Network Working Group
Request for Comments: 4823
Category: Informational

T. Harding
R. Scott
Axway

April 2007

FTP Transport for Secure Peer-to-Peer Business Data Interchange over the Internet

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This Applicability Statement (AS) describes how to exchange structured business data securely using the File Transfer Protocol (FTP) for XML, Binary, Electronic Data Interchange (EDI - ANSI X12 or UN/EDIFACT), or other data used for business-to-business data interchange for which MIME packaging can be accomplished using standard MIME content types. Authentication and data confidentiality are obtained by using Cryptographic Message Syntax (S/MIME) security body parts. Authenticated acknowledgements employ multipart/signed replies to the original message.

Table of Contents

1. Introduction

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 [5]. This document expands on RFC 1767 to specify a comprehensive set of data security features, specifically, data privacy, data integrity, authenticity, non-repudiation of origin, and non-repudiation of receipt over FTP. This document also recognizes contemporary RFCs and is attempting to "re-invent" as little as possible. While this document focuses on EDI data, any other data type describable in a MIME format is also supported.

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

- RFC 959: File Transfer Protocol
- RFC 2228: FTP Security Extensions
- RFC 1767: EDI Content Type
- RFC 3023: XML Media Types
- RFC 1847: Security Multiparts for MIME
- RFC 3462: Multipart/Report
- RFCs 2045 to 2049: MIME
- RFC 3798: Message Disposition Notification
- RFCs 3850, 3851, and 3852: S/MIME v3.1 Specifications
- RFC 3274: Compressed Data Content for Cryptographic Message Syntax
- RFC 4217: Securing FTP with TLS
- "Compressed Data for EDIINT" by T. Harding

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 [19].

2. Overview

2.1. Overall Operations

An FTP upload operation is used to send appropriately packaged EDI, XML, or other business data. The receiving application will poll the FTP server for inbound messages, unpackage and handle the message data, and generate a reply for the originator that contains a message disposition acknowledgement within a multipart/report that is signed or unsigned. This request/reply transactional interchange provides secure, reliable, and authenticated transport for EDI or other

business data using FTP. The security protocols and structures used also support auditable records of these transmissions.

2.2. Purpose of a Security Guideline for MIME EDI

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

2.3. Definitions

2.3.1. Terms

AS3	Applicability Statement 3. This is the third applicability statement produced by the IETF EDIINT working group.
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.
Signed Receipt	A receipt containing a digital signature.
Message Disposition Notification (MDN)	The Internet messaging format used to convey a receipt. This term is used interchangeably with receipt. An MDN is a receipt.
Non-repudiation of receipt (NRR)	NRR is a "legal event" that occurs when the original sender of an EDI/EC interchange has verified the signed receipt coming back from the receiver. NRR IS NOT a functional or a technical message.
S/MIME	A format and protocol for adding Cryptographic signature and/or encryption services to Internet MIME messages.

NOTE: While the S/MIME specification describes more than one format for a signed message, all signed messages or receipts used with AS3 MUST utilize the multipart/signed format.

SHA-1	A secure, one-way hash algorithm used in conjunction with digital signature. SHA-1 is the recommend algorithm for AS3.
MD5	A secure, one-way hash algorithm used in conjunction with digital signature. This algorithm is acceptable but not recommended due to its short key length and known weaknesses.
MIC	The message integrity check (MIC) is a representation of the message digest, which results from the application of the selected hash algorithm to the content to be signed. Of particular interest is the digital signature, which includes an encrypted copy of the digest. Additionally, an MDN containing a Received-content-MIC header will also contain (as that header's value) a base-64-encoded representation of the digest.
User Agent (UA)	The application that handles and processes the AS3 request.
STL	Secure Transmission Loop, described in the next section.

2.3.2. The Secure Transmission Loop

This document's focus is on the formats and protocols for exchanging EDI/EC content to which security services have been applied using the File Transmission Protocol (FTP) as the transport.

The "Secure Transmission Loop" (STL) comprises the following two steps:

a) The originator sends a signed and encrypted document with a request for a signed receipt.

b) The recipient decrypts the document, verifies the signature, and returns a signed receipt to the sender.

In other words, the following events occur during the execution of the STL:

- The organization sending EDI/EC data signs and encrypts the data using S/MIME. In addition, the message will request a signed receipt to be returned to the sender of the message.

- 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, as requested to the sending organization in the form of a message disposition notification. This signed receipt will contain the hash of the signature from the received message, indicating to the sender that the received message was verified and/or decrypted properly.

The above describes functionality that, if implemented, will satisfy all security requirements and provide non-repudiation of receipt for the exchange. While trading partners will usually want to utilize the STL, this specification does not require it.

2.3.3. 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 term receipt is used if the acknowledgment is for an interchange resulting in a receipt that is NOT signed. The term signed receipt is used if the acknowledgment is for an interchange resulting in a receipt that IS signed. A term often used in combination with receipts is non-repudiation of receipt. NRR refers to a legal event that occurs only when the original sender of an interchange has verified the signed receipt coming back from the recipient of the message. Note that NRR is not possible without signatures.

For additional information on formatting and processing receipts in AS3, refer to Section 7.

2.4. Operational Assumptions and Options

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 the application of security services.

  Specifically, for EDI ANSI X12, the entire document (including the ISA and IEA segments) is the atom to which security is applied. For EDIFACT, the corresponding definition includes the segments UNA/UNB and UNZ. In other words, EDI/EC interchanges including envelope segments remain intact and unreadable during secure 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, work may need to be done in the future to define ways to extract some elements of the envelope to make them visible; however, that is beyond the scope of this specification.

- 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. Process Options

2.4.2.1. Security Options

- Encrypted or un-encrypted data

  This specification allows for EDI/EC message exchange where the EDI/EC data can be either un-protected 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.

- Use of receipt or not

  This specification allows for EDI/EC message transmission with or without a request for receipt notification. If a signed receipt notification is requested, however, a MIC value is REQUIRED as part of the returned receipt, unless an error condition occurs that results in the inability to compute a valid digest. (Such a case would result, for instance, if an encrypted message could not be decrypted.) Under such circumstances, an unsigned receipt (MDN) SHOULD be returned with the correct "disposition modifier" error value.

- Security formatting

  This specification relies on the guidelines set forth in RFCs 3852 [9] and 3851 [10]. The first of these RFCs describes the Cryptographic Message Syntax (CMS), and the second contains the S/MIME Version 3.1 Message Specification describing a MIME container for CMS objects. Whenever the term S/MIME is used in this document, it refers to Version 3.1 as described therein.

- 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; however, both MD5 and SHA-1 MUST 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, does NOT request a receipt.

  2. Sender sends un-encrypted data, requests an unsigned receipt. The receiver sends back the unsigned receipt.

  3. Sender sends un-encrypted data, requests a signed receipt. The receiver sends back the signed receipt.

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

5. Sender sends encrypted data, requests an unsigned receipt. The receiver sends back the unsigned receipt.

6. Sender sends encrypted data, requests a signed receipt. The receiver sends back the signed receipt.

7. Sender sends signed data, does NOT request a receipt.

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

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

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

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

12. Sender sends encrypted and signed data, requests a signed receipt. Receiver sends back the signed receipt. This case represents the Secure Transmission Loop described above.

2.4.2.2. Compression Options

The AS3 specification supports compression of transmitted data directly through the application of RFC 3274. Implementation details may be found in that RFC and in Harding’s document, "Compressed Data for EDIINT".

3. Referenced RFCs and Their Contribution

3.1. RFC 959: File Transfer Protocol [3]

RFC 959 specifies how data is transferred using the File Transfer Protocol (FTP)

3.2. RFC 2228: FTP Security Extensions [4]

This RFC describes a framework for providing security services to FTP.

3.3. RFC 1847: MIME Security Multiparts [7]

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

3.4. RFC 3462: Multipart/Report [12]

RFC 3462 defines the use of the multipart/report content type, upon which RFC 3798 builds to define the Message Disposition Notification.

3.5. 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).

3.6. RFCs 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.

3.7. RFC 3798: Message Disposition Notification [6]

This Internet RFC defines how a Message Disposition Notification (MDN)is requested, as well as the format and syntax of the MDN. The MDN is the vehicle used by this specification to provide both signed and unsigned receipts.

3.8. RFC 3852: CMS [9] and RFC 3851: S/MIME Version 3.1 Message Specification [10]

This specification describes how MIME shall carry Cryptographic Message Syntax (CMS) Objects.

3.9. RFC 3850: S/MIME Version 3.1 Certificate Handling [11]

RFC 3850 describes certificate handling in the context of CMS and S/MIME.

3.10. RFC 3274: Compressed Data Content Type for Cryptographic Message Syntax (CMS) [17]

This specification provides a mechanism to wrap compressed data within a CMS object.

3.11. RFC 3023: XML Media Types [16]

This RFC defines the use of content type "application" for XML. Note that while conforming implementations SHOULD support the expanded syntax that RFC 3023 introduces for the "+xml" suffix, no support for external parsed entity types is anticipated (as it adds significant complexity to signature processing).

4. Structure of an AS3 Message

4.1. Introduction

The basic structure of AS3 messages comprises MIME encapsulated data with both customary MIME headers and a few additional AS3-specific outer headers. The structures below are described hierarchically in terms of which RFCs have been applied to form the specific structure. The reader is referred directly to the referenced RFCs for implementation details.

Any additional restrictions imposed by this AS are specifically discussed in the sections that follow.

4.2. Structure of an Internet EDI MIME Message

No encryption, no signature

```
-RFC822/2045
  -RFC1767/RFC2376 (application/EDIxxxx or /xml)
```

No encryption, signature

```
-RFC822/2045
  -RFC1847 (multipart/signed)
    -RFC1767/RFC2376 (application/EDIxxxx or /xml)
    -RFC3851 (application/pkcs7-signature)
```

Encryption, no signature

```
-RFC822/2045
  -RFC3851 (application/pkcs7-mime)
    -RFC1767/RFC2376  (application/EDIxxxx or /xml)(encrypted)
```

Encryption, signature

```
-RFC822/2045
  -RFC3851 (application/pkcs7-mime)
    -RFC1847 (multipart/signed)(encrypted)
      -RFC1767/RFC2376  (application/EDIxxxx or /xml)(encrypted)
      -RFC3851 (application/pkcs7-signature)(encrypted)
```

MDN, no signature

```
-RFC822/2045
  -RFC3798 (message/disposition-notification)
```

MDN, signature

```
-RFC822/2045
  -RFC1847 (multipart/signed)
    -RFC3798 (message/disposition-notification)
    -RFC3851 (application/pkcs7-signature)
```

While 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. AS3-Specific Headers

5.1. AS3-From and AS3-To Headers

The AS3-From and AS3-To headers have been provided to assist the sender and the recipient of an EC document to identify each other:

```
AS3-From: < AS3-name >
AS3-To:   < AS3-name >
```

These headers contain textual values, described by the ABNF [22] below, identifying the sender/receiver of a data exchange. A value may be company specific (e.g., a Data Universal Numbering System (DUNS) number), or it may be simply some string mutually acceptable to both trading partners used to identify each to the other.

```
AS3-text = "!" /           ; printable ASCII characters
           %d35-91 /       ; except double-quote (%d34)
           %d93-126        ; or backslash (%d92)

AS3-qtext = AS3-text / SP  ; allow space only in quoted text

AS3-quoted-pair = "\" DQUOTE /  ; \" or
                  "\" "\"       ; \\

AS3-quoted-name = DQUOTE 1*128( AS3-qtext /
                  AS3-quoted-pair) DQUOTE

AS3-atomic-name = 1*128AS3-text

AS3-name = AS3-atomic-name / AS3-quoted-name
```

Note: SP and DQUOTE are defined in [ABNF]RFC 4234.

The AS3-From header value and the AS3-To header value MUST each be an AS3-name comprising 1 to 128 printable ASCII characters. The header MUST NOT be folded, and the value for each of these headers is case-sensitive.

The AS3-quoted-name SHOULD be used only if the AS3-name does not conform to AS3-atomic-name.

The AS3-To and AS3-From header fields MUST be present in all AS3 messages and AS3 MDNs.

Implementations that map entities such as EDI identifiers/qualifiers to AS3 identifiers may choose to constrain the set of AS3-To/AS3-From text values to a subset of the full set defined above, but they may not extend that set.

If the AS3-From or the AS3-To or the association of the two header values is determined to be invalid or unknown to the receiving system, the receiving system MAY respond with an unsigned MDN containing an explanation of the error if the sending system requested an MDN, but it is not required to return an MDN under those circumstances.

5.2. AS3-Version Header

The AS3-Version header is a header that is required only if the value of the header is not "1.0". Its purpose is to allow systems to determine which version of this specification (should the specification evolve over time) the sender of a document has used to

package the document. A user agent MUST NOT reject a message if the version header is missing.

AS3-Version: 1*DIGIT . 1*DIGIT

A version header value of "1.1" indicates an implementation can support EDIINT data compression [20]. A user agent MUST NOT send compressed messages to trading partners who do not use a version header of "1.1" or greater.

6. FTP Considerations

6.1. FTP Security Requirements

FTP has long been viewed as an insecure protocol primarily because of its use of cleartext authentication [3]. This is addressed by RFC 2228 [4], and the use of one of the security mechanisms described therein is strongly encouraged. Specifically, conforming implementations of AS3 SHALL employ FTP client/servers that support the AUTH command described within [4]. While any authentication mechanism based upon [4] MAY be utilized, AUTH TLS (as described in [18]) MUST be supported. (Note that [18] relies on TLS Version 1.0 [13], not Version 1.1 [23].)

6.2. Large File Transfers

Large files are handled correctly by the TCP layer. However, the mechanism for compressing data, referenced in Section 2.4.2.2, efficiently reduces transmission requirements for many data types (including both XML and traditional EDI data). Additionally, some FTP implementations support compression as well.

6.3. MIME Considerations for FTP

6.3.1. Required/Optional Headers

An AS3 message MUST contain the following outer headers:

AS3-To
AS3-From
Date
Message-ID
Content-Type

An AS3 message OPTIONALLY MAY contain the following outer headers:

```
Subject
AS3-Version (assumed to be 1.0 if not present)
Content-Length
```

An AS3 message requesting a receipt MUST contain a Disposition-Notification-To header and MAY contain a Disposition-Notification-Options header (if the receipt is to be signed).

Additional headers MAY be present but are ignored.

6.3.2. Content-Transfer-Encoding

FTP defines several data structures and character encodings via the STRU[cture] and TYPE commands. AS3 requires the file-structure (default) and the image type. The Content-Transfer-Encoding header SHOULD NOT be used; if the header is present, it SHOULD have a value of binary or 8-bit. The absence of this header or the use of alternate values such as "base64" or "quoted-printable" MUST NOT result in transaction failure. Content transfer encoding of MIME parts within the AS3 message are similarly constrained.

6.3.3. Epilogue Must Be Empty

A MIME message containing an epilogue [1] SHALL NOT be used.

6.3.4. Message-Id and Original-Message-Id

Message-Id and Original-Message-Id are formatted as defined in Section 3.6.4 of RFC 2822 [15]: "<" id-left "@" id-right ">". Message-Id length is a maximum of 998 characters. Message-Id SHOULD be globally unique; 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.

NOTE: When creating the Original-Message-Id header in an MDN, always use the exact syntax contained in the original message: do not strip or add "angle brackets".

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.

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 or not, 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/pkcs7-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 multipart MIME content type. When the EDI Interchange is part of a multipart MIME content type, the MIC MUST be calculated across the entire multipart content, including the MIME headers.

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

1) As an acknowledgment that the EDI Interchange was sent, and then 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.

2) As an acknowledgment 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.

3) As an acknowledgment that the receiving trading partner has authenticated the sender of the EDI Interchange.

4) 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. Message Disposition Notifications (MDN)

The AS3-MDNs are returned on a separate FTP TCP/IP connection and are a response to an AS3 message.

The following diagram illustrates the delivery of an AS3-MDN delivery:

```
 AS3-MDN
[S] ----( connect )----> [R]   [FTP Server]
[S] ----( send )-------> [R]   [AS3-Message]
[S] ----( disconnect )-> [R]   [FTP Server]

[S] <---( connect )----- [R]   [FTP Server]
[S] <---( send )-------- [R]   [AS3-MDN]]
[S] <---( disconnect )-- [R]   [FTP Server]

Note: Refer to Section 7.4.4 for additional
      programming notes.
```

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" ":" ftpurl

This syntax is a residual of the use of MDN's in an SMTP transfer. Since this specification is adjusting the functionality from SMTP to

FTP and retaining as much as possible from the [5] functionality, the ftpurl must be present.

The ftpurl field is specified as an RFC 1738 <URL:"ftp://" login ["/" fpath [";type=" ftptype]]>, and while it MUST be present, it may be ignored if the ftpurl points to an unknown location. If the ftpurl points to an unknown location, it is RECOMMENDED that the mdn is returned back to a known ftpurl for the sender of the received message.

For requesting MDN-based receipts, the originator supplies the required extension headers that precede the message body.

The header "tags" are as follows:

A Disposition-notification-to header is added to indicate that a message disposition notification is requested. This header is specified in [6].

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 the MDN.

Other headers, especially "Date", SHOULD be supplied; the values of these headers are often mentioned in the human-readable section of an MDN to aid in identifying the original message.

Disposition-notification-options identifies characteristics of the message.

The following Disposition notification is in accordance with [6].

```
    EXAMPLE:
      Disposition-notification-to:       // Requests the MDN
        ftp://host:port/inbox            // Location to return MDN
      Disposition-notification-options:  // The signing options for
                                            MDN
        signed-receipt-protocol=optional, pkcs7-signature;
        signed-receipt-micalg=optional, sha1, md5
```

Disposition-notification-options syntax:

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

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

```
parameter = attribute "=" importance ", " value *("," value)

importance = "required" / "optional"

attribute = "signed-receipt-protocol" / "signed-receipt-micalg"
```

So the Disposition-notification-options string could be:

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

The currently supported 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 Used	Value
MD5	md5
SHA-1	sha1

Receiving agents SHOULD be able to recover gracefully from a <micalg> parameter value that they do not recognize.

The semantics of the "signed-receipt-protocol" parameter is 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 and calculating the micalg in the Received-content-MIC header.

   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.

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 an MDN. A UA that does not

understand the "signed-receipt-protocol" parameter, or the "signed-receipt-micalg" parameter, 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 as well as 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 must be determined by the trading partners. Typically, if a signed receipt is required by 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" for processing a receipt-request follow:

1) When a receipt is requested, explicitly specifying that the receipt be signed, then the receipt MUST be returned with a signature unless conditions (2) or (3) below are applicable.

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

3) When a receipt is requested, explicitly specifying that the receipt be signed, but the recipient is unable to compute the digest (e.g., message was encrypted, and recipient unable to decrypt), then the recipient SHOULD NOT return "Received-content-MIC" in the MDN to the requestor. If the MDN sets the disposition (e.g., "processed/error: decryption-failed") appropriately, then the "Received-content-MIC" may be returned, but the value must be discarded.

4) 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.

5) If a message is received without a request for a receipt, then a receipt (signed or unsigned) MAY be returned.

The "Received-content-MIC" MUST be calculated as follows:

- For any signed messages, the MIC to be returned is calculated on the RFC 1767 MIME header and content. Canonicalization as specified in RFC 1848 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 MIME header and content. The content after decryption MUST be canonicalized before the MIC is calculated.

- For unsigned, un-encrypted messages, the MIC MUST be calculated over the message contents prior to Content-Transfer-Encoding and without the MIME or any other RFC 822 [14] headers, since these are sometimes altered or reordered by message transfer agents (MTAs).

7.4. MDN Format and Value

This section defines the format of the AS3 Message Disposition Notification (AS3-MDN).

7.4.1. AS3-MDN General Formats

The AS3-MDN follows the MDN specification [6] except where noted in this section. The modified entity definitions in this document use the vertical-bar character, '|', to denote a logical "OR" construction. Refer to RFC 2045 for the format of MIME-message-headers.

The format of the AS3-MDN is

MDN, no signature

```
-RFC822/2045
  -RFC3798 (message/disposition-notification)
```

MDN, signature

```
-RFC822/2045
  -RFC1847 (multipart/signed)
    -RFC3798 (message/disposition-notification)
    -RFC3851 (application/pkcs7-signature)
```

7.4.2. AS3-MDN Construction

The AS3-MDN-body is formatted as a MIME multipart/report with a report-type of "disposition-notification".

When unsigned, the transfer-layer ("outermost") entity-headers of the AS3-MDN contain the Content-Type header that specifies a content type of "multipart/report", parameters indicating the report-type, and the value of the outermost multipart boundary.

When the AS3-MDN is signed, the transfer-layer ("outermost") entity-headers of the AS3-MDN contain a Content-Type header that specifies a content type of "multipart/signed", 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 imbedded 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

```
AS3-disposition-notification-content =
          [ reporting-ua-field CRLF ]
          [ mdn-gateway-field CRLF ]
          [ original-recipient-field CRLF ]
          final-recipient-field CRLF
          [ original-message-id-field CRLF ]
          AS3-disposition-field CRLF
          *( failure-field CRLF )
          *( error-field CRLF )
          *( warning-field CRLF )
          *( extension-field CRLF )
          [ AS3-received-content-MIC-field CRLF ]
```

It is noted that several of the optional fields defined by RFC 3798 and shown above are not relevant to a point-to-point transport such as FTP and would not normally appear in an AS3 MDN.

7.4.3. AS3-MDN Fields

The rules for constructing the AS3-disposition-notification-content are identical to the rules for constructing the disposition-notification-content as defined in Section 7 of RFC 3798 [6] except that the RFC 3798 disposition-field has been replaced with the AS3-disposition-field and that the AS3-received-content-MIC field has been added. The differences between the RFC 3798 disposition-field and the AS3-disposition-field are described below. Where there are differences between this document and RFC 3798, those entity names have been changed by prepending "AS3-". Entities below that do not differ from RFC 3798 are not necessarily further defined in this document.

Refer to RFC 3798 [6] and RFC 4234 [22] for entities that are not further defined in this document.

```
AS3-disposition-field = "Disposition:" disposition-mode ";"
                AS3-disposition-type [ "/" AS3-disposition-modifier]

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

action-mode = "manual-action" / "automatic-action"

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

AS3-disposition-type = "processed" / "failed"

AS3-disposition-modifier = ( "error" / "warning" ) /
                           AS3-disposition-modifier-extension

AS3-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: " AS3-MDN-warning-description /
           "failure: " AS3-MDN-failure-description

AS3-MDN-warning-description = *( TEXT )

AS3-MDN-failure-description = *( TEXT )

AS3-received-content-MIC-field =
            "Received-content-MIC:" encoded-message-digest
            "," digest-alg-id CRLF
```

```
encoded-message-digest =
           1*( ALPHA / DIGIT / "/" / "+" ) *3"="
           ;( i.e. base64( message-digest ) )

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

The "Received-content-MIC" extension field is set after 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.

The algorithm used to calculate the message digest MUST be the same as the "micalg" value used by the sender in the multipart/signed message. When no signature is received, the message-digest MUST be calculated using the algorithm specified by the "micalg" value in the Disposition-Notification-Options header. When no signature is received and no micalg parameter is provided, then the SHA-1 algorithm MUST be used to calculate the digest. 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 in order for the sender to verify non-repudiation of receipt.

AS3-MDN field names (e.g., "Disposition:", "Final-Recipient:") are case-insensitive (cf. RFC 3798, Section 3.1.1).

AS3-MDN action-modes, sending-modes, AS3-disposition-types, and AS3-disposition-modifier values that are defined above, and user-supplied *(TEXT) values are also case-insensitive. AS3 implementations MUST NOT make assumptions regarding the values supplied for AS3-MDN-warning-description or AS3-MDN-failure-description or for the values of any (optional) error, warning, or failure fields.

7.4.4. Additional AS3-MDN Programming Notes

1. Unlike SMTP, for FTP transactions, Original-Recipient and Final Recipient SHOULD NOT be different. The value in Original-Message-ID MUST match the original Message-ID header value.

2. Refer to RFC 3462 and RFC 3798 for the formatting of the Content-Type entity-headers for the MDN.

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

4. 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.

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

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

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

8. The "failed" disposition type MAY 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.

9. An AS3 implementation MUST present to its trading partners an FTP-compliant server interface where inbound documents and MDNs are received.

10. An AS3 implementation MUST be able to retrieve inbound messages from its currently configured FTP server interface.

Note: Programming Notes 9 and 10 do not imply any specific method for supplying the FTP server interface. But, they do allow for several different types of implementations. Some vendors may choose to imbed an FTP-compliant server interface within their product, and others may choose to utilize off-the-shelf FTP servers to supply the required FTP server interface. Some may choose to utilize hosting services provided by their trading partner or by a third-party hosting service. Whichever method is utilized, an AS3 implementation MUST support rules 9 and 10.

11. AS3 implementations MAY imbed an FTP server interface within their product.

12. AS3 implementations MUST be configurable to allow the use of an external FTP hosting service.

Note: An external FTP hosting service may be hosted by a third-party or possibly hosted by your trading partner.

13. An AS3 implementation MUST be able to send business documents and MDNs to a trading partner's currently configured FTP server interface.

14. An AS3 implementation may imbed FTP client code into their product or use a third-party FTP client.

15. Example Configurations

    1. Peer to Peer
       Trading Partner A (TPA) is using a local FTP server, and Trading Partner B (TPB) is using an imbedded FTP server.

```
[A Client] ----( connect )----> [B Server]
[A Client] ----( send )-------> [B Server] [AS3-Message]
[A Client] ----( disconnect )-> [B Server]

[A Server] <---( connect )----- [B Client]
[A Server] <---( send )-------- [B Client] [AS3-MDN]]
[A Server] <---( disconnect )-- [B Server]
[A Client] <---( GET )--------- [A Server]
```

    2. Third-Party Hosting
       Both parties are using the same third-party-hosted FTP server.

```
[A Client] ----( connect )----> [Hosted Server]
[A Client] ----( send )-------> [Hosted Server] [AS3-Message]
[A Client] ----( disconnect )-> [Hosted Server]
[Hosted Server]( GET )--------> [B Client]

[Hosted Server] <---( connect )----- [B Client]
[Hosted Server] <---( send )-------- [B Client] [AS3-MDN]]
[Hosted Server] <---( disconnect )-- [B Client]
[A Client]      <---( GET )--------- [Hosted Server]
```

    3. Trading Partner Hosting
       TPA is using the imbedded FTP server hosted by TPB.

```
[A Client] ----( connect )----> [B Server]
[A Client] ----( send )-------> [B Server] [AS3-Message]
[A Client] ----( disconnect )-> [B Server]

[B Server] <---( connect )----- [B Client]
[B Server] <---( send )-------- [B Client] [AS3-MDN]]
[B Server] <---( disconnect )-- [B Client]
[A Client] <---( GET )--------- [B Server]
```

7.5. Disposition Mode, Type, and Modifier

7.5.1. Disposition Mode Overview

This section will provide a brief overview of how processed, error, failure, or warning notifications are used.

7.5.2. Successful Processing Status Indication

When a receipt or signed receipt is requested, 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 a user’s, an administrator’s, or software control.

Since 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 this specification does not restrict the use of the "disposition-mode" to just 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 in the case where the message content is being rejected or ignored should be specified in the MDN "disposition-field" as below. (An example of this case is when the EDI UA determines that a signed receipt cannot be returned because it does not support the requested protocol format, so the EDI UA chooses not to process the message contents itself.)

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

The "failed" AS3-disposition-type should 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:" AS3-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 WILL 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" should be set to the "processed" "disposition-type" value and the "error" "disposition-modifier" value.

The "error" AS3-disposition-modifier with the "processed" disposition-type should 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:" AS3-disposition-modifier-extension should be used to combine the indication of an error with a pre-defined description of a specific, well-known error. Further information about the error may be contained in an error-field.

For use in Internet EDI, the following "error" "disposition-modifier" values are defined:

"Error: decryption-failed"
The receiver could not decrypt the message contents.

"Error: authentication-failed"
The receiver could not authenticate the sender.

"Error: integrity-check-failed"
The receiver could not verify content integrity.

"Error: insufficient-message-security"
The security level of the message did not match the agreed level between TPs.

"Error: decompression-failed"
The receiver could not decompress the message contents.

"Error: unexpected-processing-error"
A catch-all for any additional processing errors.

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

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

7.5.5. Processing Warnings

Situations arise in EDI where 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 described above, the "disposition-field" SHOULD be set to the "processed" "disposition-type" value, and the "warning" "disposition-modifier" value.

The "warning" AS3-disposition-modifier should 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:" AS3-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" values are defined:

"Warning: authentication-failed, processing continued"

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

EXAMPLE

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

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 EDI trading partner ID and FTP 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, that they also exchange public key certificates using the recommendations specified in the S/MIME Version 3.1 Message Specification.

The message formats and S/MIME conformance requirements for certificate exchange are specified in this document. In the long term, additional Internet-EDI standards may be developed to simplify the process of establishing a trading partnership, including the third-party authentication of trading partners, as well as 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 privacy and authentication issues.

Extracted from S/MIME Version 2 Message Specification [21]:

> 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 tripleDES 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 the relative cryptographic strength of messages.

Extracted from S/MIME Version 3.1 Certificate Handling [11]:

> 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 has 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 places where signature and certificate checking might fail include:
>
> - no Internet mail addresses in a certificate matches the sender of a message, if the certificate contains at least one mail address
> - 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.

The following certificate types MUST be supported.

With URL
Without URL
Self Certified
Certification Authority Certified

The complete certification chain MUST be included in all certificates. All certificate verifications MUST "chain to root". Additionally, the certificate hash should match the hash recomputed by the receiver.

10. References

10.1. 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] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[4] Horowitz, M. and S. Lunt, "FTP Security Extensions", RFC 2228, October 1997.

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

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

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

[8] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

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

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

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

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

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

[14] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", STD 11, RFC 822, August 1982.

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

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

[17] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, June 2002.

[18] Ford-Hutchinson, P., "Securing FTP with TLS", RFC 4217, October 2005.

[19] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

[20] Harding, T., "Compressed Data for EDIINT", Work in Progress, January 2007.

[21] Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L., and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, March 1998.

[22] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[23] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

Appendix A. Message Examples

NOTE: All examples are provided as an illustration only, and are not considered part of the protocol specification. If an example conflicts with the protocol definitions specified above or with that of a referenced RFC, the example is wrong.

A.1. Signed Message Requesting a Signed Receipt

```
Date: Wed, 31 Jul 2002 13:34:50 GMT
AS3-Version: 1.0
AS3-From:  cyclone
AS3-To: "trading partner"
Message-Id: <200207310834482A70BF63@host.com>
Disposition-Notification-To: ftp://host:port/mdnbox
Disposition-Notification-Options: signed-receipt-
  protocol=optional,pkcs7-signature;
  signed-receipt-micalg=optional,sha1
Content-Type: multipart/signed; boundary="as3BouNdary1as3";
   protocol="application/pkcs7-signature"; micalg=sha1
Content-Length: 3075

--as3BouNdary1as3
Content-Type: application/edi-x12
Content-Disposition: Attachment; filename=rfc1767.dat

[ISA ...EDI transaction data...IEA...]

--as3BouNdary1as3
Content-Type: application/pkcs7-signature

  [omitted binary pkcs7 signature data]
--as3BouNdary1as3--
```

A.2. MDN for Message A.1 Above

```
Date: Wed, 31 Jul 2002 13:34:50 GMT
AS3-From: "trading partner"
AS3-To: cyclone
AS3-Version: 1.0
Message-ID: <709700825.1028122454671.JavaMail@ediXchange>
Content-Type: multipart/signed; micalg=sha1;
  protocol="application/pkcs7-signature";
  boundary="----=_Part_57_648441049.1028122454671"
Content-Length: 1024

------=_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@host.com>
&  From: cyclone
&  To: "trading partner"
&  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: AS3 Server
&   Original-Recipient: rfc822; "trading partner"
&   Final-Recipient: rfc822; "trading partner"
&   Original-Message-ID: <200207310834482A70BF63@host.com>
&   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 RFC 3851 [10], "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification".

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 [12], 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.

Authors' Addresses

Terry Harding
Axway
8388 E. Hartford Drive, Suite 100
Scottsdale, AZ 85255 USA

EMail: tharding@us.axway.com

Richard Scott
Axway
8388 E. Hartford Drive, Suite 100
Scottsdale, AZ 85255 USA

EMail: rscott@us.axway.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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