Network Working Group
Request for Comments: 4130
Category: Standards Track

D. Moberg
Cyclone Commerce
R. Drummond
Drummond Group Inc.

July 2005

MIME-Based Secure Peer-to-Peer Business Data Interchange Using HTTP, Applicability Statement 2 (AS2)

Status of This Memo

Copyright Notice

Abstract

This document provides an applicability statement (RFC 2026, Section 3.2) that describes how to exchange structured business data securely using the HTTP transfer protocol, instead of SMTP; the applicability statement for SMTP is found in RFC 3335. Structured business data may be XML; Electronic Data Interchange (EDI) in either the American National Standards Committee (ANSI) X12 format or the UN Electronic Data Interchange for Administration, Commerce, and Transport (UN/EDIFACT) format; or other structured data formats. The data is packaged using standard MIME structures. Authentication and data confidentiality are obtained by using Cryptographic Message Syntax with S/MIME security body parts. Authenticated acknowledgements make use of multipart/signed Message Disposition Notification (MDN) responses to the original HTTP message. This applicability statement is informally referred to as "AS2" because it is the second applicability statement, produced after "AS1", RFC 3335.

Table of Contents

1. Introduction

1.1. Applicable RFCs

Previous work on Internet EDI focused on specifying MIME content types for EDI data [2] and extending this work to support secure EC/EDI transport over SMTP [4]. This document expands on RFC 1767 to specify a comprehensive set of data security features, specifically data confidentiality, data integrity/authenticity, non-repudiation of origin, and non-repudiation of receipt over HTTP. This document also recognizes contemporary RFCs and is attempting to "re-invent" as little as possible. Although this document focuses on EDI data, any other data types describable in a MIME format are also supported.

Internet MIME-based EDI can be accomplished by using and complying with the following RFCs:

- RFC 2616 Hyper Text Transfer Protocol
- RFC 1767 EDI Content Type
- RFC 3023 XML Media Types
- RFC 1847 Security Multiparts for MIME
- RFC 3462 Multipart/Report
- RFC 2045 to 2049 MIME RFCs
- RFC 3798 Message Disposition Notification
- RFC 3851, 3852 S/MIME v3.1 Specification

Our intent here is to define clearly and precisely how these are used together, and what is required by user agents to be compliant with 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 [13].

1.2. Terms

AS2: Applicability Statement 2 (this document); see RFC 2026 [11], Section 3.2

EDI: Electronic Data Interchange

EC: Business-to-Business Electronic Commerce

B2B: Business to Business

Receipt: The functional message that is sent from a receiver to a sender to acknowledge receipt of an EDI/EC interchange. This message may be either synchronous or asynchronous in nature.

Signed Receipt: A receipt with a digital signature.

Synchronous Receipt: A receipt returned to the sender during the same HTTP session as the sender's original message.

Asynchronous Receipt: A receipt returned to the sender on a different communication session than the sender's original message session.

Message Disposition Notification (MDN): The Internet messaging format used to convey a receipt. This term is used interchangeably with receipt. A MDN is a receipt.

Non-repudiation of receipt (NRR): A "legal event" that occurs when the original sender of an signed EDI/EC interchange has verified the signed receipt coming back from the receiver. The receipt contains data identifying the original message for which it is a receipt, including the message-ID and a cryptographic hash (MIC). The original sender must retain suitable records providing evidence concerning the message content, its message-ID, and its hash value. The original sender verifies that the retained hash value is the same as the digest of the original message, as reported in the signed receipt. NRR is not considered a technical message, but instead is thought of as an outcome of possessing relevant evidence.

S/MIME: A format and protocol for adding cryptographic signature and/or encryption services to Internet MIME messages.

Cryptographic Message Syntax (CMS): An encapsulation syntax used to digitally sign, digest, authenticate, or encrypt arbitrary messages.

SHA-1: A secure, one-way hash algorithm used in conjunction with digital signature. This is the recommended algorithm for AS2.

MD5: A secure, one-way hash algorithm used in conjunction with digital signature. This algorithm is allowed in AS2.

MIC: The message integrity check (MIC), also called the message digest, is the digest output of the hash algorithm used by the digital signature. The digital signature is computed over the MIC.

User Agent (UA): The application that handles and processes the AS2 request.

2. Overview

2.1. Overall Operation

A HTTP POST operation [3] is used to send appropriately packaged EDI, XML, or other business data. The Request-URI ([3], Section 9.5) identifies a process for unpacking and handling the message data and for generating a reply for the client that contains a message disposition acknowledgement (MDN), either signed or unsigned. The MDN is either returned in the HTTP response message body or by a new HTTP POST operation to a URL for the original sender.

This request/reply transactional interchange can provide secure, reliable, and authenticated transport for EDI or other business data using HTTP as a transfer protocol.

The security protocols and structures used also support auditable records of these document data transmissions, acknowledgements, and authentication.

2.2. Purpose of a Security Guideline for MIME EDI

The purpose of these specifications is to ensure interoperability between B2B EC user agents, invoking some or all of the commonly expected security features. This document is also NOT limited to strict EDI use; it applies to any electronic commerce application for which business data needs to be exchanged over the Internet in a secure manner.

2.3. Definitions

2.3.1. The Secure Transmission Loop

This document’s focus is on the formats and protocols for exchanging EDI/EC content securely in the Internet’s HTTP environment.

In the "secure transmission loop" for EDI/EC, one organization sends a signed and encrypted EDI/EC interchange to another organization and

requests a signed receipt, and later the receiving organization sends this signed receipt back to the sending organization. In other words, the following transpires:

- The organization sending EDI/EC data signs and encrypts the data using S/MIME. In addition, the message will request that a signed receipt be returned to the sender. To support NRR, the original sender retains records of the message, message-ID, and digest (MIC) value.

- The receiving organization decrypts the message and verifies the signature, resulting in verified integrity of the data and authenticity of the sender.

- The receiving organization then returns a signed receipt using the HTTP reply body or a separate HTTP POST operation to the sending organization in the form of a signed message disposition notification. This signed receipt will contain the hash of the received message, allowing the original sender to have evidence that the received message was authenticated and/or decrypted properly by the receiver.

The above describes functionality that, if implemented, will satisfy all security requirements and implement non-repudiation of receipt for the exchange. This specification, however, leaves full flexibility for users to decide the degree to which they want to deploy those security features with their trading partners.

2.3.2. Definition of Receipts

The term used for both the functional activity and the message for acknowledging delivery of an EDI/EC interchange is "receipt" or "signed receipt". The first term is used if the acknowledgment is for an interchange resulting in a receipt that is NOT signed. The second term is used if the acknowledgement is for an interchange resulting in a receipt that IS signed.

The term non-repudiation of receipt (NRR) is often used in combination with receipts. NRR refers to a legal event that occurs only when the original sender of an interchange has verified the signed receipt coming back from recipient of the message, and has verified that the returned MIC value inside the MDN matches the previously recorded value for the original message.

NRR is best established when both the original message and the receipt make use of digital signatures. See the Security Considerations section for some cautions regarding NRR.

For information on how to format and process receipts in AS2, refer to Section 7.

2.4. Assumptions

2.4.1. EDI/EC Process Assumptions

- Encrypted object is an EDI/EC Interchange.

This specification assumes that a typical EDI/EC interchange is the lowest-level object that will be subject to security services.

Specifically, in EDI ANSI X12, this means that anything between and including, segments ISA and IEA is secured. In EDIFACT, this means that anything between, and including, segments UNA/UNB and UNZ is secured. In other words, the EDI/EC interchanges including envelope segments remain intact and unreadable during fully secured transport.

- EDI envelope headers are encrypted.

Congruent with the above statement, EDI envelope headers are NOT visible in the MIME package.

In order to optimize routing from existing commercial EDI networks (called Value Added Networks or VANs) to the Internet, it would be useful to make some envelope information visible. This specification, however, provides no support for this optimization.

- X12.58 and UN/EDIFACT Security Considerations

The most common EDI standards bodies, ANSI X12 and EDIFACT, have defined internal provisions for security. X12.58 is the security mechanism for ANSI X12, and AUTACK provides security for EDIFACT. This specification does NOT dictate use or non-use of these security standards. They are both fully compatible, though possibly redundant, with this specification.

2.4.2. Flexibility Assumptions

- Encrypted or Unencrypted Data

This specification allows for EDI/EC message exchange in which the EDI/EC data can be either unprotected or protected by means of encryption.

- Signed or Unsigned Data

This specification allows for EDI/EC message exchange with or without digital signature of the original EDI transmission.

- Optional Use of Receipt

This specification allows for EDI/EC message transmission with or without a request for receipt notification. A signed receipt notification is requested; however, a MIC value is REQUIRED as part of the returned receipt, except when a severe error condition prevents computation of the digest value. In the exceptional case, a signed receipt should be returned with an error message that effectively explains why the MIC is absent.

- Use of Synchronous or Asynchronous Receipts

In addition to a receipt request, this specification allows the specification of the type of receipt that should be returned. It supports synchronous or asynchronous receipts in the MDN format specified in Section 7 of this document.

- Security Formatting

This specification relies on the guidelines set forth in RFC 3851/3852 [7] "S/MIME Version 3.1 Message Specification; Cryptographic Message Syntax".

- Hash Function, Message Digest Choices

When a signature is used, it is RECOMMENDED that the SHA-1 hash algorithm be used for all outgoing messages, and that both MD5 and SHA-1 be supported for incoming messages.

- Permutation Summary

In summary, the following twelve security permutations are possible in any given trading relationship:

1. Sender sends un-encrypted data and does NOT request a receipt.

2. Sender sends un-encrypted data and requests an unsigned receipt. Receiver sends back the unsigned receipt.

3. Sender sends un-encrypted data and requests a signed receipt. Receiver sends back the signed receipt.

4. Sender sends encrypted data and does NOT request a receipt.

5. Sender sends encrypted data and requests an unsigned receipt. Receiver sends back the unsigned receipt.

6. Sender sends encrypted data and requests a signed receipt. Receiver sends back the signed receipt.

7. Sender sends signed data and does NOT request a signed or unsigned receipt.

8. Sender sends signed data and requests an unsigned receipt. Receiver sends back the unsigned receipt.

9. Sender sends signed data and requests a signed receipt. Receiver sends back the signed receipt.

10. Sender sends encrypted and signed data and does NOT request a signed or unsigned receipt.

11. Sender sends encrypted and signed data and requests an unsigned receipt. Receiver sends back the unsigned receipt.

12. Sender sends encrypted and signed data and requests a signed receipt. Receiver sends back the signed receipt.

Users can choose any of the twelve possibilities, but only the last example (12), when a signed receipt is requested, offers the whole suite of security features described in Section 2.3.1, "The Secure Transmission Loop".

Additionally, the receipts discussed above may be either synchronous or asynchronous depending on the type requested. The use of either the synchronous or asynchronous receipts does not change the nature of the secure transmission loop in support of NRR.

3. Referenced RFCs and Their Contributions

3.1. RFC 2616 HTTP v1.1 [3]

This document specifies how data is transferred using HTTP.

3.2. RFC 1847 MIME Security Multiparts [6]

This document defines security multipart for MIME: multipart/encrypted and multipart/signed.

3.3. RFC 3462 Multipart/Report [8]

This RFC defines the use of the multipart/report content type, something that the MDN RFC 3798 builds upon.

3.4. RFC 1767 EDI Content [2]

This RFC defines the use of content type "application" for ANSI X12 (application/EDI-X12), EDIFACT (application/EDIFACT), and mutually defined EDI (application/EDI-Consent).

3.5. RFC 2045, 2046, and 2049 MIME [1]

These are the basic MIME standards, upon which all MIME related RFCs build, including this one. Key contributions include definitions of "content type", "sub-type", and "multipart", as well as encoding guidelines, which establish 7-bit US-ASCII as the canonical character set to be used in Internet messaging.

3.6. RFC 3798 Message Disposition Notification [5]

This Internet RFC defines how an MDN is requested, and the format and syntax of the MDN. The MDN is the basis upon which receipts and signed receipts are defined in this specification.

3.7. RFC 3851 and 3852 S/MIME Version 3.1 Message Specifications and Cryptographic Message Syntax (CMS) [7]

This specification describes how S/MIME will carry CMS Objects.

3.8. RFC 3023 XML Media Types [10]

This RFC defines the use of content type "application" for XML (application/xml).

4. Structure of an AS2 Message

4.1. Introduction

The basic structure of an AS2 message consists of MIME format inside an HTTP message with a few additional specific AS2 headers. The structures below are described hierarchically in terms of which RFCs are applied to form the specific structure. For details of how to code in compliance with all RFCs involved, turn directly to the RFCs referenced. Any difference between AS2 implantations and RFCs are mentioned specifically in the sections below.

4.2. Structure of an Internet EDI MIME Message

```
No encryption, no signature
   -RFC2616/2045
      -RFC1767/RFC3023 (application/EDIxxxx or /xml)

No encryption, signature
   -RFC2616/2045
     -RFC1847 (multipart/signed)
       -RFC1767/RFC3023 (application/EDIxxxx or /xml)
       -RFC3851 (application/pkcs7-signature)

Encryption, no signature
   -RFC2616/2045
     -RFC3851 (application/pkcs7-mime)
       -RFC1767/RFC3023  (application/EDIxxxx or /xml)(encrypted)

Encryption, signature
   -RFC2616/2045
     -RFC3851 (application/pkcs7-mime)
       -RFC1847 (multipart/signed)(encrypted)
         -RFC1767/RFC3023  (application/EDIxxxx or /xml)(encrypted)
         -RFC3851 (application/pkcs7-signature)(encrypted)

MDN over HTTP, no signature
   -RFC2616/2045
     -RFC3798 (message/disposition-notification)

MDN over HTTP, signature
   -RFC2616/2045
     -RFC1847 (multipart/signed)
      -RFC3798 (message/disposition-notification)
      -RFC3851 (application/pkcs7-signature)

MDN over SMTP, no signature
MDN over SMTP, signature
  Refer to the EDI over SMTP standard [4].
```

Although all MIME content types SHOULD be supported, the following MIME content types MUST be supported:

```
          Content-type: multipart/signed
          Content-Type: multipart/report
          Content-type: message/disposition-notification
          Content-Type: application/PKCS7-signature
          Content-Type: application/PKCS7-mime
          Content-Type: application/EDI-X12
```

```
Content-Type: application/EDIFACT
Content-Type: application/edi-consent
Content-Type: application/XML
```

5. HTTP Considerations

5.1. Sending EDI in HTTP POST Requests

The request line will have the form: "POST Request-URI HTTP/1.1", with spaces and followed by a CRLF. The Request URI is typically exchanged out of band, as part of setting up a bilateral trading partner agreement. Applications SHOULD be prepared to deal with an initial reply containing a status indicating a need for authentication of the usual types used for authorizing access to the Request-URI ([3], Section 10.4.2 and elsewhere).

The request line is followed by entity headers specifying content length ([3], Section 14.14) and content type ([3], Section 14.18). The Host request header ([3], Sections 9 and 14.23) is also included.

When using Transport Layer Security [15] or SSLv3, the request-URI SHOULD indicate the appropriate scheme value, HTTPS. Usually only a multipart/signed message body would be sent using TLS, as encrypted message bodies would be redundant. However, encrypted message bodies are not prohibited.

The receiving AS2 system MAY disconnect from the sending AS2 system before completing the reception of the entire entity if it determines that the entity being sent is too large to process.

For HTTP version 1.1, TCP persistent connections are the default, ([3] Sections 8.1.2, 8.2, and 19.7.1). A number of other differences exist because HTTP does not conform to MIME [1] as used in SMTP transport. Relevant differences are summarized below.

5.2. Unused MIME Headers and Operations

5.2.1. Content-Transfer-Encoding Not Used in HTTP Transport

HTTP can handle binary data and so there is no need to use the content transfer encodings of MIME [1]. This difference is discussed in [3], Section 19.4.5. However, a content transfer encoding value of binary or 8-bit is permissible but not required. The absence of this header MUST NOT result in transaction failure. Content transfer encoding of MIME bodyparts within the AS2 message body is also allowed.

5.2.2. Message Bodies

In [3], Section 3.7.2, it is explicitly noted that multiparts MUST have null epilogues.

In [4], Section 5.4.1, options for large file processing are discussed for SMTP transport. For HTTP, large files SHOULD be handled correctly by the TCP layer. However, in [3], Sections 3.5 and 3.6 discuss some options for compressing or chunking entities to be transferred. In [3], Section 8.1.2.2 discusses a pipelining option that is useful for segmenting large amounts of data.

5.3. Modification of MIME or Other Headers or Parameters Used

5.3.1. Content-Length

The use of the content-length header MUST follow the guidelines of [3], specifically Sections 4.4 and 14.13.

5.3.2. Final Recipient and Original Recipient

The final and original recipient values SHOULD be the same value. These values MUST NOT be aliases or mailing lists.

5.3.3. Message-Id and Original-Message-Id

Message-Id and Original-Message-Id is formatted as defined in RFC 2822 [9]:

"<" id-left "@" id-right ">" (RFC 2822, 3.6.4)

Message-Id length is a maximum of 998 characters. For maximum backward compatibility, Message-Id length SHOULD be 255 characters or less. Message-Id SHOULD be globally unique, and id-right SHOULD be something unique to the sending host environment (e.g., a host name).

When sending a message, always include the angle brackets. Angle brackets are not part of the Message-Id value. For maximum backward compatibility, when receiving a message, do not check for angle brackets. When creating the Original-Message-Id header in an MDN, always use the exact syntax as received on the original message; don’t strip or add angle brackets.

5.3.4. Host Header

The host request header field MUST be included in the POST request made when sending business data. This field is intended to allow one server IP address to service multiple hostnames, and potentially to conserve IP addresses. See [3], Sections 14.23 and 19.5.1.

5.4. HTTP Response Status Codes

The status codes return status concerning HTTP operations. For example, the status code 401, together with the WWW-Authenticate header, is used to challenge the client to repeat the request with an Authorization header. Other explicit status codes are documented in [3], Section 6.1.1 and throughout Section 10.

For errors in the request-URI, 400 ("Bad Request"), 404 ("Not Found"), and similar codes are appropriate status codes. These codes and their semantics are specified by [3]. A careful examination of these codes and their semantics should be made before implementing any retry functionality. Retries SHOULD NOT be made if the error is not transient or if retries are explicitly discouraged.

5.5. HTTP Error Recovery

If the HTTP client fails to read the HTTP server response data, the POST operation with identical content, including same Message-ID, SHOULD be repeated, if the condition is transient.

The Message-ID on a POST operation can be reused if and only if all of the content (including the original Date) is identical.

Details of the retry process (including time intervals to pause, number of retries to attempt, and timeouts for retrying) are implementation dependent. These settings are selected as part of the trading partner agreement.

Servers SHOULD be prepared to receive a POST with a repeated Message-ID. The MIME reply body previously sent SHOULD be resent, including the MDN and other MIME parts.

6. Additional AS2-Specific HTTP Headers

The following headers are to be included in all AS2 messages and all AS2 MDNs, except for asynchronous MDNs that are sent using SMTP and that follow the AS1 semantics[4].

6.1. AS2 Version Header

To promote backward compatibility, AS2 includes a version header:

AS2-Version: 1.0 - Used in all implementations of this specification. 1.x will be interpreted as 1.0 by all implementations with the "AS2 Version: 1.0" header. That is, only the most significant digit is used as the version identifier for those not implementing additional non-AS2-specified functionality. "AS2-Version: 1.0 through 1.9" MAY be used. All implementations MUST interpret "1.0 through 1.9" as implementing this specification. However, an implementation MAY extend this specification with additional functionality by specifying versions 1.1 through 1.9. If this mechanism is used, the additional functionality MUST be completely transparent to implementations with the "AS2-Version: 1.0" designation.

AS2-Version: 1.1 - Designates those implementations that support compression as defined by RFC 3274.

Receiving systems MUST NOT fail due to the absence of the AS2-Version header. Its absence would indicate that the message is from an implementation based on a previous version of this specification.

6.2. AS2 System Identifiers

To aid the receiving system in identifying the sending system, AS2-From and AS2-To headers are used.

```
AS2-From: < AS2-name >
AS2-To: < AS2-name >
```

These AS2 headers contain textual values, as described below, identifying the sender/receiver of a data exchange. Their values may be company specific, such as Data Universal Numbering System (DUNS) numbers, or they may be simply identification strings agreed upon between the trading partners.

```
AS2-text = "!" /           ; printable ASCII characters
           %d35-91 /       ; except double-quote (%d34)
           %d93-126        ; or backslash (%d92)

AS2-qtext = AS2-text / SP  ; allow space only in quoted text

AS2-quoted-pair = "\" DQUOTE /  ; \" or
                  "\" "\"       ; \\

AS2-quoted-name = DQUOTE 1*128( AS2-qtext /
                                AS2-quoted-pair) DQUOTE

AS2-atomic-name = 1*128AS2-text

AS2-name = AS2-atomic-name / AS2-quoted-name
```

The AS2-From header value and the AS2-To header value MUST each be an AS2-name, MUST each be comprised of from 1 to 128 printable ASCII characters, and MUST NOT be folded. The value in each of these headers is case-sensitive. The string definitions given above are in ABNF format [14].

The AS2-quoted-name SHOULD be used only if the AS2-name does not conform to AS2-atomic-name.

The AS2-To and AS2-From header fields MUST be present in all AS2 messages and AS2 MDNs whether asynchronous or synchronous in nature, except for asynchronous MDNs, which are sent using SMTP.

The AS2-name for the AS2-To header in a response or MDN MUST match the AS2-name of the AS2-From header in the corresponding request message. Likewise, the AS2-name for the AS2-From header in a response or MDN MUST match the AS2-name of the AS2-To header in the corresponding AS2 request message.

The sending system may choose to limit the possible AS2-To/AS2-From textual values but MUST not exceed them. The receiving system MUST make no restrictions on the textual values and SHOULD handle all possible implementations. However, implementers must be aware that older AS2 products may not adhere to this convention. Trading partner agreements should be made to ensure that older products can support the system identifiers that are used.

There is no required response to a client request containing invalid or unknown AS2-From or AS2-To header values. The receiving AS2 system MAY return an unsigned MDN with an explanation of the error, if the sending system requested an MDN.

7. Structure and Processing of an MDN Message

7.1. Introduction

In order to support non-repudiation of receipt, a signed receipt, based on digitally signing a message disposition notification, is to be implemented by a receiving trading partner's UA. The message disposition notification, specified by RFC 3798, is digitally signed by a receiving trading partner as part of a multipart/signed MIME message.

The following support for signed receipts is REQUIRED:

1. The ability to create a multipart/report; where the report-type = disposition-notification.

2. The ability to calculate a message integrity check (MIC) on the received message. The calculated MIC value will be returned to the sender of the message inside the signed receipt.

3. The ability to create a multipart/signed content with the message disposition notification as the first body part, and the signature as the second body part.

4. The ability to return the signed receipt to the sending trading partner.

5. The ability to return either a synchronous or an asynchronous receipt as the sending party requests.

The signed receipt is used to notify a sending trading partner that requested the signed receipt that:

1. The receiving trading partner acknowledges receipt of the sent EC Interchange.

2. If the sent message was signed, then the receiving trading partner has authenticated the sender of the EC Interchange.

3. If the sent message was signed, then the receiving trading partner has verified the integrity of the sent EC Interchange.

Regardless of whether the EDI/EC Interchange was sent in S/MIME format, the receiving trading partner's UA MUST provide the following basic processing:

1. If the sent EDI/EC Interchange is encrypted, then the encrypted symmetric key and initialization vector (if applicable) is decrypted using the receiver's private key.

2. The decrypted symmetric encryption key is then used to decrypt the EDI/EC Interchange.

3. The receiving trading partner authenticates signatures in a message using the sender's public key. The authentication algorithm performs the following:

   a. The message integrity check (MIC or Message Digest), is decrypted using the sender's public key.

   b. A MIC on the signed contents (the MIME header and encoded EDI object, as per RFC 1767) in the message received is calculated using the same one-way hash function that the sending trading partner used.

   c. The MIC extracted from the message that was sent and the MIC calculated using the same one-way hash function that the sending trading partner used are compared for equality.

4. The receiving trading partner formats the MDN and sets the calculated MIC into the "Received-content-MIC" extension field.

5. The receiving trading partner creates a multipart/signed MIME message according to RFC 1847.

6. The MDN is the first part of the multipart/signed message, and the digital signature is created over this MDN, including its MIME headers.

7. The second part of the multipart/signed message contains the digital signature. The "protocol" option specified in the second part of the multipart/signed is as follows:

   S/MIME: protocol = "application/pkcs-7-signature"

8. The signature information is formatted according to S/MIME specifications.

The EC Interchange and the RFC 1767 MIME EDI content header can actually be part of a multi-part MIME content-type. When the EDI Interchange is part of a multi-part MIME content-type, the MIC MUST be calculated across the entire multi-part content, including the MIME headers.

The signed MDN, when received by the sender of the EDI Interchange, can be used by the sender as follows:

- As an acknowledgement that the EDI Interchange sent was delivered and acknowledged by the receiving trading partner. The receiver does this by returning the original-message-id of the sent message in the MDN portion of the signed receipt.

- As an acknowledgement that the integrity of the EDI Interchange was verified by the receiving trading partner. The receiver does this by returning the calculated MIC of the received EC Interchange (and 1767 MIME headers) in the "Received-content-MIC" field of the signed MDN.

- As an acknowledgement that the receiving trading partner has authenticated the sender of the EDI Interchange.

- As a non-repudiation of receipt when the signed MDN is successfully verified by the sender with the receiving trading partner’s public key and the returned MIC value inside the MDN is the same as the digest of the original message.

7.2. Synchronous and Asynchronous MDNs

The AS2-MDN exists in two varieties: synchronous and asynchronous.

The synchronous AS2-MDN is sent as an HTTP response to an HTTP POST or as an HTTPS response to an HTTPS POST. This form of AS2-MDN is called synchronous because the AS2-MDN is returned to the originator of the POST on the same TCP/IP connection.

The asynchronous AS2-MDN is sent on a separate HTTP, HTTPS, or SMTP TCP/IP connection. Logically, the asynchronous AS2-MDN is a response to an AS2 message. However, at the transfer-protocol layer, assuming that no HTTP pipelining is utilized, the asynchronous AS2-MDN is delivered on a unique TCP/IP connection, distinct from that used to deliver the original AS2 message. When handling an asynchronous request, the HTTP response MUST be sent back before the MDN is processed and sent on the separate connection.

When an asynchronous AS2-MDN is requested by the sender of an AS2 message, the synchronous HTTP or HTTPS response returned to the sender prior to terminating the connection MUST be a transfer-layer response indicating the success or failure of the data transfer. The format of such a synchronous response MAY be the same as that response returned when no AS2-MDN is requested.

The following diagram illustrates the synchronous versus asynchronous varieties of AS2-MDN delivery using HTTP:

Synchronous AS2-MDN

```
[Peer1] ----( connect )----> [Peer2]
[Peer1] -----( send )------> [Peer2]   [HTTP Request [AS2-Message]]
[Peer1] <---( receive )----- [Peer2]   [HTTP Response [AS2-MDN]]
```

Asynchronous AS2-MDN

```
[Peer1] ----( connect )----> [Peer2]
[Peer1] -----( send )------> [Peer2]   [HTTP Request [AS2-Message]]
[Peer1] <---( receive )----- [Peer2]   [HTTP Response]

[Peer1]*<---( connect )----- [Peer2]
[Peer1] <--- ( send )------- [Peer2]   [HTTP Request [AS2-MDN]]
[Peer1] ----( receive )----> [Peer2]   [HTTP Response]
```

* Note: An AS2-MDN may be directed to a host different from that of the sender of the AS2 message. It may utilize a transfer protocol different from that used to send the original AS2 message.

The advantage of the synchronous MDN is that it can provide the sender of the AS2 Message with a verifiable confirmation of message delivery within a synchronous logic flow. However, if the message is relatively large, the time required to process this message and to return an AS2-MDN to the sender on the same TCP/IP connection may exceed the maximum configured time permitted for an IP connection.

The advantage of the asynchronous MDN is that it provides for the rapid return of a transfer-layer response from the receiver, confirming the receipt of data, therefore not requiring that a TCP/IP connection necessarily remain open for very long. However, this design requires that the asynchronous AS2-MDN contain enough information to identify the original message uniquely so that, when received by the AS2 Message originator, the status of the original AS2 Message can be properly updated based on the contents of the AS2-MDN.

Synchronous or asynchronous HTTP or HTTPS MDNs are handled according to the requirements of this specification.

However, SMTP MDNs are formatted according to the requirements of RFC 3335 [4].

7.3. Requesting a Signed Receipt

Message disposition notifications are requested as per RFC 3798. A request that the receiving user agent issue a message disposition notification is made by placing the following header into the message to be sent:

```
MDN-request-header = "Disposition-notification-to"
                     ":"  mail-address
```

The following example is for requesting an MDN:

```
Disposition-notification-to: xxx@example.com
```

This syntax is a residue of the use of MDNs using SMTP transfer. Because this specification is adjusting the functionality from SMTP to HTTP while retaining as much as possible from the [4] functionality, the mail-address MUST be present. The mail-address field is specified as an RFC 2822 localpart@domain [addr-spec] address. However, the address is not used to identify where to return the MDN. Receiving applications MUST ignore the value and MUST not complain about RFC 2822 address syntax violations.

When requesting MDN-based receipts, the originator supplies additional extension headers that precede the message body. These header "tags" are as follows:

A Message-ID header is added to support message reconciliation, so that an Original-Message-Id value can be returned in the body part of MDN. Other headers, especially "Subject" and "Date", SHOULD be supplied; the values of these headers are often mentioned in the human-readable section of a MDN to aid in identifying the original message.

MDNs will be returned in the HTTP response when requested, unless an asynchronous return is requested.

To request an asynchronous message disposition notification, the following header is placed into the message that is sent:

```
Receipt-Delivery-Option: return-URL
```

Here is an example requesting that the MDN be asynchronous:

```
Receipt-Delivery-Option: http://www.example.com/Path
```

Receipt-delivery-option syntax allows return-url to use some schemes other than HTTP using the POST method.

The "receipt-delivery-option: return-url" string indicates the URL to use for an asynchronous MDN. This header is NOT present if the receipt is to be synchronous. The email value in Disposition-notification-to is not used in this specification because it was limited to RFC 2822 addresses; the extension header "Receipt-delivery-option" has been introduced to provide a URL for the MDN return by several transfer options.

The receipt-delivery-option's value MUST be a URL indicating the delivery transport destination for the receipt.

An example request for an asynchronous MDN via an HTTP transport:

```
Receipt-delivery-option: http://www.example.com
```

An example request for an asynchronous MDN via an HTTP/S transport:

```
Receipt-delivery-option: https://www.example.com
```

An example request for an asynchronous MDN via an SMTP transport:

```
Receipt-delivery-option: mailto:as2@example.com
```

For more information on requesting SMTP MDNs, refer to RFC 3335 [4].

Finally, the header, Disposition-notification-options, identifies characteristics of message disposition notification as in [5]. The most important of these options is for indicating the signing options for the MDN, as in the following example:

```
Disposition-notification-options:
     signed-receipt-protocol=optional,pkcs7-signature;
     signed-receipt-micalg=optional,sha1,md5
```

For signing options, consider the disposition-notification-options syntax:

```
Disposition-notification-options =
          "Disposition-Notification-Options" ":"
           disposition-notification-parameters
```

```
    where
             disposition-notification-parameters =
                              parameter *(";" parameter)

    where
             parameter = attribute "=" importance ", " 1#value"

    where
             importance = "required" | "optional"
```

So the Disposition-notification-options string could be:

```
     signed-receipt-protocol=optional,<protocol symbol>;
     signed-receipt-micalg=optional,<micalg1>,<micalg2>,...;
```

The currently used value for <protocol symbol> is "pkcs7-signature" for the S/MIME detached signature format.

The currently supported values for MIC algorithm <micalg> values are:

Algorithm	Value Used
SHA-1	sha1
MD5	md5

The semantics of the "signed-receipt-protocol" and the "signed-receipt-micalg" parameters are as follows:

1. The "signed-receipt-protocol" parameter is used to request a signed receipt from the recipient trading partner. The "signed-receipt-protocol" parameter also specifies the format in which the signed receipt SHOULD be returned to the requester.

   The "signed-receipt-micalg" parameter is a list of MIC algorithms preferred by the requester for use in signing the returned receipt. The list of MIC algorithms SHOULD be honored by the recipient from left to right.

   Both the "signed-receipt-protocol" and the "signed- receipt-micalg" option parameters are REQUIRED when requesting a signed receipt.

   The lack of the presence of the "Receipt-Delivery-Option" indicates that a receipt is synchronous in nature. The presence of the "Receipt-Delivery-Option: return-url" indicates that an asynchronous receipt is requested and SHOULD be sent to the "return-url".

2. The "importance" attribute of "Optional" is defined in RFC 3798, Section 2.2, and has the following meaning:

   Parameters with an importance of "Optional" permit a UA that does not understand the particular options parameter to still generate an MDN in response to a request for a MDN.

   A UA that does not understand the "signed-receipt-protocol" parameter or the "signed-receipt-micalg" will obviously not return a signed receipt.

   The importance of "Optional" is used for the signed receipt parameters because it is RECOMMENDED that an MDN be returned to the requesting trading partner even if the recipient could not sign it.

   The returned MDN will contain information on the disposition of the message and on why the MDN could not be signed. See the Disposition field in Section 7.5 for more information.

   Within an EDI trading relationship, if a signed receipt is expected and is not returned, then the validity of the transaction is up to the trading partners to resolve.

   In general, if a signed receipt is required in the trading relationship and is not received, the transaction will likely not be considered valid.

7.3.1. Signed Receipt Considerations

The method used to request a receipt or a signed receipt is defined in RFC 3798, "An Extensible Message Format for Message Disposition Notifications".

The "rules" are as follows:

1. When a receipt is requested, explicitly specifying that the receipt be signed, then the receipt MUST be returned with a signature.

2. When a receipt is requested, explicitly specifying that the receipt be signed, but the recipient cannot support either the requested protocol format or the requested MIC algorithms, then either a signed or unsigned receipt SHOULD be returned.

3. When a signature is not explicitly requested, or if the signed receipt request parameter is not recognized by the UA, then no receipt, an unsigned receipt, or a signed receipt MAY be returned by the recipient.

NOTE: For Internet EDI, it is RECOMMENDED that when a signature is not explicitly requested, or if parameters are not recognized, the UA send back, at a minimum, an unsigned receipt. If, however, a signed receipt was always returned as a policy, whether requested or not, then any false unsigned receipts can be repudiated.

When a request for a signed receipt is made, but there is an error in processing the contents of the message, a signed receipt MUST still be returned. The request for a signed receipt SHALL still be honored, though the transaction itself may not be valid. The reason why the contents could not be processed MUST be set in the "disposition-field".

When a signed receipt request is made, the "Received-content-MIC" MUST always be returned to the requester (except when corruption prevents computation of the digest in accordance with the following specification). The "Received-content-MIC" MUST be calculated as follows:

- For any signed messages, the MIC to be returned is calculated on the RFC1767/RFC3023 MIME header and content. Canonicalization on the MIME headers MUST be performed before the MIC is calculated, since the sender requesting the signed receipt was also REQUIRED to canonicalize.

- For encrypted, unsigned messages, the MIC to be returned is calculated on the decrypted RFC 1767/RFC3023 MIME header and content. The content after decryption MUST be canonicalized before the MIC is calculated.

- For unsigned, unencrypted messages, the MIC MUST be calculated over the message contents without the MIME or any other RFC 2822 headers, since these are sometimes altered or reordered by Mail Transport Agents (MTAs).

7.4. MDN Format and Values

This section defines the format of the AS2 Message Disposition Notification (AS2-MDN).

7.4.1. AS2-MDN General Formats

The AS2-MDN follows the MDN specification [5] except where noted in this section. The modified ABNF definitions in this document use the vertical-bar character, '|', to denote a logical "OR" construction. This usage follows RFC 2616 [3]. HTTP entities referred to below are not further defined in this document. Refer to RFC 2616 [3] for complete definitions of HTTP entities. The format of the AS2-MDN is:

```
AS2-MDN = AS2-sync-MDN | AS2-async-http-MDN |
    AS2-async-smtp-MDN

AS2-sync-MDN =
    Status-Line
    *(( general-header | response-header | entity-header )
    CRLF )
    CRLF
    AS2-MDN-body

Status-Line =
    HTTP-Version SP Status-Code SP Reason-Phrase CRLF

AS2-async-http-MDN =
    Request-Line
    *(( general-header | request-header | entity-header )
    CRLF )
    CRLF
    AS2-MDN-body

Request-Line =
    Method SP Request-URI SP HTTP-Version CRLF

AS2-async-smtp-MDN =
    *(( general-header | request-header | entity-header )
    CRLF )
    CRLF
    AS2-MDN-body

AS2-MDN-body =
    AS2-signed-MDN-body | AS2-unsigned-MDN-body
```

7.4.2. AS2-MDN Construction

The AS2-MDN-body is formatted as a MIME multipart/report with a report-type of "disposition-notification". When the message is unsigned, the transfer-layer ("outermost") entity-headers of the AS2-MDN contain the content-type header that specifies a content-type

of "multipart/report" and parameters indicating the report-type, and the value of the outermost multipart boundary.

When the AS2-MDN is signed, the transfer-layer ("outermost") entity-headers of the AS2-MDN contain a content-type header that specifies a content-type of "multipart/signed" and parameters indicating the algorithm used to compute the message digest, the signature-formatting protocol (e.g., pkcs7-signature), and the value of the outermost multipart boundary. The first part of the MIME multipart/signed message is an embedded MIME multipart/report of type "disposition-notification". The second part of the multipart/signed message contains a MIME application/pkcs7-signature message.

The first part of the MIME multipart/report is a "human-readable" portion that contains a general description of the message disposition. The second part of the MIME multipart/report is a "machine-readable" portion that is defined as:

```
AS2-disposition-notification-content =
    [ reporting-ua-field CRLF ]
    [ mdn-gateway-field CRLF ]
    final-recipient-field CRLF
    [ original-message-id-field CRLF ]
    AS2-disposition-field CRLF
    *( failure-field CRLF )
    *( error-field CRLF )
    *( warning-field CRLF )
    *( extension-field CRLF )
    [ AS2-received-content-MIC-field CRLF ]
```

7.4.3. AS2-MDN Fields

The rules for constructing the AS2-disposition-notification content are identical to the disposition-notification-content rules provided in Section 7 of RFC 3798 [5], except that the RFC 3798 disposition-field has been replaced with the AS2-disposition-field and that the AS2-received-content-MIC field has been added. The differences between the RFC 3798 disposition-field and the AS2-disposition-field are described below. Where there are differences between this document and RFC 3798, those entity names have been changed by pre-pending "AS2-". Entities that do not differ from RFC 3798 are not necessarily further defined in this document; refer to RFC 3798, Section 7, "Collected Grammar", for the original grammar.

```
AS2-disposition-field =
    "Disposition" ":" disposition-mode ";"
    AS2-disposition-type [ '//' AS2-disposition-modifier ]

disposition-mode =
    action-mode "/" sending-mode

action-mode =
    "manual-action" | "automatic-action"

sending-mode =
    "MDN-sent-manually" | "MDN-sent-automatically"

AS2-disposition-type =
    "processed" | "failed"

AS2-disposition-modifier =
    ( "error" | "warning" ) | AS2-disposition-modifier-extension

AS2-disposition-modifier-extension =
    "error: authentication-failed" |
    "error: decompression-failed" |
    "error: decryption-failed" |
    "error: insufficient-message-security" |
    "error: integrity-check-failed" |
    "error: unexpected-processing-error" |
    "warning: " AS2-MDN-warning-description |
    "failure: " AS2-MDN-failure-description

AS2-MDN-warning-description = *( TEXT )

AS2-MDN-failure-description = *( TEXT )

AS2-received-content-MIC-field =
    "Received-content-MIC" ":" encoded-message-digest ","
    digest-alg-id CRLF

encoded-message-digest =
    1*( 'A'-Z' | 'a'-'z' | '0'-'9' | '/' | '+' | '=' )  (
    i.e. base64( message-digest ) )

digest-alg-id = "sha1" | "md5"
```

"Insufficient-message-security" and "decompression-failed" are new error codes that are not mentioned in the AS1 RFC 3335, and may not be compatible with earlier implementations of AS2.

The "Received-content-MIC" extension field is set when the integrity of the received message is verified. The MIC is the base64-encoded message-digest computed over the received message with a hash function. This field is required for signed receipts but optional for unsigned receipts. For details defining the specific content over which the message digest is to be computed, see Section 7.3.1 of this document.

For signed messages, the algorithm used to calculate the MIC MUST be the same as that used on the message that was signed. If the message is not signed, then the SHA-1 algorithm SHOULD be used. This field is set only when the contents of the message are processed successfully. This field is used in conjunction with the recipient's signature on the MDN so that the sender can verify non-repudiation of receipt.

AS2-MDN field names (e.g., "Disposition:", "Final-Recipient:") are case insensitive (cf. RFC 3798, Section 3.1.1). AS2-MDN action-modes, sending-modes, AS2-disposition-types, and AS2-disposition-modifier values, which are defined above, and user-supplied *(TEXT) values are also case insensitive. AS2 implementations MUST NOT make assumptions regarding the values supplied for AS2-MDN-warning-description or AS2-MDN-failure-description, or for the values of any (optional) error, warning, or failure fields.

7.4.4. Additional AS2-MDN Programming Notes

- Unlike SMTP, for HTTP transactions, Original-Recipient and Final-Recipient SHOULD not be different. The value in Original-Message-ID SHOULD match the original Message-ID header value.

- Refer to RFC 3798 for the formatting of the MDN, except for the specific deviations mentioned above.

- Refer to RFC 3462 and RFC 3798 for the formatting of the content-type entity-headers for the MDN.

- Use an action-mode of "automatic-action" when the disposition described by the disposition type was a result of an automatic action rather than that of an explicit instruction by the user for this message.

- Use an action-mode of "manual-action" when the disposition described by the disposition type was a result of an explicit instruction by the user rather than some sort of automatically performed action.

- Use a sending-mode of "MDN-sent-automatically" when the MDN is sent because the UA had previously been configured to do so.

- Use a sending-mode of "MDN-sent-manually" when the user explicitly gave permission for this particular MDN to be sent.

- The sending-mode "MDN-sent-manually" is meaningful ONLY with "manual-action", not with "automatic-action".

- The "failed" disposition type MUST NOT be used for the situation in which there is some problem in processing the message other than interpreting the request for an MDN. The "processed" or other disposition type with appropriate disposition modifiers is to be used in such situations.

7.5. Disposition Mode, Type, and Modifier

7.5.1. Disposition Mode Overview

This section provides a brief overview of how "processed", "error", "failure", and "warning" are used.

7.5.2. Successful Processing Status Indication

When the request for a receipt or signed receipt, and the received message contents are successfully processed by the receiving EDI UA, a receipt or MDN SHOULD be returned with the disposition-type set to "processed". When the MDN is sent automatically by the EDI UA, and there is no explicit way for a user to control the sending of the MDN, then the first part of the "disposition-mode" SHOULD be set to "automatic-action". When the MDN is being sent under user-configurable control, then the first part of the "disposition-mode" SHOULD be set to "manual-action". Since a request for a signed receipt should always be honored, the user MUST not be allowed to configure the UA not to send a signed receipt when the sender requests one.

The second part of the disposition-mode is set to "MDN-sent-manually" if the user gave explicit permission for the MDN to be sent. Again, the user MUST not be allowed to explicitly refuse to send a signed receipt when the sender requests one. The second part of the "disposition-mode" is set to "MDN-sent-automatically" whenever the EDI UA sends the MDN automatically, regardless of whether the sending was under the control of a user, administrator, or software.

Because EDI content is generally handled automatically by the EDI UA, a request for a receipt or signed receipt will generally return the following in the "disposition-field":

Disposition: automatic-action/MDN-sent-automatically; processed

Note that this specification does not restrict the use of the "disposition-mode" just to automatic actions. Manual actions are valid as long as it is kept in mind that a request for a signed receipt MUST be honored.

7.5.3. Unsuccessful Processed Content

The request for a signed receipt requires the use of two "disposition-notification-options", which specify the protocol format of the returned signed receipt, and the MIC algorithm used to calculate the MIC over the message contents. The "disposition-field" values that should be used if the message content is being rejected or ignored (for instance, if the EDI UA determines that a signed receipt cannot be returned because it does not support the requested protocol format, the EDI UA chooses not to process the message contents itself) MUST be specified in the MDN "disposition-field" as follows:

Disposition: "disposition-mode"; failed/Failure:
unsupported format

The "failed" AS2-disposition-type MUST be used when a failure occurs that prevents the proper generation of an MDN. For example, this disposition-type would apply if the sender of the message requested the application of an unsupported message-integrity-check (MIC) algorithm.

The "failure:" AS2-disposition-modifier-extension SHOULD be used with an implementation-defined description of the failure. Further information about the failure may be contained in a failure-field.

The syntax of the "failed" disposition-type is general, allowing the sending of any textual information along with the "failed" disposition-type. Implementations MUST support any printable textual characters after the Failure disposition-type. For use in Internet EDI, the following "failed" values are pre-defined and MUST be supported:

"Failure: unsupported format"

"Failure: unsupported MIC-algorithms"

7.5.4. Unsuccessful Non-Content Processing

When errors occur in processing the received message (other than content), the "disposition-field" MUST be set to the "processed" value for disposition-type and the "error" value for disposition-modifier.

The "error" AS2-disposition-modifier with the "processed" disposition-type MUST be used to indicate that an error of some sort occurred that prevented successful processing of the message. Further information may be contained in an error-field.

An "error:" AS2-disposition-modifier-extension SHOULD be used to combine the indication of an error with a predefined description of a specific, well-known error. Further information about the error may be contained in an error field.

For internet EDI use, the following "error" AS2-disposition-modifier values are defined:

- o "Error: decryption-failed" - the receiver could not decrypt the message contents.
- o "Error: authentication-failed" - the receiver could not authenticate the sender.
- o "Error: integrity-check-failed" - the receiver could not verify content integrity.
- o "Error: unexpected-processing-error" - a catch-all for any additional processing errors.

An example of how the "disposition-field" would look when errors other than those in content processing are detected is as follows:

```
    Disposition: "disposition-mode"; processed/Error:
      decryption-failed
```

7.5.5. Processing Warnings

Situations arise in EDI when, even if a trading partner cannot be authenticated correctly, the trading partners still agree to continue processing the EDI transactions. Transaction reconciliation is done between the trading partners at a later time. In the content

processing warning situations as described above, the "disposition-field" MUST be set to the "processed" disposition-type value, and the "warning" to the "disposition-modifier" value.

The "warning" AS2-disposition-modifier MUST be used with the "processed" disposition-type to indicate that the message was successfully processed but that an exceptional condition occurred. Further information may be contained in a warning-field.

A "warning:" AS2-disposition-modifier-extension SHOULD be used to combine the indication of a warning with an implementation-defined description of the warning. Further information about the warning may be contained in a warning-field.

For use in Internet EDI, the following "warning" disposition-modifier-extension value is defined:

"Warning: authentication-failed, processing continued"

An example of how the "disposition-field" would look when warning other than those for content processing are detected is as follows:

Example:

Disposition: "disposition-mode"; processed/Warning:
authentication-failed, processing continued

7.5.6. Backward Compatibility with Disposition Type, Modifier, and Extension

The following set of examples represents typical constructions of the Disposition field that have been in use by AS2 implementations. This is NOT an exhaustive list of possible constructions. However, AS2 implementations MUST accept constructions of this type to be backward compatible with earlier AS2 versions.

Disposition: automatic-action/MDN-sent-automatically; processed

Disposition: automatic-action/MDN-sent-automatically;
processed/error: authentication-failed

Disposition: automatic-action/MDN-sent-automatically;
processed/warning: duplicate-document

Disposition: automatic-action/MDN-sent-automatically;
failed/failure: sender-equals-receiver

The following set of examples represents allowable constructions of the Disposition field that combine the historic constructions above with optional RFC 3798 error, warning, and failure fields. AS2 implementations MAY produce these constructions. However, AS2 servers are not required to recognize or process optional error, warning, or failure fields at this time. Note that the use of the multiple error fields in the second example below provides for the indication of multiple error conditions.

```
Disposition: automatic-action/MDN-sent-automatically; processed

Disposition: automatic-action/MDN-sent-automatically;
  processed/error: decryption-failed
Error: The signature did not decrypt into a valid PKCS#1
  Type-2 block.
Error: The length of the decrypted key does not equal the
  octet length of the modulus.

Disposition: automatic-action/MDN-sent-automatically;
  processed/warning: duplicate-document
Warning: An identical message already exists at the
  destination server.

Disposition: automatic-action/MDN-sent-automatically;
  failed/failure: sender-equals-receiver
Failure: The AS2-To name is identical to the AS2-From name.
```

The following set of examples represents allowable constructions of the Disposition field that employ pure RFC 3798 Disposition-modifiers with optional error, warning, and failure fields. These examples are provided as informational only. These constructions are not guaranteed to be backward compatible with AS2 implementations prior to version 1.1.

```
Disposition: automatic-action/MDN-sent-automatically; processed

Disposition: automatic-action/MDN-sent-automatically;
  processed/error
Error: authentication-failed
Error: The signature did not decrypt into a valid PKCS#1 Type-2
  block.
Error: The length of the decrypted key does not equal the
  octet length of the modulus.

Disposition: automatic-action/MDN-sent-automatically;
  processed/warning
Warning: duplicate-document
```

```
Disposition: automatic-action/MDN-sent-automatically; failed
Failure: sender-equals-receiver
```

7.6. Receipt Reply Considerations in an HTTP POST

The details of the response to the POST command vary depending upon whether a receipt has been requested.

With no extended header requesting a receipt, and with no errors accessing the request-URI specified processing, the status line in the Response to the POST request SHOULD be in the 200 range. Status codes in the 200 range SHOULD also be used when an entity is returned (a signed receipt in a multipart/signed content type or an unsigned receipt in a multipart/report). Even when the disposition of the data was an error condition at the authentication, decryption or other higher level, the HTTP status code SHOULD indicate success at the HTTP level.

The HTTP server-side application may respond with an unsolicited multipart/report as a message body that the HTTP client might not have solicited, but the client may discard this. Applications SHOULD avoid emitting unsolicited receipt replies because bandwidth or processing limitations might have led administrators to suspend asking for acknowledgements.

Message Disposition Notifications, when used in the HTTP reply context, will closely parallel a SMTP MDN. For example, the disposition field is a required element in the machine-readable second part of a multipart/report for a MDN. The final-recipient-field ([5], Section 3.1) value SHOULD be derived from the entity headers of the request.

In an MDN, the first part of the multipart/report (the human-readable part) SHOULD include items such as the subject, the date, and other information when those fields are present in entity header fields following the POST request. An application MUST report the Message-ID of the request in the second part of the multipart/report (the machine-readable part). Also, an MDN SHOULD have its own unique Message-ID HTTP header. The HTTP reply SHOULD normally omit the third optional part of the multipart/report (used to return the original message or its headers in the SMTP context).

8. Public Key Certificate Handling

In the near term, the exchange of public keys and certification of these keys MUST be handled as part of the process of establishing a trading partnership. The UA and/or EDI application interface must maintain a database of public keys used for encryption or signatures,

in addition to the mapping between the EDI trading partner ID and the RFC 2822 [9] email address and HTTP URL/URI. The procedures for establishing a trading partnership and configuring the secure EDI messaging system might vary among trading partners and software packages.

X.509 certificates are REQUIRED. It is RECOMMENDED that trading partners self-certify each other if an agreed-upon certification authority is not used. This applicability statement does NOT require the use of a certification authority. The use of a certification authority is therefore OPTIONAL. Certificates may be self-signed.

It is RECOMMENDED that when trading partners are using S/MIME they also exchange public key certificates, considering advice provided in [12].

The message formats useful for certificate exchange are found in [7] and [13].

In the long term, additional standards may be developed to simplify the process of establishing a trading partnership, including the third-party authentication of trading partners, as well as the attributes of the trading relationship.

9. Security Considerations

This entire document is concerned with secure transport of business to business data, and it considers both data confidentiality and authentication issues.

Extracted from RFC 3851 [7]:
40-bit encryption is considered weak by most cryptographers. Using weak cryptography in S/MIME offers little actual security over sending plaintext. However, other features of S/MIME, such as the specification of Triple DES and the ability to announce stronger cryptographic capabilities to parties with whom you communicate, allow senders to create messages that use strong encryption. Using weak cryptography is never recommended unless the only alternative is no cryptography. When feasible, sending and receiving agents SHOULD inform senders and recipients of the relative cryptographic strength of messages.

Extracted from RFC 3850 [12]:
When processing certificates, there are many situations where the processing might fail. Because the processing may be done by a user agent, a security gateway, or other program, there is no single way to handle such failures. Just because the methods to handle the failures have not been listed, however, the reader should not assume

that they are not important. The opposite is true: if a certificate is not provably valid and associated with the message, the processing software should take immediate and noticeable steps to inform the end user about it.

Some of the many situations in which signature and certificate checking might fail include the following:

- No certificate chain leads to a trusted CA.
- No ability to check the Certificate Revocation List (CRL) for a certificate.
- An invalid CRL was received.
- The CRL being checked is expired.
- The certificate is expired.
- The certificate has been revoked.

There are certainly other instances where a certificate may be invalid, and it is the responsibility of the processing software to check them all thoroughly, and to decide what to do if the check fails. See RFC 3280 for additional information on certificate path validation.

The following are additional security considerations to those listed in [7] and [12].

9.1. NRR Cautions

This specification seeks to provide multiple mechanisms that can be combined in accordance with local policies to achieve a wide range of security needs as determined by threat and risk analyses of the business peers. It is required that all these mechanisms be implemented by AS2 software so that the software has capabilities that promote strong interoperability, no matter what policies are adopted.

One strong cluster of mechanisms (the secure transmission loop) can provide good support for meeting the evidentiary needs of non-repudiation of receipt by the original sender and by a third party supplied with all stated evidence. However, this specification does not itself define non-repudiation of receipt nor enumerate its essential properties because NRR is a business analysis and/or legal requirement, and not relevantly defined by a technical applicability statement.

Some analyses observe that non-repudiation of receipt presupposes that non-repudiation of the sender of the original message is obtained, and further that non-repudiation should be implemented by means of digital signature on the original message. To satisfy strict NRR evidence, authentication and integrity MUST be provided by some mechanism, and the RECOMMENDED mechanism is digital signatures on both the original message and the receipt message.

Given that this specification has selected several mechanisms that can be combined in several ways, it is important to realize that if a digital signature is omitted from the original message, in order to satisfy the preceding analysis of NRR requirements, some authentication mechanism MUST accompany the request for a signed receipt and its included Received-content-MIC value. This authentication might come from using client-side SSL, authentication via IPsec, or HTTP authentication (while using SSL). In any case, records of the message content, its security basis, and the digest value need to be retained for the NRR process.

Therefore, if NRR is one of the goals of the policy that is adopted, by using the mechanisms of the secure transmission loop mentioned above and by retaining appropriate records of authentication at the original message sender site, strong evidentiary requirements proposed for NRR can be fulfilled.

Other ways of proceeding may fall short of fulfilling the most stringent sets of evidence required for NRR to obtain, but may nevertheless be part of a commercial trading agreement and, as such, are good enough for the parties involved. However, if MDNs are returned unsigned, evidentiary requirements for NRR are weak; some authentication of the identity of the receiver is needed.

9.2. HTTPS Remark

The following certificate types MUST be supported for SSL server-side certificates:

- with URL in the Distinguished Name Common Name attribute
- without URL in the Distinguished Name Common Name attribute
- self-signed (self-issued)
- certification authority certified

The URL, which matches the source server identity, SHOULD be carried in the certificate. However, it is not required that DNS checks or reverse lookups to vouch for the accuracy of the URL or server value.

Because server-side certificates are exchanged, and also trust is established during the configuration of the trading partner relationship, runtime checks are not required by implementations of this specification.

The complete certification chain MUST be included in all certificates. All certificate verifications MUST "chain to root" or to an accepted trust anchor. Additionally, the certificate hash SHOULD match the hash recomputed by the receiver.

9.3. Replay Remark

Because business data documents normally contain transaction ids, replays (such as resends of not-yet-acknowledged messages) are discarded as part of the normal process of duplicate detection. Detection of duplicates by Message-Id or by business transaction identifiers is recommended.

10. IANA Considerations

RFC 3335 registered two Disposition-Notification-Options parameters

```
Parameter-name: signed-receipt-protocol
Parameter-name: signed-receipt-micalg
```

that are also used by this specification (see Section 7.3).

RFC 3335 also registered on MDN Extension field name

```
Extension field name: Received-content-MIC
```

that is also used by this specification (see Section 7.4.3). Registration of the above is therefore NOT needed.

10.1. Registration

This specification defines an extension to the Message Disposition Notification (MDN) protocol for a disposition-modifier in the Disposition field of a body of content-type "message/disposition-notification".

10.1.1. Disposition Modifier 'warning'

Parameter-name: warning
Semantics: See Sections 7.4.3 and 7.5.5 of this document.

11. Acknowledgements

Carl Hage, Karen Rosenfeld, Chuck Fenton, and many others have provided valuable suggestions that improved this applicability statement. The authors would also like to thank the vendors who participated in the Drummond Group Inc. AS2 interoperability testing. Their contributions led to great improvement in the clarity of this document.

12. References

12.1. Normative References

[1] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, November 1996.

[2] Crocker, D., "MIME Encapsulation of EDI Objects", RFC 1767, March 1995.

[3] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[4] Harding, T., Drummond, R., and C. Shih, "MIME-based Secure Peer-to-Peer Business Data Interchange over the Internet", RFC 3335, September 2002.

[5] Hansen, T. and G. Vaudreuil, "Message Disposition Notification", RFC 3798, May 2004.

[6] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[7] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[8] Vaudreuil, G., "The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages", RFC 3462, January 2003.

[9] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[10] Murata, M., Laurent, S. St., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[11] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[12] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling", RFC 3850, July 2004.

[13] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[14] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

12.2. Informative References

[15] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

Appendix A: Message Examples

NOTE: All examples are provided for illustration only, and are not considered part of the protocol specification. If an example conflicts with the protocol definitions specified above or in the other referenced RFCs, the example is wrong.

A.1. Signed Message Requesting a Signed, Synchronous Receipt

```
POST /receive HTTP/1.0
Host: 10.234.160.12:80
User-Agent: AS2 Company Server
Date: Wed, 31 Jul 2002 13:34:50 GMT
From: mrAS2@example.com
AS2-Version: 1.1
AS2-From: "\"  as2Name  \""
AS2-To: 0123456780000
Subject: Test Case
Message-Id: <200207310834482A70BF63@\"~~foo~~\">
Disposition-Notification-To: mrAS2@example.com
Disposition-Notification-Options: signed-receipt-protocol=optional,
  pkcs7-signature; signed-receipt-micalg=optional,sha1
Content-Type: multipart/signed; boundary="as2BouNdary1as2";
  protocol="application/pkcs7-signature"; micalg=sha1
Content-Length: 2464

--as2BouNdary1as2
Content-Type: application/edi-x12
Content-Disposition: Attachment; filename=rfc1767.dat
  [ISA ...EDI transaction data...IEA...]

--as2BouNdary1as2
Content-Type: application/pkcs7-signature

  [omitted binary pkcs7 signature data]
--as2BouNdary1as2--
```

A.2. MDN for Message A.1, Above

```
HTTP/1.0 200 OK
AS2-From: 0123456780000
AS2-To: "\"  as2Name  \""
AS2-Version: 1.1
Message-ID: <709700825.1028122454671.JavaMail@ediXchange>
Content-Type: multipart/signed; micalg=sha1;
     protocol="application/pkcs7-signature";
     boundary="----=_Part_57_648441049.1028122454671"
Connection: Close
```

```
   Content-Length: 1980

   ------=_Part_57_648441049.1028122454671

   & Content-Type: multipart/report;
   & Report-Type=disposition-notification;
   &    boundary="----=_Part_56_1672293592.1028122454656"
   &
   &------=_Part_56_1672293592.1028122454656
   &Content-Type: text/plain
   &Content-Transfer-Encoding: 7bit
   &
   &MDN for -
   & Message ID: <200207310834482A70BF63@\"~~foo~~\">
   &  From: "\"  as2Name  \""
   &  To: "0123456780000"
   &  Received on: 2002-07-31 at 09:34:14 (EDT)
   & Status: processed
   & Comment: This is not a guarantee that the message has
   &  been completely processed or &understood by the receiving
   &  translator
   &
   &------=_Part_56_1672293592.1028122454656
   &Content-Type: message/disposition-notification
   &Content-Transfer-Encoding: 7bit
   &
   &Reporting-UA: AS2 Server
   &Original-Recipient: rfc822; 0123456780000
   &Final-Recipient: rfc822; 0123456780000
   &Original-Message-ID: <200207310834482A70BF63@\"~~foo~~\">
   &Received-content-MIC: 7v7F++fQaNB1sVLFtMRp+dF+eG4=, sha1
   &Disposition: automatic-action/MDN-sent-automatically;
   &  processed
   &
   &------=_Part_56_1672293592.1028122454656--

   ------=_Part_57_648441049.1028122454671
   Content-Type: application/pkcs7-signature; name=smime.p7s
   Content-Transfer-Encoding: base64
   Content-Disposition: attachment; filename=smime.p7s

   MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQ
   cp24hMJNbxDKHnlB9jTiQzLwSwo+/90Pc87x+Sc6EpFSUYWGAAAAAAAA
   ------=_Part_57_648441049.1028122454671--
```

Notes:

1. The lines proceeded with "&" are what the signature is calculated over.

2. For details on how to prepare the multipart/signed with protocol = "application/pkcs7-signature", see the "S/MIME Message Specification, PKCS Security Services for MIME".)

3. Note that the textual first body part of the multipart/report can be used to include a more detailed explanation of the error conditions reported by the disposition headers. The first body part of the multipart/report, when used in this way, allows a person to better diagnose a problem in detail.

4. As specified by RFC 3462 [8], returning the original or portions of the original message in the third body part of the multipart/report is not required. This is an optional body part. However, it is RECOMMENDED that this body part be omitted or left blank.

A.3. Signed, Encrypted Message Requesting a Signed, Asynchronous Receipt

```
Message-ID: <#as2_company#01#a4260as2_companyout#>
Date: Thu, 19 Dec 2002 15:04:18 GMT
From: me@example.com
Subject: Async MDN request
Mime-Version: 1.0
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename=smime.p7m
Recipient-Address: 10.240.1.2//
Disposition-Notification-To:
  http://10.240.1.2:8201/exchange/as2_company
Disposition-Notification-Options: signed-receipt-protocol=optional,
 pkcs7-signature; signed-receipt-micalg=optional,sha1
Receipt-Delivery-Option:
  http://10.240.1.2:8201/exchange/as2_company
AS2-From: as2_company
AS2-To: "AS2 Test"
AS2-Version: 1.1
Host: 10.240.1.2:8101
Connection: close
Content-Length: 3428

  [omitted binary encrypted data]
```

A.4. Asynchronous MDN for Message A.3, Above

```
POST / HTTP/1.1
Host: 10.240.1.2:8201
Connection: close, TE
TE: trailers, deflate, gzip, compress
User-Agent: RPT-HTTPClient/0.3-3I (Windows 2000)
Date: Thu, 19 Dec 2002 15:03:38 GMT
Message-ID: <AS2-20021219_030338@as2_company.dgi_th>
AS2-Version: 1.1
Mime-Version: 1.0
Recipient-Address:
http://10.240.1.2:8201/exchange/as2_company
AS2-To: as2_company
AS2-From: "AS2 Test"
Subject: Your Requested MDN Response
From: as2debug@example.com
Accept-Encoding: deflate, gzip, x-gzip, compress, x-compress
Content-Type: multipart/signed; micalg=sha1;
  protocol="application/pkcs7-signature";
  boundary="----=_Part_337_6452266.1040310218750"
Content-Length: 3103

------=_Part_337_6452266.1040310218750
Content-Type: multipart/report;
  report-type=disposition-notification;
  boundary="----=_Part_336_6069110.1040310218718"

------=_Part_336_6069110.1040310218718
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

The message <x12.edi> sent to Recipient <AS2 Test> on Thu, 19 Dec
2002 15:04:18 GMT with Subject <async MDN request> has been received.
The EDI Interchange was successfully decrypted, and its integrity was
verified.  In addition, the sender of the message, Sender
<as2_company> at Location http://10.240.1.2:8201/exchange/as2_company
was authenticated as the originator of the message.  There is no
guarantee, however, that the EDI interchange was syntactically
correct, or that it was received by the EDI application/translator.
```

```
   ------=_Part_336_6069110.1040310218718
   Content-Type: message/disposition-notification
   Content-Transfer-Encoding: 7bit

   Reporting-UA: AS2@test:8101
   Original-Recipient: rfc822; "AS2 Test"
   Final-Recipient: rfc822; "AS2 Test"
   Original-Message-ID: <#as2_company#01#a4260as2_companyout#>
   Disposition: automatic-action/MDN-sent-automatically;
     processed
   Received-Content-MIC: Hes6my+vIxIYxmvsA+MNpEOTPAc=, sha1

   ------=_Part_336_6069110.1040310218718--

   ------=_Part_337_6452266.1040310218750
   Content-Type: application/pkcs7-signature; name=smime.p7s
   Content-Transfer-Encoding: base64
   Content-Disposition: attachment; filename=smime.p7s

   BhbWjEfbyXoTAS/H0zpnEqLqbaBh29y2v82b8bdeGw8pipBQWmf53hIcqHGM
   4ZBF3CHw5Wrf1JIE+8TwOzdbal30zeChw88WfRfD7c/j1fIA8sxsujvf2d9j
   UxCUga8BVdVB9kH0GeexytyT0KvWQXfaEEcgZGUAAAAAAAA=

   ------=_Part_337_6452266.1040310218750-
```

Authors' Addresses


```
   Dale Moberg
   Cyclone Commerce
   8388 E. Hartford Drive, Suite 100
   Scottsdale, AZ  85255 USA

   EMail: dmoberg@cyclonecommerce.com


   Rik Drummond
   Drummond Group Inc.
   4700 Bryant Irvin Court, Suite 303
   Fort Worth, TX  76107 USA

   EMail: rvd2@drummondgroup.com
```

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.