.1.1:80/exchange/as2-company
Content-Type: multipart/signed; micalg=sha1;
protocol="application/pkcs7-signature";
  boundary="--OUTER-BOUNDARY"

----OUTER-BOUNDARY
Content-Type: multipart/related; type="application/ediint-cert-
exchange+xml"; boundary="--INNER-BOUNDARY"

----INNER-BOUNDARY
Content-Type: application/ediint-cert-exchange+xml
Content-ID: <20040201-1.alpha@example.org>

[CEM XML data]
----INNER-BOUNDARY--

----OUTER-BOUNDARY
Content-Type: application/pkcs7-signature

[Digital Signature]
----OUTER-BOUNDARY--
```

If possible, both the CEM Request and CEM Response message SHOULD be signed. Applying digital signatures will allow for automatic exchange based on a previous trust relationship. However, it may not be possible in the initial exchange of a new trading partner. If a CEM message is signed, the signing certificate MUST be included in the digital signature. Extra security such as applying data encryption or compression is OPTIONAL. Also, CEM messages SHOULD request a MDN and SHOULD request a signed MDN. The MDN can be either synchronous or asynchronous. All necessary headers MUST be applied to the message per the underlying transport standard.

2.2 EDIINT Features Header

To indicate support for CEM, an EDIINT application MUST use the EDIINT Features header [EDIINT-FEATURE]. The Feature Header indicates the instance application can support various features, such as certification exchange. The header is present in all messages from the instance application, not just those which feature certification exchange.

For applications implementing certification exchange, the CEM-Feature-Name MUST be used within the EDIINT Features header:

```
CEM-Feature-Name = "CEM"
```

An example of the EDIINT Features header in a CEM Message:

```
EDIINT-Features: CEM
```

2.3 Certificate Exchanging

After obtaining the desired certificate, the initiator of the certificate exchange transmits the end-entity certificate in the CEM Request message. If the end-entity certificate is not self-signed, then the CA certificate and any other certificates needed to create the chain of trust for the end-entity certificate MUST be included in the CEM Request message. Multiple end-entity certificates MAY also be present.

The entire certificate trust chain is stored in a BER encoded P7C format [REFERENCE LIKELY NEEDED] and placed within the SMIME certs-only MIME envelope which is then stored in a single part of the multipart/related structure. Each P7C trust chain MUST include a single end-entity certificate and its trust authorities. No other certificates are to be part of this chain. The number of P7C trust

chains in a CEM Request message MUST be equal to the number of end-entity certificates being communicated in the CEM XML document. If different end-entity certificates have common trust authorities' certificates, each P7C cert chain still MUST include each certificate necessary to create a trust anchor. Thus, if a recipient can not create a trust relationship from the P7C cert chain, it MAY reject the end-entity certificate in the CEM Request.

End-entity certificates are referenced and identified in the XML data by their content-id used in the multipart/related structure. Information on how the certificate is to be used, or certificate usage, by the receiving user agent and other related information is found in the XML data. A certificate can be used for a single function, like digital signatures, or used for multiple functions, such as both digital signatures and data encryption. If a certificate is intended for multiple usages, such as for both digital signatures and data encryption, the certificate MUST be listed only once in the CEM Request message and its multiple usage listed through the CertUsage XML element.

Upon receipt of the CEM Request, the recipient trading partner processes the transport message as normal and returns the MDN. The recipient MAY parse the CEM XML data prior to returning the MDN. If the XML is not well-formed and can not be interpreted, the UA MAY return the MDN with the error disposition modifier of "error: unexpected-processing-error". The returned MDN does not provide information on the acceptance of the certificate(s) being exchanged. An UA who receives an MDN with an error disposition modifier MUST consider the CEM Message was not understood and needs to be corrected and retransmitted.

2.4 Certificate Implementation

The new certificate is considered to be in the Pending state for the recipient who MUST decide whether to accept the certificate as trustworthy. This decision is arbitrary and left to each individual trading partner. Upon accepting the certificate, it is to be considered an Accepted certificate within the trading partner relationship. If the certificate is not accepted, it is considered Rejected.

When a certificate is intended for use in data encryption, the initiator MUST consider the certificate to be Accepted and be prepared for its trading partner to begin using the certificate upon generating the CEM Request message. After a recipient generates a positive CEM Response message for a certificate, the recipient MUST

immediately begin using the certificate in trading with the initiator of the request. The recipient MAY apply encryption to the CEM Response message using the new Accepted certificate or MAY apply encryption to the CEM Response message using the previously Accepted encryption certificate.

When a certificate is intended for use in digital signatures or TLS/SSL authentication, the initiator MUST NOT use the certificate until the recipient trading partner generates a CEM Response accepting the certificate or the respond by date, which is listed in the RespondByDate XML element. The initiator MAY use the certificate after the respond by date, regardless of whether the partner has accepted it or not. The certificate used for the digital signature of the CEM Request message MUST be the one which is currently Accepted within the trading partner relationship.

Since implementers of EDIINT often use the same certificate with multiple trading partners, implementers of CEM MUST be able to keep both the old and new certificates as Accepted. If the initiator has generated a CEM Request and exchanged a new encryption certificate to multiple trading partners, it MUST be able to accept encrypted data which uses either the older, existing encryption certificate or the newly exchanged encryption certificate. Likewise, a recipient of a CEM Request MUST be able to authenticate digital signatures using either the new or old certificates, since the initiator may not be able to switch certificates until all trading partners accept the new certificate. Similar provisions MUST be made for certificates intended for TLS/SSL server and client authentication. Revoking a certificate MUST be done outside of CEM.

If a CEM Request message contains a certificate which is currently Accepted and has the identical usage for the certificate that has been Accepted, the recipient MUST NOT reject the duplicate certificate but MUST respond with a CEM Response message indicating the certificate has been accepted. For example, if Certificate A is currently Accepted as the encryption certificate for a user agent,

any CEM Request message containing Certificate A with the usage as encryption only MUST be accepted by an existing trading partner. This situation may be necessary for an implementation intending to verify its current trading partner certificate.

If two trading partners utilize multiple EDIINT protocols for trading, such as AS2 for a primary transport and AS1 as the backup transport, it is dependent upon implementation and trading partner agreement how CEM messages are sent and which transports the exchanged certificates affect.

2.5 CEM Response

The CEM Response message is a multipart/related envelope which contains the CEM XML under the MIME type of application/ediint-cert-exchange+xml. If a requestId is used in a CEM Request, then the requestId MUST be present in the CEM Response with the same value. The requestId allows for the CEM Response to be matched to the CEM Request. If the CEM Request contains multiple TrustRequest elements and the corresponding TrustResponse elements are returned in multiple CEM Response messages, each CEM Response message MUST use the same requestId from the originating CEM Request message. This is critical when multiple CEM Requests are sent with the same certificate and the CEM Response can not be matched solely through the TrustResponse elements.

A TrustResponse XML element provides information needed to match the end-entity certificate sent in an earlier CEM Request and indicate if the certificate was accepted or rejected by the recipient. The CertificateReference in a TrustResponse matches the CertificateIdentifier value for the end-entity certificate in the CEM Request. CertStatus indicates if the certificate was accepted or rejected. If a CEM Request is received, the recipient MUST respond with a CEM Response message indicating if the certificate is Accepted or Rejected. More information about the XML attributes and value for CEM Response can be found in 3.2.

If the certificate in the CEM Request message contains multiple usages, such as for both digital signature and data encryption, only a single TrustResponse is needed for that certificate. The CertStatus value in the TrustResponse is the response for both usages of the certificate. A recipient MUST NOT choose to accept the certificate for one specified use and not the other.

If multiple end-entity certificates were included within the CEM Request, the recipient MAY generate individual CEM Response messages for each certificate or the recipient MAY consolidate the TrustResponse for multiple certificates into one CEM Response message. A CEM Response may contain multiple TrustResponse elements for different certificates but MUST NOT contain two or more TrustResponses for the same certificate.

If a second TrustResponse is received in a different message matching the same certificate as that of an earlier TrustRespnse but the CertStatus has a different value than the other, the originator MAY accept the CertStatus value in the most recent TrustResponse but MAY choose to ignore it. If the CertStatus in both TrustResponses are the

same, the originator should disregard the second TrustResponse.

If the originator receives a CEM Response message which violates the rules listed above or is invalid in any way, the originator MAY reject the message entirely but MUST return an MDN if requested.

3. CEM XML Schema Description

The CEM schema has two top-level elements, EDIINTCertificateExchangeRequest and EDIINTCertificateExchangeResponse. The EDIINTCertificateExchangeRequest element is present only in the CEM Request message, and the EDIINTCertificateExchangeResponse is present only in the CEM Response message. All other elements nest directly or indirectly from these. CEM XML data must be well-formed and valid relative to the CEM XML Schema. Please refer to the appendix for the actual schema document.

3.1 EDIINTCertificateExchangeRequest element

EDIINTCertificateExchangeRequest contains element TradingPartnerInfo, which can only appear once, and TrustRequest, which may be present multiple times. TrustRequest contains information on a certificate and its intended usage. TradingPartnerInfo exists to provide information on the publication of the CEM Request message since processing of the XML data may occur apart from the handling of the accompanying transport message, for example the AS2 request.

```
<xs:element name="EDIINTCertificateExchangeRequest">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="tns:TradingPartnerInfo"/>
         <xs:element name="TrustRequest"
            type="tns:TrustRequestType"
            maxOccurs="unbounded"/>
         <xs:element ref="tns:Extensions" minOccurs="0"
         maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="requestId" type="tns:RequestIdType"
      use="optional"/>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
   </xs:complexType>
</xs:element>
```

EDIINTCertificateExchangeRequest also contains the attribute requestId. RequestId uniquely identifies each CEM Request message.

Its value MUST be between 1 and 255 characters. The requestId is returned in the CEM Response message to assist the UA in matching the CEM Response with the CEM Request.

```
<xs:simpleType name="RequestIdType">
   <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="255"/>
   </xs:restriction>
</xs:simpleType>
```

An optional Extension element is also present along with the anyAttribute attribute. They exist to provide future extendibility for new features which may be developed but not yet defined within CEM. They are present in several locations in the schema for this future extendibility.

```
<xs:element name="Extensions">
   <xs:complexType>
      <xs:sequence>
         <xs:any namespace="##any" processContents="lax"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

TradingPartnerInfo identifies the entity that created the CEM message through the nested Name element. Both the qualifier attribute and the element value of Name follow mandatory naming conventions. The qualifier attribute is to be the transport standard utilized. For example, "AS1", "AS2" or "AS3". The value of the Name element is the same value in the From header utilized by the transport. For AS2 and AS3, this is the value in the AS2-From and AS3-From headers, respectively. For AS1, this is the value of the From header. If other transports besides AS1, AS2, AS3 are used, the same naming convention SHOULD be followed.

MessageOriginated is included in TradingPartnerInfo to identify the time and date the message was created. The MessageOriginated date and time values MUST follow XML standard dateTime type syntax and be listed to at least the nearest second and expressed in local time with UTC offset. For example, a message originating from the US Eastern Standard timezone would use 2005-03-01T14:05:00-05:00.

```
      <xs:element name="TradingPartnerInfo">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Name">
                  <xs:complexType>
                     <xs:simpleContent>
                        <xs:extension base="xs:string">
                           <xs:attribute name="qualifier"
                            type="xs:string" />
                        </xs:extension>
                     </xs:simpleContent>
                  </xs:complexType>
               </xs:element>
               <xs:element name="MessageOriginated" type="xs:dateTime"/>
               <xs:any namespace="##any" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute namespace="##any" processContents="lax"/>
         </xs:complexType>
      </xs:element>
```

The TrustRequest element contains the EndEntity, CertUsage, RespondByDate and ResponseURL elements. The required EndEntity element is found only once in a TrustRequest element and contains the content-id reference to the end-entity certificate being exchanged.

```
      <xs:complexType name="TrustRequestType">
         <xs:sequence>
            <xs:element ref="tns:CertUsage" maxOccurs="unbounded"/>
            <xs:element ref="tns:RespondByDate" minOccurs="0"/>
            <xs:element ref="tns:ResponseURL"/>
            <xs:element name="EndEntity" type="tns:EndEntityType"/>
            <xs:any namespace="##any" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
```

EndEntity contains the nested elements of CertificateIdentifier and CertificateContentID. CertificateContentID is a string element which references the content-id of the multipart/related structure where the certificate is stored. CertificateIdentifier comes from the XML Signature schema namespace [XML-DSIG].

```
      <xs:complexType name="EndEntityType">
         <xs:sequence>
            <xs:element name="CertificateIdentifier"
```

```
                type="ds:X509IssuerSerialType"/>
```

```
         <xs:element name="CertificateContentID" type="xs:string"/>
         <xs:any namespace="##any" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
   </xs:complexType>
```

CertificateIdentifier contains the string element X509IssuerName and the integer element X509SerialNumber. X509SerialNumber is the assigned serial number of the end entity certificate as it is listed. X509IssuerName contains the issuer name information of the end-entity certificate, such as common name, organization, etc. This information MUST be described in a string format per the rules of RFC 4514 [RFC4514]. This results in the attributes within the Issuer Name to be listed with their attribute type followed by an "=" and the attribute value. Each attribute type and value are separated by a "," and any escape characters in the value are preceded by a "\". Refer to the appendix and the sample CEM Request message for an example of the X509IssuerName.

```
      <complexType name="X509IssuerSerialType">
            <sequence>
                 <element name="X509IssuerName" type="string"/>
                 <element name="X509SerialNumber" type="integer"/>
           </sequence>
     </complexType>
```

CertUsage is an unbounded element which contains enumerated values on how the exchanged certificate is to be used. There are enumerated values for SMIME digital signatures (digitalSignature), SMIME data encryption (keyEncipherment), the server certificate used in TLS transport encryption (tlsServer) and the client certificate used in TLS transport encryption (tlsClient). While the element is unbounded, CertUsage only has a potential number of four occurrences due to the limit of the enumerated values.

```
   <xs:element name="CertUsage" type="tns:CertUsageType"/>
   <xs:simpleType name="CertUsageType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="tlsClient"/>
         <xs:enumeration value="tlsServer"/>
         <xs:enumeration value="keyEncipherment"/>
         <xs:enumeration value="digitalSignature"/>
      </xs:restriction>
   </xs:simpleType>
```

RespondByDate is a required element of the XML standard dateTime type expressed in local time with UTC offset, which provides information on when the certificate should be trusted, inserted into the trading partner relationship and responded to by a CEM Response message. If the certificate can not be trusted or inserted into the trading partner relationship, the CEM Response message should still be returned by the date indicated.

```
<xs:element name="RespondByDate" type="xs:dateTime"/>
```

ResponseURL is an element which indicates where the CEM Response message should be sent. This value takes precedence over the existing inbound URL of the current trading partner relationship. The Response MUST use the same transport protocol (AS1, AS2, or AS3) as the Request.

```
<xs:element name="ResponseURL" type="xs:anyURI"/>
```

3.2 EDIINTCertificateExchangeResponse element

EDIINTCertificateExchangeResponse contains the two elements TradingPartnerInfo and TrustResponse and the attribute requestId. TradingPartnerInfo, which is also found in EDIINTCertificateExchangeRequest, describes the trading partner generating this response message. TrustResponse provides information on the acceptance of a previously sent end entity certificate. There can be multiple TrustResponse elements within an EDIINTCertificateExchangeResponse. The requestId is the same value from a previously sent CEM Request message. The requestId from the CEM Response is matched up with the CEM Request.

```
<xs:element name="EDIINTCertificateExchangeResponse">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="tns:TradingPartnerInfo"/>
         <xs:element name="TrustResponse"
             type="tns:TrustResponseType"
             maxOccurs="unbounded"/>
         <xs:element ref="tns:Extensions" minOccurs="0"
            maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="requestId" type="tns:RequestIdType"
       use="optional"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
   </xs:complexType>
</xs:element>

<xs:complexType name="TrustResponseType">
```

```
        <xs:sequence>
```

```
            <xs:element ref="tns:CertStatus"/>
            <xs:element ref="tns:ReasonForRejection" minOccurs="0"/>
            <xs:element name="CertificateReference"
                        type="ds:X509IssuerSerialType"/>
             <xs:any namespace="##any" processContents="lax"
             minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
```

A TrustResponse element identifies a certificate which has been previously exchanged within the trading partner relationship through a CEM Request and now has been either accepted or rejected by the partner. The CertificateReference element is of the same type as the CertificateIdentifier element. A CertificateReference element in a CEM Response MUST be identical to its CertificateIdentifier counterpart in the associated CEM Request since they identify the same certificate in question.

The required element CertStatus has the enumerated values of "Accepted" or "Rejected". "Accepted" indicates the certificate was trusted by the trading partner and is now ready for use within the trading partner relationship, and "Rejected" indicates the certificate is not trusted by the trading partner nor can it be currently used with the trading partner relationship. If the value of "Rejected" is chosen, the optional string element ReasonForRejection may be included. If present, ReasonForRejection should contain a brief description of why the certificate was not accepted. Since the value for this element is not enumerated but open, it MUST be interpreted through human means.

```
      <xs:element name="CertStatus" type="tns:CertStatusType"/>
      <xs:simpleType name="CertStatusType">
         <xs:restriction base="xs:string">
            <xs:enumeration value="Rejected"/>
            <xs:enumeration value="Accepted"/>
         </xs:restriction>
      </xs:simpleType>
      <xs:element name="ReasonForRejection" type="xs:string"/>
```

4. Use Case Scenario

This scenario illustrates how the CEM Request and CEM Response messages described in Section 2 and 3 can be used to exchange certificates. The scenario is only illustrative and any differences between it and the rules above should defer to the rules in Section 2 and 3.

Two trading partners, Alpha Arrows and Bravo Bows, have an established trading partner relationship using AS2. Alpha Arrows is using a single certificate, CertA, for both digital signatures and data encryption. Alpha Arrows wants to issue a new certificate, CertB, for digital signatures but keep CertA for data encryption.

Bravo Bows is using one certificate, Cert1, for digital signatures and another certificate, Cert2, for data encryption. Bravo Bows wants to introduce a new certificate, Cert3, for digital signature and a new certificate, Cert4, for data encryption.

1. Alpha Arrows sends a CEM Request to Bravo Bows containing only CertB. The CertUsage has a value of "digitalSignature". Bravo Bows immediately returns the MDN but must make an internal security decision before accepting CertB.

2. While waiting for a CEM Response, Alpha Arrows continues to send AS2 messages to Bravo Bows which have been signed using CertA. The messages originating from Bravo Bows are encrypted using CertA.

3. Eventually, Bravo Bows returns a CEM Response with the CertStatus of "Accepted" for CertB. Upon receipt, an MDN is returned which is signed using CertA. Bravo Bows MUST be able to accept the MDN if it has a digital signature from either CertA or CertB as Alpha Arrows may not be able to switch certificates simply upon receipt of the CEM Response message without parsing the XML payload. Also, Alpha Arrows may need to wait for CEM Responses from other trading partners before switching to the new CertB. However, as soon as possible, Alpha Arrows should use CertB exclusively for digital signatures.

4. Bravo Bows sends a CEM Request to Alpha Arrows containing both Cert3 and Cert4. The CertUsage for Cert3 and Cert4 are "digitalSignature" and "keyEncipherment", respectively. Alpha Arrows returns an MDN immediately. Bravo Bows is now prepared to receive any inbound messages encrypted by either Cert2 or Cert4, but all its digital signatures are still done through Cert1.

5. Eventually, Alpha Arrows returns a single CEM Response message. It contains two TrustResponse elements: one for Cert3 and another for Cert4. The CertStatus for Cert3 is "Rejected" with the ReasonForRejection field present and populated with the string "KeyUsage value was incorrect". CertStatus for Cert4 was "Accepted." Bravo Bows returns the MDN signed through Cert1.

6. Immediately after this, an AS2 message is received from Alpha Arrows which is encrypted using Cert4, and Bravo Bows is able to

decrypt the message successfully. Because Alpha Arrows rejected Cert3, Bravo Bows is only using Cert1 for digital signatures and returns the MDN signed with Cert1.

7. After creating a new certificate, Cert5, which corrects the previous keyUsage problem, Bravo Bows sends Cert5 in a CEM Request.

8. Shortly after this, Alpha Arrows sends a CEM Response message for Cert5. It contains a CertStatus of "Accepted". This CEM Response message was encrypted using Cert4, but Bravo Bows was prepared for encryption from either Cert2 or Cert4. The message is processed and a good MDN is returned signed with Cert1. While, Bravo Bows can now sign messages to Alpha Arrows with either Cert1 or Cert5, Bravo Bows should use Cert5 exclusively as soon as possible.

5. Profile Exchange Messaging

CEM provides the means to exchange certificates among trading partners. However, other profile information, such as URLs and preferred security settings, is needed to create a trading partner relationship. A future standard is needed to describe profile descriptions and how they will be exchanged. The format for this profile attachment is not defined in this specification but is planned for a future document. It will build upon the existing CEM protocol with profile information stored with XML data. Both certificate and profile description information will be placed into a multipart/related [RFC2387] body part entity. A possible format for a profile description message is as follows:

```
Various EDIINT headers
EDIINT-Features: profile-exchange
Disposition-Notification-To: http://10.1.1.1:80/exchange/as2_company
Disposition-Notification-Options: signed-receipt-protocol=optional,
  pkcs7-signature; signed-receipt-micalg=optional, sha1
Content-Type: multipart/signed; micalg=sha1;
  protocol="application/pkcs7-signature"; boundary="--BOUNDARY1"

----BOUNDARY1
Content-Type: multipart/related;
  start="<aayxdfl01012000@foo.com>";
  type="application/ediint-cert-exchange+xml";
  boundary="--BOUNDARY2"

----BOUNDARY2
Content-Type: application/ediint-cert-exchange+xml
Content-ID: <aayxdfl01012000@foo.com>
```

```
[CEM XML data]
----BOUNDARY2
[Profile information attachment]
----BOUNDARY2--
----BOUNDARY1

Content-Type: application/pkcs7-signature

[Digital Signature]
----BOUNDARY1--
```

6. Implementation Considerations

This section contains various points to explain practical implementation considerations.

* If the EDIINT transport message carrying a CEM Request or CEM Response fails resulting in a negative MDN, the CEM message, its contents and instructions are to be ignored. The User Agent receiving the negative MDN is to consider the CEM message to be ignored and retransmit as needed.

* While a single end-entity certificate can be only be used once in a single CEM Request message, the same certificate can be sent multiple times in multiple CEM Request messages. The requestId is used for matching the CEM Request and CEM Response messages.

* Certificate usage is cumulative. Thus, if a User Agent receives a valid CEM Request message with Certificate A with certUsage set to digitalSignature and then a second valid CEM Request message with Certificate A with certUsage set to keyEncipherment, then the User Agent MUST configure the certificate to be used both for digitalSignature and keyEncipherment. As well, if at a later time a valid CEM Request message is received with Certificate A with certUsage set only to digitalSignature, Certificate A is still valid for keyEncipherment.

7. Future Considerations for CEM I-D

This section contains ideas for consideration in future versions of CEM and addressed in the future. If deemed necessary, they will be added into the I-D else they will be removed. This section will be removed prior to RFC submission.

8. Security Considerations

Certificate exchange is safe for transmitting. However, implementers SHOULD verify the received certificate to determine if it is truly from the stated originator through out-of-band means or whenever the request is not signed.

9. IANA Considerations

MIME Media type name: Application

MIME subtype name: EDIINT-cert-exchange+xml

Required parameters: None

Optional parameters: This parameter has identical semantics to the charset parameter of the "application/xml" media type as specified in [RFC3023].

Encoding considerations: Identical to those of "application/xml" as described in [RFC3023], section 3.2.

Security considerations: See section 6.

Interoperability Considerations: See section 2.2

Published specification: This document.

Applications which use this media type: EDIINT applications, such as AS1, AS2 and AS3 implementations.

Additional Information: None

Intended Usage: Common

Author/Change controller: See Author's section of this document.

10. References

10.1 Normative References

[AS1] RFC3335 "MIME-based Secure Peer-to-Peer Business Data Interchange over the Internet using SMTP", T. Harding, R. Drummond, C. Shih, 2002.

[AS2] RFC4130 "MIME-based Secure Peer-to-Peer Business Data Interchange over the Internet using HTTP", D. Moberg, R. Drummond, 2005.

[AS3] RFC4823 "FTP Transport for Secure Peer-to-Peer Business Data Interchange over the Internet", T. Harding, R. Scott, 2007.

[EDIINT-FEATURE] RFC 6017, "Electronic Data Interchange - Internet Integration (EDIINT) Features Header Field", Meadors, K., September 2010.

[RFC2119] RFC2119 "Key Words for Use in RFC's to Indicate Requirement Levels", S.Bradner, March 1997.

[RFC4514] RFC4514 "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", K. Zeilenga, Ed. June 2006.

[RFC2387] RFC2387 "The MIME Multipart/Related Content-type", E. Levinson, August 1998.

[RFC4949] RFC4949 "Internet Security Glossary, Version 2", R. Shirley, August 2007.

[RFC3023] RFC3023 "XML Media Types", M. Murata, October 2001.

[XML-DSIG] RFC3275 "XML-Signature Syntax and Processing", D. Eastlake, March 2002.

[X.520] ITU-T Recommendation X.520: Information Technology - Open Systems Interconnection - The Directory: Selected Attribute Types, 1993.

10.2 Informative References

11. Acknowledgments

The authors wish to extend gratitude to the ecGIF sub-committee within the GS1 organization from which this effort began. Many have contributed to the development of this specification, but some deserve special recognition. John Duker who chaired the sub-committee and provided valuable editing. John Koehring with his work on the reference ID and shared important insights on implementation. Aaron Gomez in the coordinating of vendors testing CEM. Richard Bigelow who greatly assisted development of the ideas presented, and Debra Petta for her review and comments.

Author's Addresses

Kyle Meadors
Drummond Group Inc.
7333 Riverfront Dr.
Nashville, TN
Email: kyle@drummondgroup.com

Dale Moberg
Axway, Inc.
8388 E. Hartford Drive, Suite 100
Scottsdale, AZ 85255 USA
Email: dmoberg@us.axway.com

Appendix

A.1 EDIINT Certificate Exchange XML Schema

```
<?xml version="1.0"?>
<xs:schema
xmlns:tns="urn:ietf:params:xml:ns:ediintcertificateexchange"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
targetNamespace="urn:ietf:params:xml:ns:ediintcertificateexchange"
elementFormDefault="qualified">
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="xmldsig-core-schema.xsd"/>
   <xs:element name="EDIINTCertificateExchangeRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="tns:TradingPartnerInfo"/>
            <xs:element name="TrustRequest"
type="tns:TrustRequestType" maxOccurs="unbounded"/>
            <xs:element ref="tns:Extensions" minOccurs="0"
maxOccurs="1"/>
         </xs:sequence>
         <xs:attribute name="requestId" type="tns:RequestIdType"
          use="optional"/>
         <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="EDIINTCertificateExchangeResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="tns:TradingPartnerInfo"/>
            <xs:element name="TrustResponse"
type="tns:TrustResponseType" maxOccurs="unbounded"/>
            <xs:element ref="tns:Extensions" minOccurs="0"
maxOccurs="1"/>
         </xs:sequence>
         <xs:attribute name="requestId" type="tns:RequestIdType"
          use="optional"/>
         <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="TradingPartnerInfo">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Name">
               <xs:complexType>
                  <xs:simpleContent>
```

```
                        <xs:extension base="xs:string">
                           <xs:attribute name="qualifier"
   type="xs:string" use="optional"/>
                        </xs:extension>
                     </xs:simpleContent>
                  </xs:complexType>
               </xs:element>
               <xs:element name="MessageOriginated" type="xs:dateTime"/>
               <xs:any namespace="##any" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute namespace="##any" processContents="lax"/>
         </xs:complexType>
      </xs:element>
      <xs:simpleType name="RequestIdType">
         <xs:restriction base="xs:string">
            <xs:minLength value="1"/>
            <xs:maxLength value="255"/>
         </xs:restriction>
      </xs:simpleType>
      <xs:element name="CertUsage" type="tns:CertUsageType"/>
      <xs:simpleType name="CertUsageType">
         <xs:restriction base="xs:string">
            <xs:enumeration value="tlsClient"/>
            <xs:enumeration value="tlsServer"/>
            <xs:enumeration value="keyEncipherment"/>
            <xs:enumeration value="digitalSignature"/>
         </xs:restriction>
      </xs:simpleType>
      <xs:element name="CertStatus" type="tns:CertStatusType"/>
      <xs:simpleType name="CertStatusType">
         <xs:restriction base="xs:string">
            <xs:enumeration value="Rejected"/>
            <xs:enumeration value="Accepted"/>
         </xs:restriction>
      </xs:simpleType>
      <xs:element name="ReasonForRejection" type="xs:string"/>
      <xs:element name="RespondByDate" type="xs:dateTime"/>
      <xs:element name="ResponseURL" type="xs:anyURI"/>
      <xs:complexType name="EndEntityType">
         <xs:sequence>
            <xs:element name="CertificateIdentifier"
   type="ds:X509IssuerSerialType"/>
            <xs:element name="CertificateContentID" type="xs:string"/>
            <xs:any namespace="##any" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
```

```
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="TrustRequestType">
         <xs:sequence>
            <xs:element ref="tns:CertUsage" maxOccurs="unbounded"/>
            <xs:element ref="tns:RespondByDate" minOccurs="0"/>
            <xs:element ref="tns:ResponseURL"/>
            <xs:element name="EndEntity" type="tns:EndEntityType"/>
            <xs:any namespace="##any" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="TrustResponseType">
         <xs:sequence>
            <xs:element ref="tns:CertStatus"/>
            <xs:element ref="tns:ReasonForRejection" minOccurs="0"/>
            <xs:element name="CertificateReference"
   type="ds:X509IssuerSerialType"/>
            <xs:any namespace="##any" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
      <xs:element name="Extensions">
         <xs:complexType>
            <xs:sequence>
               <xs:any namespace="##any" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
      </xs:element>
   </xs:schema>
```

A.2 Example of EDIINT Certificate Exchange Request XML

```
   <?xml version="1.0" standalone="yes"?>
   <EDIINTCertificateExchangeRequest
   xmlns="urn:ietf:params:xml:ns:ediintcertificateexchange"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   requestId="01170702153479-msgAlphaBravo">
      <TradingPartnerInfo>
         <Name qualifier="AS2">DGI_Test_CEM</Name>
         <MessageOriginated>
            2005-08-30T00:30:00-05:00</MessageOriginated>
      </TradingPartnerInfo>
      <TrustRequest>
```

```
        <CertUsage>keyEncipherment</CertUsage>
        <CertUsage>digitalSignature</CertUsage>
        <RespondByDate>2005-09-30T12:00:00-05:00</RespondByDate>
        <ResponseURL>http://10.1.1.1/as2</ResponseURL>
        <EndEntity>
           <CertificateIdentifier>
              <ds:X509IssuerName>CN=Cleo-
SP,E=as2selfpacedsupport@drummondgroup.com,O=DGI,OU=DGI,L=Ft.
Worth,S=Texas,C=US</ds:X509IssuerName>
   <ds:X509SerialNumber>96596846110948734748886</ds:X509SerialNumber>
           </CertificateIdentifier>
           <CertificateContentID>
           SignEncCert-Example_vs02@example.org</CertificateContentID>
        </EndEntity>
     </TrustRequest>
     <TrustRequest>
        <CertUsage>tlsServer</CertUsage>
        <RespondByDate>2005-09-30T12:00:00-05:00</RespondByDate>
        <ResponseURL>http://10.1.1.1/as2</ResponseURL>
        <EndEntity>
           <CertificateIdentifier>
              <ds:X509IssuerName>CN=VeriSign Class 1 CA Individual
Subscriber-Persona Not Validated,OU=www.verisign.com/repository/RPA
Incorp. By Ref.\,LIAB.LTD(c)98,OU=VeriSign Trust Network,O=VeriSign\,
Inc.</ds:X509IssuerName>
   <ds:X509SerialNumber>267361101459781766955086174427996682</ds:X50
9SerialNumber>
           </CertificateIdentifier>
           <CertificateContentID>
              SSLCert-Example_vs02@example.org</CertificateContentID>
        </EndEntity>
      </TrustRequest>
</EDIINTCertificateExchangeRequest>
```

A.3 Example of EDIINT Certificate Exchange Response XML

```
<?xml version="1.0"?>
<EDIINTCertificateExchangeResponse
xmlns="urn:ietf:params:xml:ns:ediintcertificateexchange"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
requestId="01170702153479-msgAlphaBravo">
   <TradingPartnerInfo>
      <Name qualifier="AS2">DGI_Test_CEM_Trading_Partner</Name>
      <MessageOriginated>
      2005-08-31T00:21:00-05:00</MessageOriginated>
   </TradingPartnerInfo>
   <TrustResponse>
```

```
         <CertStatus>Accepted</CertStatus>
         <CertificateReference>
            <ds:X509IssuerName>CN=Cleo-
   SP,E=as2selfpacedsupport@drummondgroup.com,O=DGI,OU=DGI,L=Ft.
   Worth,S=Texas,C=US</ds:X509IssuerName>
      <ds:X509SerialNumber>9659684611094873474886</ds:X509SerialNumber>
         </CertificateReference>
      </TrustResponse>
      <TrustResponse>
         <CertStatus>Accepted</CertStatus>
         <CertificateReference>
            <ds:X509IssuerName>CN=VeriSign Class 1 CA Individual
   Subscriber-Persona Not Validated,OU=www.verisign.com/repository/RPA
   Incorp. By Ref.\,LIAB.LTD(c)98,OU=VeriSign Trust Network,O=VeriSign\,
   Inc.</ds:X509IssuerName>
      <ds:X509SerialNumber>267361101459781766955086174427996682</ds:X50
   9SerialNumber>
         </CertificateReference>
      </TrustResponse>
   </EDIINTCertificateExchangeResponse>
```

Changes from Previous Versions

B.1 Updates from Version 00

- Updated security requirements in section 2.1, specifically in regards to digital signatures.
- The XML element responseURL is now required. Modified section 3.1 and example messages in appendix accordingly.
- Certificates are exchanged within a full P7C cert chain. Section 2.3 reflects this.
- The XML element TrustChain is not longer necessary since the entire cert chain is stored. Removed references in schema and document.
- Added statement in 2.5 that multiple CEM Responses SHOULD NOT be sent and that if this occurs, the action of the CEM Request initiator is not defined.
- Updated the examples in Appendix B to reflect the current usage.

B.2 Updates from Version 01

- Added information for handling different scenarios with CEM Response message.
- Rewrote use case scenarios.
- Added the EDIINT Features header information.

B.3 Updates from Version 02

- Modified use of SSL certificates to match real-world needs.
- Modified schema in changing namespace value and removed schema location.
- Added statement that CEM XML must be well-formed and valid to schema.
- Modified Use Case to correct an error and improve clarity.
- Added section 1.4 to describe CEM process.

B.4 Updates from Version 03

- None. Update done because vs03 expired.

B.5 Updates from Version 04

- Clarified requirement of using multipart/related for CEM Response.
- Added sections on Implementation Considerations and Future Implementation.
- Modified schema to allow future extensions.
- Changed requirements on qualifier attribute in the Name element.
- Changed functionality to allow error MDN to be returned when CEM XML can not be parsed.

B.6 Updates from Version 05

- Added requestId to CEM.
- Removed normative reference to RFC 3821.

B.7 Updates from Version 06/07/08/09/10/11/12/13

- None. Updated for 6-month I-D expiration.