Network Working Group
Internet-Draft
Intended status: Informational
Expires: 19 September 2026

D. Petta
Drummond Group, LLC

18 March 2026

AS2 Specification Modernization
draft-petta-rfc4130bis-05

Abstract

This document provides an applicability statement (RFC 2026, Section 3.2) describing how to securely exchange structured business data over HTTP. 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 (see Section 10.1). 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). This document obsoletes RFC 4130 and stands on its own without reference to AS1 or SMTP, except where required for IANA registry updates.

This document also updates IANA registries originally created by RFC 3335 and RFC 4130.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://DrummondGroup.github.io/draft-petta-rfc4130bis/draft-petta-rfc4130bis.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-petta-rfc4130bis/.

Source for this draft and an issue tracker can be found at https://github.com/DrummondGroup/draft-petta-rfc4130bis.

Status of This Memo

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

This Internet-Draft will expire on 19 September 2026.

Copyright Notice

Table of Contents

1. Introduction

This document is a revision ("bis") of RFC 4130, which defined the Applicability Statement 2 (AS2) protocol for secure and reliable transport of business data over HTTP. It obsoletes RFC 4130. The purpose of this revision is to modernize the specification, clarify ambiguities, and incorporate implementation experience gathered since the publication of RFC 4130. Subsequent versions of this draft will refine these updates based on discussion and consensus in the IETF community. This revision also adheres to the principle of backward compatibility. Implementations conformant with RFC 4130 remain valid under this specification, and no breaking changes are introduced. In addition, this document updates existing IANA registrations from RFC 3335 and RFC 4130. The specific IANA actions are described in Section 11.

Note to readers: Some contributors have suggested that this work could eventually be split into two documents: a minimal RFC4130bis for errata and clarifications, and a separate AS2 v2 specification with a clean modern baseline. This document currently attempts to balance both objectives within a single text, but further discussion may refine the scope.

1.1. Applicable RFCs

Previous work on Internet EDI focused on specifying MIME content types for EDI data. [RFC1767] expands on this 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 recognizes contemporary RFCs and avoids re-inventing mechanisms wherever 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 (baseline: HTTP/1.1)
- 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 8098 Message Disposition Notification (updates RFC 3798)
- RFC 5751 S/MIME v3.2 Specification (obsoletes RFC 3851)
- RFC 8551 S/MIME v4.0 (obsoletes RFC 5751)
- RFC 5652 Cryptographic Message Syntax (CMS) (obsoletes RFC 3852)

This specification references S/MIME Version 4.0 [RFC8551] as the baseline for algorithm requirements and security message formats. S/MIME 4.0 introduces AuthEnvelopedData, which provides authenticated encryption for algorithms such as AES-GCM and AES-CCM. For backward compatibility with implementations that have not yet migrated to S/MIME 4.0, this specification also permits the use of EnvelopedData from S/MIME 3.2 [RFC5751] when using algorithms such as AES-CBC that require separate integrity protection. The choice between AuthEnvelopedData and EnvelopedData is determined by the content encryption algorithm selected (see Section 7.2 for details).

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. Implementers should note that HTTP/2 and HTTP/3 MAY be used as transports, but are not required for interoperability.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Backward Compatibility and Interoperability

A central design principle of this specification is "backward compatibility" with RFC 4130 and with the underlying RFCs it references. This specification does not redefine or override backward-compatibility rules established in those RFCs. Implementations MUST rely on the mechanisms provided in underlying standards.

Consistent with the Robustness Principle ("be conservative in what you send and liberal in what you receive"), this document clarifies requirements and aligns terminology but does not introduce breaking changes. Implementations that conformed to RFC 4130 remain conformant to this specification. Any deviations are limited to clarifications intended to improve interoperability.

This specification establishes S/MIME Version 4.0 [RFC8551] as the baseline for conformant implementations. Implementations MUST support S/MIME 4.0 message formats (including AuthEnvelopedData) and the algorithm requirements specified in Section 7. Implementations SHOULD also support S/MIME Version 3.2 [RFC5751] for backward compatibility with legacy trading partners that have not yet migrated to S/MIME 4.0. When both partners support S/MIME 4.0, implementations SHOULD use AuthEnvelopedData with authenticated encryption algorithms (AES-GCM, AES-CCM) for improved security. When interoperating with S/MIME 3.2 systems, implementations SHOULD use EnvelopedData with algorithms such as AES-CBC that require separate integrity protection via digital signatures.

This specification defines requirements for modern AS2 deployments using contemporary cryptographic algorithms. It does not redefine or extend the use of weak algorithms such as 3DES or SHA-1. When both partners support this version of AS2, only modern algorithms are in scope.

When interoperability with RFC 4130 systems is required, implementers SHOULD apply the clarifications provided in Section 1.2.1 (Legacy Interoperability). Section 1.2.1 is non-normative and does not alter the algorithm requirements defined in Section 7, but records expected behavior when communicating with legacy systems.

1.2.1. Legacy Interoperability (Non-Normative)

This section provides the conditions for interoperability with legacy AS2 implementations conforming to [RFC4130]. These provisions apply only when a modern AS2 implementation communicates with a partner that has not migrated to this specification. In such cases, both parties are effectively operating under [RFC4130], not this document.

These notes are provided to reduce ambiguity and ensure consistent behavior across implementations. They do not alter the algorithm requirements specified in Section 7, nor do they extend the use of deprecated algorithms.

Examples of legacy considerations include:

- **Message Integrity Checks (MICs):** Implementations may continue to accept SHA-1 for MIC calculations when required by legacy partners. SHA-1 SHOULD NOT be generated by conformant modern implementations, but SHA-1 values SHOULD continue to be used to maintain backward compatibility.

- **Encryption Algorithms:** Implementations may accept inbound messages encrypted with 3DES from legacy partners. However, 3DES SHOULD NOT be generated by conformant implementations. AES (128-bit or stronger) remains the normative requirement in Section 7.2.

  Note: NIST withdrew the 3DES specification on 1 January 2024, and disallowed the two-key variant in 2017. Any residual use of 3DES is for backward compatibility only and SHOULD NOT be generated by conformant implementations.

- **Multiple-Recipient Encryption:** RFC 4130 did not clearly specify expected behavior for multiple-recipient support. Modern implementations SHOULD support recoverable encryption by including a copy of the content-encryption key (CEK) for each recipient, and SHOULD include one for the originator when feasible. Legacy implementations may omit this; modern systems should tolerate it.

- **Error Handling:** When encountering unsupported algorithms or malformed cryptographic structures in legacy exchanges, implementations SHOULD generate a clear error condition (e.g., an unsigned MDN reporting "unsupported-mic-algorithm"). Silent fallback to weaker algorithms is NOT RECOMMENDED.

- **Profile Selection:** Implementations may provide administrators the ability to select profiles (e.g., "AS2-1.2 legacy mode" versus "AS2-1.3 modern mode") for specific trading partner agreements, ensuring predictable behavior without runtime handshakes.

These clarifications are provided for reference and consistency across vendors. They are non-normative and are not intended to redefine [RFC4130] or to weaken the algorithm requirements of this specification. Refer to Section 1.2 for discussion of backward compatibility principles, and Section 7 for normative algorithm requirements.

1.3. Rationale

The updates in this specification reflect community consensus to:

- Preserve backward compatibility with RFC 4130 and the underlying RFCs it references.
- Provide explicit guidance on which protocol versions form the interoperability baseline for certification and testing.
- Incorporate de facto updates already widely deployed (e.g., RFC 8098 for MDNs, migration from SHA-1 to SHA-2 wherever possible).
- Document stronger security requirements while allowing backward-compatible fallback to enable phased adoption.
- Avoid unnecessary disruption by permitting, but not requiring, newer transport features such as HTTP/2, and by clarifying rather than redefining MDN behavior.

This approach reduces ambiguity, simplifies certification, and ensures interoperability across implementations.

1.4. Terms

AS2: Applicability Statement 2 (this document) and [RFC4130]; see RFC 2026 [RFC2026], Section 3.2

EDI: Electronic Data Interchange

EC: Electronic Commerce (often referred to as Business to Business, B2B).

B2B: Business to Business

Receipt: The functional message that is sent from a receiver to a sender to acknowledge that an EDI/EC interchange has been received. 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 over the same TCP/IP connection as the sender’s original message.

Asynchronous Receipt: A receipt returned to the sender over a different TCP/IP connection than the sender’s original message.

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): 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 algorithm is retained for backward compatibility but is deprecated in this specification. Implementations MUST support SHA-256 or stronger where possible, and SHOULD prefer these algorithms in production environments unless legacy partners cannot yet migrate to SHA-256 or stronger.

MD5: A secure, one-way hash algorithm used in conjunction with digital signature. This algorithm is obsolete and MUST NOT be generated by conformant implementations.

MIC: The Message Integrity Check (MIC) is a cryptographic method used to verify that a message has not been altered or tampered with during transmission or storage, ensuring the data is trustworthy and complete. It works by generating a unique hash value from the message’s contents, which is then transmitted with the message. The recipient recalculates the hash on the received message and compares it to the provided MIC; if they don’t match, the message is discarded, indicating it was modified.

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

2. Overview

2.1. Overall Operation

An HTTP POST operation [RFC2616] is used to send appropriately packaged EDI, XML, or other business data. The Request-URI ([RFC2616], Section 10.5) 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. HTTPS is RECOMMENDED as the default transport for modern implementations (see Section 10.1).

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

The message formats and processing requirements described below maintain strict backward compatibility (see Section 1.2).

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 not limited to strict EDI use; it applies to any electronic commerce application for which business data needs to be exchanged securely over the Internet.

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, encrypted and compressed 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, encrypts and compresses 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 decompresses and 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 the 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 refer to Section 8.

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 (i.e., the payload) 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 was previously useful to make some envelope information visible. Since the EDI/EC message exchanges are routed over the public Internet and not over VANs, this specification 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.

- Compressed or Uncompressed Data

This specification allows for optional compression and MAY be applied alone or in combination with signing and/or encryption, as defined in [RFC3274]. It is supported by AS2-Version: 1.1 and higher.

- 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 required MIC value is absent.

- Use of Synchronous or Asynchronous Receipts

In addition to a receipt request, this specification allows for the designation of the type of receipt that should be returned. It supports synchronous or asynchronous receipts in the MDN format specified in {{structure-and-processing-of-an-mdn-message}} of this document.

- Security Formatting

This specification relies on the guidelines set forth in RFC 5751/5652 [RFC5751] / [RFC5652] "S/MIME Version 3.2 Message Specification; Cryptographic Message Syntax" as well as RFC 8551 [RFC8551] "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0" for modern implementations.

- Hash Function, Message Digest Choices

When a signature is used, implementations MUST support SHA-256 and SHOULD support SHA-384 or stronger. SHA-1 MAY be supported for incoming messages for backward compatibility, but SHOULD NOT be generated for outgoing messages unless it is strictly required for interoperability. MD5 is obsolete and MUST NOT be generated by conformant implementations.

- Encryption Algorithms

For content encryption, implementations MUST support AES-128-CBC and AES-256-CBC. Implementations are RECOMMENDED to support authenticated encryption modes such as AES-GCM and AES-CCM, which use AuthEnvelopedData (S/MIME 4.0). When using AES-GCM or AES-CCM, implementations MUST use AuthEnvelopedData. When using AES-CBC or other non-authenticated modes, implementations MUST use EnvelopedData with separate integrity protection via digital signatures. Triple-DES (3DES) and RC2 are deprecated and SHOULD NOT be generated by conformant implementations, though they SHOULD be accepted for backward compatibility with legacy systems. A single content encryption algorithm MUST be used for all recipients of a given message; it is not

permitted to encrypt the same message with AES-CBC for some recipients and AES-GCM for others.

- Key Management Algorithms

For key transport, implementations MUST support RSA with a minimum key length of 2048 bits. Implementations MAY support key agreement algorithms such as Diffie-Hellman or Elliptic Curve Diffie-Hellman (ECDH) as specified in [RFC5753]. When using elliptic curves, implementations SHOULD support NIST P-256 (secp256r1) or stronger curves.

- Permutation Summary

The optional use of compression, as defined in [RFC3274] was introduced in AS2-Version 1.1.
Compression can be applied to the message payload before signing and/or encryption, reducing transmission size and improving efficiency. Most modern AS2 implementations support compression, and it can be used by itself or in combination with signing and encryption.

The following summary therefore extends the original AS2 security permutations to include the additional possibilities created by the use of compression, resulting in a total of twenty-four security permutations.

Without Compression (12 permutations)

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.

With Compression (12 permutations)

13. Sender sends compressed data (not encrypted or signed) and does NOT request a receipt.

14. Sender sends compressed data and requests an unsigned receipt. Receiver sends back the unsigned receipt.

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

16. Sender sends compressed and encrypted data and does NOT request a receipt.

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

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

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

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

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

22. Sender sends compressed, encrypted, and signed data and does NOT request a signed or unsigned receipt.

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

24. Sender sends compressed, encrypted, and signed data and requests a signed receipt. Receiver sends back the signed receipt.

Key Notes

- Compression MAY be applied alone or in combination with signing and/or encryption, as defined in [RFC3274] and is supported by AS2-Version 1.1 and higher.

- Users may choose any of these twenty-four possibilities, but only the final example (24), when compression, signing, encryption, and a signed receipt are all used, offers the full suite of security and efficiency features described in Section 2.3.1, “The Secure Transmission Loop.”

- As with the original permutations, the receipts may be either synchronous or asynchronous, and the choice 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

[RFC2616] specifies how data is transferred using HTTP.

3.2. RFC 1847 MIME Security Multiparts

[RFC1847] defines security multipart for MIME: multipart/encrypted and multipart/signed.

3.3. RFC 3462 Multipart/Report

[RFC3462] defines the use of the multipart/report content type, something that the MDN RFC 3798 builds upon.

3.4. RFC 1767 EDI Content

[RFC1767] 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

[RFC2045], [RFC2046], and [RFC2049] 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

[RFC3798] 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 5751 and 5652 S/MIME Version 3.2 Message Specifications and Cryptographic Message Syntax (CMS)

[RFC5751] and [RFC5652] describe how S/MIME carries CMS Objects.

3.8. RFC 3023 XML Media Types

[RFC3023] defines the use of content type "application" for XML (application/xml).

3.9. RFC 3274 Compressed Data Content Type for Cryptographic Message Syntax (CMS)

[RFC3274] defines a mechanism for compressing data within the Cryptographic Message Syntax (CMS), which is the foundation for Secure/Multipurpose Internet Mail Extensions (S/MIME). It specifies a CompressedData content type that allows data to be compressed prior to being signed or encrypted. This reduces the size of transmitted messages and improves efficiency without altering the security services provided by signing or encryption. AS2-Version 1.1 incorporated the compression capability described in RFC 3274, enabling trading partners to optionally apply compression to message payloads before signing and/or encrypting. Most modern AS2 implementations support this feature to reduce bandwidth usage and improve transmission performance, particularly for large payloads.

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 on how to code in compliance with all RFCs involved, refer to the specific RFCs. Any difference between AS2 implementations and RFCs are mentioned specifically in the sections below.

4.2. Structure of an Internet EDI MIME Message

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

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

```
        - RFC5751 (application/pkcs7-signature)

   Encryption, no signature, no compression
      - RFC2616/2045
        - RFC5751 (application/pkcs7-mime)
          - RFC1767/RFC3023  (application/EDIxxxx or /xml) (encrypted)

   Encryption, signature, no compression
      - RFC2616/2045
        - RFC5751 (application/pkcs7-mime)
          - RFC1847 (multipart/signed)(encrypted)
            - RFC1767/RFC3023  (application/EDIxxxx or /xml) (encrypted)
            - RFC5751 (application/pkcs7-signature)(encrypted)

   No encryption, no signature (with optional compression)
      - RFC2616/2045
        - RFC3274 (application/pkcs7-mime; CompressedData) [optional]
          - RFC1767/RFC3023 (application/EDIxxxx or /xml)

   No encryption, signature (compression may occur before or after signing)
      - RFC2616/2045
        - [optional RFC3274 (CompressedData) if compress-before-sign]
          - RFC1847 (multipart/signed)
            - [optional RFC3274 (CompressedData) if compress-after-sign]
              - RFC1767/RFC3023 (application/EDIxxxx or /xml)
              - RFC5751 (application/pkcs7-signature)

   Encryption, no signature (with optional compression)
       - RFC2616/2045
         - RFC5751 (application/pkcs7-mime)
           - [optional RFC3274 (CompressedData)]
             - RFC1767/RFC3023 (application/EDIxxxx or /xml) (encrypted)

   Encryption, signature (compression may occur before or after signing)
      - RFC2616/2045
        - RFC5751 (application/pkcs7-mime)
          - [optional RFC3274 (CompressedData) if compress-before-sign]
            - RFC1847 (multipart/signed) (encrypted)
              - [optional RFC3274 (CompressedData) if compress-after-sign]
              - RFC1767/RFC3023 (application/EDIxxxx or /xml) (encrypted)
              - RFC5751 (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)
- RFC5751 (application/pkcs7-signature)

Key Notes

- RFC 3274 (CompressedData) is the normative reference for compression.
- Compression MAY be combined with signing and/or encryption in either order, but the choice affects what the digital signature covers.
- Many implementations compress before signing and encrypting to maximize size reduction, but compression after signing and before encrypting MUST also be supported.
- 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

This specification permits but does not require the use of HTTP/2 or HTTP/3 as transport protocols. Interoperability testing and certification are scoped to HTTP/1.1 unless otherwise agreed upon between trading partners.

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 ([RFC2616], Section 10.4.2 and elsewhere).

The request line is followed by entity headers specifying content length ([RFC2616], Section 14.14) and content type ([RFC2616], Section 14.18). The Host request header ([RFC2616], Sections 9 and 14.23) is also included.

When using Transport Layer Security (TLS), the request-URI MUST indicate the appropriate scheme value, HTTPS. Implementations MUST support TLS 1.2 or higher and SHOULD follow the guidance in Section 10.1. Encrypted message bodies MAY be used in addition to TLS when required by business policy.

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, ([RFC2616] Sections 8.1.2, 8.2, and 19.7.1). A number of other differences exist because HTTP does not conform to MIME [RFC2616] 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 [RFC2616]. This difference is discussed in [RFC2616], 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 [RFC2616], Section 3.7.2, it is explicitly noted that multiparts MUST have null epilogues.

For HTTP transport, large files SHOULD be handled correctly by the TCP layer. In addition, [RFC2616], Sections 3.5 and 3.6 describe options for compressing or chunking entities to be transferred, and Section 8.1.2.2 describes a pipelining option that is useful for segmenting large amounts of data. These clarifications are consistent with existing AS2 practice and maintain full backward compatibility (see Section 1.2).

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 [RFC2616], 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

The Message-Id and Original-Message-Id headers identify a message uniquely and are formatted as defined in [RFC5322], Section 3.6.4:

"<" id-left "@" id-right ">"

The length of a Message-Id value MUST NOT exceed 998 characters. For maximum interoperability, the length SHOULD be 255 characters or less.

The Message-Id value MUST be globally unique, and the id-right portion SHOULD be something unique to the sending host environment (for example, a fully qualified domain name).

Implementations that generate Message-Id values MUST NOT include spaces or control characters. Implementations SHOULD remove spaces rather than substitute another character when constructing identifiers from other message attributes such as AS2-From or AS2-To.

Receivers are not required to accept malformed identifiers. If a message is received with a Message-Id that contains spaces or control characters, the implementation SHOULD treat it as syntactically invalid and SHOULD return an MDN with a disposition of processed/ error and a human-readable explanation such as "invalid-message-id" (see [RFC8098]). If an implementation chooses to proceed despite the malformed identifier, it MUST NOT propagate or generate a new message using that malformed value.

When sending a message, the Message-Id field value MUST be enclosed in angle brackets (“<” and “>”). The brackets are not part of the actual identifier value. For backward compatibility, receiving implementations SHOULD NOT reject a message that omits angle brackets.

When creating the Original-Message-Id header in an MDN, always use the exact syntax as received on the original message; do not strip or add angle brackets.

See [RFC5322], Section 3.6.4.

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 [RFC2616], Sections 14.23 and 19.5.1.

5.4. HTTP Response Status Codes

Implementations MUST use standard HTTP response codes to signal the outcome of the message transfer. The meaning of the HTTP status code is limited to the success or failure of the transport operation itself, not the semantic processing of the AS2 message content. 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 [RFC2616], Section 6.1.1 and throughout Section 10.

Receiving implementations MAY send an interim 102 (Processing) response [RFC4918] under HTTP/1.1 to indicate that the inbound message has been fully received and that processing is underway. The 102 response can help prevent sender-side network timeouts for large synchronous transfers by signaling progress while decryption, signature verification, or storage continues.

Use of 102 (Processing) is OPTIONAL. It has been deprecated in later HTTP specifications and *MUST NOT* be used with HTTP/2 or HTTP/3, where interim responses have different semantics. Implementations that do not receive a 102 response MUST NOT assume that a failure has occurred solely because no interim status was returned. They SHOULD continue waiting for the final status response for at least the duration of their configured HTTP read timeout or any timeout agreed upon between trading partners.

To minimize the risk of network timeouts during lengthy message processing, receivers SHOULD return an appropriate transfer-layer response as quickly as possible after receiving the full message content. For asynchronous message exchanges, the preferred response is 204 No Content, which indicates that the message has been received successfully and that an asynchronous MDN will follow once processing has completed. This convention is maintained for interoperability with existing AS2 products and certification profiles.

Some implementations MAY instead use 202 Accepted to indicate successful receipt and deferred processing; however, 204 No Content remains the recommended and most widely deployed response for asynchronous workflows.

Implementations MAY close the connection immediately after sending this response if persistent connections are not required by configuration.

After processing completes, the receiver MUST return a final HTTP status code indicating the success or failure of the message transfer. The sender MUST use this final response to determine whether retry is appropriate.

Retry *MUST NOT* be attempted when:

- the final HTTP response indicates successful receipt (e.g., ‘200 OK‘ or ‘204 No Content‘ for asynchronous transfers, or ‘202 Accepted‘ for implementations that use deferred processing semantics) **and** a valid MDN has been received confirming the message disposition; or
- a permanent-failure status code is returned (4xx other than 408), or

Retry *MAY* be attempted when:

- the HTTP connection fails before the final status is received,
- a transient error such as 408 (Request Timeout) or 5xx (Server Error) occurs, or
- no response is received within the configured timeout.

Implementations SHOULD refer to Section 5.5 for additional guidance on retry logic, back-off behavior, and use of partial-transfer recovery. The 102 (Processing) status code, if used, MUST NOT be treated as a trigger for retry.

5.5. HTTP Error Recovery and Reliability

When an AS2 message transfer fails due to a transient transport-layer condition (for example, an HTTP 408 Request Timeout, 425 Too Early, 500 Internal Server Error, 503 Service Unavailable or network interruption before the final response), the sending system SHOULD attempt an automatic retry.

Each retry attempt MUST reuse the same Message-ID value so that the receiving system can identify duplicate transmissions and prevent double-processing. A receiving system detecting a duplicate Message-ID MUST NOT treat the message as new and SHOULD return the previously generated MDN, if available.

Implementations SHOULD permit configuration of retry behavior rather than enforcing fixed intervals or limits. The following guidelines are RECOMMENDED but not required:

- **Retry intervals** SHOULD increase exponentially (e.g., 5 min, 10 min, 20 min, 40 min, …) to reduce congestion.
- **Retry duration** SHOULD be configurable based on business requirements; some environments may continue for several days, while others may terminate after one or two attempts.
- **Maximum attempts** SHOULD be limited to prevent indefinite retries when persistent errors occur.

Implementations SHOULD NOT retry when:

- A final 2xx response and/or valid MDN has been received;
- The HTTP response indicates a permanent failure (e.g., 400, 401, 403, 404);
- The partner has explicitly rejected the message by sending a signed MDN with a "failed" disposition.

The HTTP 102 (Processing) interim status MAY be used under HTTP/1.1 to indicate progress on long-running synchronous operations. It MUST NOT be used as a signal to initiate or suppress retries. Implementations MUST ignore 102 responses when determining whether a retry is required. The 102 response MUST NOT be used with HTTP/2 or HTTP/3.

Implementations MAY also support *AS2 Restart*, which allows a partially uploaded message to resume from the point of interruption rather than retransmitting the entire payload. This optional feature is defined in [I-D.draft-harding-as2-restart-02]. Implementations supporting Restart MUST ensure message integrity through signature or checksum validation of all resumed segments.

Additional guidance for retry management, error classification, and duplicate detection is described in [I-D.draft-duker-as2-reliability-16]. While both of these drafts are expired, they remain widely referenced in AS2 interoperability testing and provide a useful operational baseline for error-recovery behavior.

The objective of error recovery is reliability, not speed. Systems SHOULD favor successful delivery over strict timing, provided that duplicate protection and security requirements are preserved.

6. Additional AS2-Specific HTTP Headers

The following headers are to be included in all AS2 messages and all AS2 MDNs. [RFC3335].

6.1. AS2 Version Header

To promote backward compatibility, AS2 includes a version header. The major version digit indicates wire-level compatibility; minor version digits designate feature sets, clarifications, or extensions that remain compatible within the same major version. Thus, all values in the "1.x" range are compatible with AS2-Version 1.0, while a potential future "2.0" version would indicate a non-backward-compatible revision.

Receiving systems MUST NOT fail due to the absence of the AS2-Version header. Its absence MUST be assumed to be equivalent to the default AS2-Version value of 1.0.

AS2-Version: 1.0 - All implementations of this specification MUST support and advertise "AS2-Version: 1.0". Versions in the range "1.0" through "1.9" MAY be used. All implementations MUST interpret any value in that range as conforming to this specification, with no differences in baseline behavior. In other words, only the major version digit ("1") defines compatibility for implementations that do not support additional, non-AS2-specified functionality.

Implementations MAY use "1.1" through "1.9" to signal extensions of this specification. Any such extensions MUST be fully transparent to implementations that recognize only "AS2-Version: 1.0".

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

AS2-Version: 1.2 - Indicates those implementations that include an EDIINT-Features header as defined in RFC 6017. The values in an EDIINT-Features header specify the features supported by the AS2 implementation. Examples may include CEM, AS2-Reliability and multiple-attachments, however others may also be included. A receiving implementation MUST NOT fail if it does not support or understand any of the supported values contained within an EDIINT-Features header.

AS2-Version: 1.3 - Indicates those implementations that support the modernization defined by this specification, including updated algorithm requirements (e.g.,

SHA-256 for MIC/signatures; AES as the encryption baseline per RFC 8551), alignment with MDN handling as specified in RFC 8098, and support for multiple-recipient encryption as described in Section 7.2 of this specification.

When both partners are configured for AS2 version 1.3, weak algorithms (e.g., SHA-1, 3DES, RC2, MD5) MUST NOT be generated by conformant implementations. When interoperating with a legacy partner that operates at AS2 version 1.2 or lower, implementations SHOULD apply the legacy interoperability clarifications described in {{backward-compatibility-and-interoperability}}.1 (non-normative).

Future minor versions (1.x) may designate additional extensions or clarifications that remain backward-compatible with AS2 version 1.0. A major version update (2.0 or higher) would indicate a non-backward-compatible revision and may come later.

6.2. AS2 Product header

The AS2-Product header value identifies the AS2 product and version used by the sender. This information enables interoperability testing, certification, and troubleshooting by allowing trading partners to detect known product-specific behaviors or version-related quirks.

The AS2-Product header value is OPTIONAL for AS2-Version 1.x systems but MUST be included in messages generated by implementations declaring *AS2-Version: 1.3* (or later).

The header field value MUST follow the format:

* <product-name>: lowercase alphanumeric and hyphen characters (az, 09, "-") without spaces.
* <major.minor[.patch]>: version string consistent with the product's release version, with one or more numeric components separated by dots.

Examples:

AS2-Product: biztalk:2025.1
AS2-Product: example.com-connect:4.2.3

Implementations *MUST NOT* use arbitrary identifiers or vendor aliases that do not reflect the actual product in use. The value is static and determined at build time. If a product supports multiple AS2 variants, the version portion MAY include an implementation-specific suffix (e.g., "1.2-drummond").

Implementations MAY use the AS2-Product value for automated interoperability tuning or to apply compatibility workarounds for known product versions. However, this field is not intended for feature-negotiation purposes; supported feature tokens belong in the EDIINT-Features header, as defined in RFC 6017.

6.3. 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 [RFC2234].

The AS2-quoted-name SHOULD be used only if the AS2-name does not conform to AS2-atomic-name. This explicitly includes situations where embedded spaces are part of the AS2-name.

The AS2-To and AS2-From header fields MUST be present in all AS2 messages and AS2 MDNs whether they are synchronous or asynchronous in nature.

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, such as an MDN error disposition value of "unknown-trading-relationship", if the sending system requested an MDN.

7. Algorithm Requirements

This section defines the normative requirements for cryptographic algorithms used in AS2. These requirements apply to all conformant implementations. Guidance on interoperability with legacy AS2 systems that continue to use older algorithms is provided separately in Section 1.2.1.

7.1. Hash Algorithms

Implementations MUST support SHA-256 for message integrity check (MIC) calculations and digital signatures. Implementations SHOULD support SHA-384 or stronger algorithms. SHA-1 and MD5 are deprecated due to known weaknesses and SHOULD NOT be generated by conformant implementations.

See Section 1.2.1 for clarifications on handling legacy algorithms when interoperating with RFC 4130 systems.

7.2. Encryption Algorithms

Implementations MUST support AES encryption algorithms as defined in S/MIME Version 4.0 [RFC8551]. At a minimum, AES-128-CBC and AES-256-CBC MUST be supported. Implementations are also RECOMMENDED to support AES-128-GCM and AES-256-GCM. Support for AES-CCM is also RECOMMENDED for environments requiring authenticated encryption.

Triple DES (3DES) and RC2 are deprecated and SHOULD NOT be generated by conformant implementations.

7.2.1. EnvelopedData vs AuthEnvelopedData

The choice between EnvelopedData and AuthEnvelopedData depends on the content encryption algorithm selected:

- **AuthEnvelopedData** MUST be used when employing authenticated encryption algorithms such as AES-GCM or AES-CCM. These algorithms provide both confidentiality and integrity protection in a single cryptographic operation. AuthEnvelopedData was introduced in S/MIME 4.0 [RFC8551] specifically to support these modes.

- **EnvelopedData** MUST be used when employing non-authenticated encryption algorithms such as AES-CBC or when maintaining backward compatibility with S/MIME 3.2 implementations [RFC5751]. When using EnvelopedData, integrity protection MUST be provided separately through digital signatures (multipart/signed).

Implementations MUST NOT mix content encryption algorithms for different recipients of the same message. A single content encryption algorithm MUST be selected and used for all recipients. For example, if a message is encrypted with AES-128-GCM, all recipient information MUST use AES-128-GCM; it is not permitted to encrypt the content- encryption key with AES-CBC for some recipients and AES-GCM for others.

7.2.2. Multiple-Recipient Encryption

To support recoverable decryption and regulatory requirements, implementations SHOULD support multiple-recipient encryption of the content-encryption key (CEK), consistent with [RFC8551] Section 3.3. A copy of the CEK encrypted for the originator SHOULD also be included in the EnvelopedData, and the same principle applies to AuthEnvelopedData when using AES-CCM or AES-GCM.

See Section 1.2.1 for guidance on handling 3DES and other weak algorithms when interoperating with legacy AS2 systems.

8. Structure and Processing of an MDN Message

This document aligns MDN behavior with RFC 8098, clarifying semantics for interoperability. It does not redefine the MDN format. Implementations MUST be able to parse historic MDN forms as described in RFC 3798 for backward compatibility.

8.1. 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 requirements in this section update but do not alter the compatibility of MDN formats with existing AS2 implementations (see Section 1.2). This ensures interoperability with both RFC 3798 and RFC 8098 implementations.

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/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 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 contained within the multi-part MIME content.

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.

8.2. 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 synchronous response MUST indicate transfer-layer success or failure, such as 200 OK or 202 Accepted. The format of this response MAY be identical to that used when no AS2-MDN is requested.

The asynchronous AS2-MDN is sent on a separate HTTP or HTTPS 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 receiving system *SHOULD* return a transfer-layer response (typically 202 Accepted or 204 No Content) as soon as the last byte of the inbound message has been received, without waiting for decryption, signature verification, or message persistence. This minimizes the risk of network timeouts and

ensures that the sender can begin awaiting the asynchronous MDN promptly. The asynchronous MDN MUST be transmitted as an independent HTTP message, separate from the original connection used to submit the AS2 message.

Implementations *MAY* use persistent (keep-alive) HTTP connections. Closing the TCP connection immediately after sending the response is *RECOMMENDED* for simplicity, but not required. Some application servers and frameworks manage connection lifecycles automatically and may keep the socket open. The AS2 specification does not mandate that the AS2 layer explicitly close the connection.

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 (e.g., "200 OK" or "204 No Con
tent")
   {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 also utilize a transfer protocol different from that used to send the original AS2 message.

The advantage of the synchronous MDN is that it provides the sender of the AS2 message with a verifiable confirmation of delivery within a single synchronous logic flow. However, if the message is large, the time required to process it and return the AS2-MDN on the same connection may exceed the maximum configured time permitted for maintaining an open connection.

The advantage of the asynchronous MDN is that it provides for the rapid return of a transfer-layer acknowledgment from the receiver, confirming receipt of data, while allowing full processing to occur later. This reduces connection duration and timeout risk. However, the asynchronous AS2-MDN MUST include sufficient identifying information (for example, Original-Message-ID and Final-Recipient) so that the message originator can correlate the MDN with its original message and update the processing status accordingly.

Synchronous and asynchronous HTTP or HTTPS MDNs are both valid under this specification. Implementations MUST support receiving both types and SHOULD support sending both.

8.3. 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
```

The "Disposition-notification-to" header field is retained for compatibility with the MDN specification [RFC3798], but its value is not used by AS2 implementations to determine where to return the MDN. Its presence just indicates that an MDN receipt is to be returned to the originator. In AS2, the field value may be an email address, a URL, a fully qualified domain name, an AS2 identifier, or any other implementation-specific string. Implementations MUST NOT reject a message based on the syntax of this field. This document relaxes the original requirement from RFC 4130, which mandated an email address, in order to reflect current AS2 practice while maintaining backward compatibility (see Section 1.2).

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 portion of a MDN to aid in identifying the original message.

MDNs will be returned in the HTTP response when requested, unless an asynchronous MDN 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
```

This is an example requesting that the MDN be asynchronous:

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

Receipt-delivery-option syntax allows the 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 (now replaced by [RFC5322]); 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
```

Finally, the header, Disposition-notification-options, identifies characteristics of message disposition notification as in [RFC3798]. 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,sha-256,sha1
```

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-256	sha-256
SHA-384	sha-384
SHA-512	sha-512
SHA-1	sha-1 or sha1 (should *only* be used in legacy deployments that cannot support SHA-256 or greater)

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 identifies one or more message integrity check (MIC) algorithms, in order of preference, that the requester supports for signing the returned receipt. Although multiple values MAY be listed to indicate fallback options, only a single MIC algorithm is used in the returned MDN because the "Received-content-MIC" field conveys exactly one digest value.

Recipients MUST select the first algorithm in the list that they also support and MUST compute the Received-content-MIC using that algorithm. Senders SHOULD list the strongest algorithm first. Modern implementations SHOULD include only a single value unless multiple values are needed to support phased migration away from weaker algorithms. Implementations MUST accept messages that contain multiple values and MUST ignore unsupported values.

When a sender lists multiple algorithms, recipients MUST NOT fall back to an algorithm that is not explicitly listed by the sender. Trading partners typically pre-configure acceptable MIC algorithms through bilateral agreement, and runtime negotiation is not needed. If none of the algorithms listed is supported, the recipient SHOULD reject the message and MAY return an unsigned MDN indicating "unsupported-mic-algorithm" rather than silently selecting a weaker algorithm. When the header is absent (e.g., unsigned messages), an implementation MUST use a locally configured default algorithm; SHA-256 SHOULD be preferred, but SHA-1 MAY be used when required for legacy partners.

The following algorithm requirements apply to all implementations:

- Implementations **MUST** support SHA-256.
- Implementations **SHOULD** support SHA-384 or stronger.
- SHA-1 **MAY** be accepted only when required for interoperability with legacy partners and SHOULD NOT be generated for modern implementations once both parties support stronger algorithms.
- MD5 is obsolete and MUST NOT be generated.

See Section 10 for additional algorithm requirements and deprecation timelines.

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".

1. 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 8.5 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 be considered invalid.

8.3.1. 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 outer MIME or any other RFC 5322 headers, since these may sometimes be altered or reordered by intermediary user agents or proxies.

8.4. MDN Format and Values

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

8.4.1. AS2-MDN General Formats

```
AS2-MDN = AS2-sync-MDN | AS2-async-http-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-MDN-body =
   AS2-signed-MDN-body | AS2-unsigned-MDN-body
```

8.4.2. 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 containing 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
```

8.4.3. 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 [RFC3798], 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: duplicate-filename" |
    "error: illegal-filename" |
    "error: insufficient-message-security" |
    "error: integrity-check-failed" |
    "error: invalid-message-id" |
    "error: unexpected-processing-error" |
    "error: unknown-trading-relationship" |
    "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 = "sha-256" | "sha-384" | "sha-512" | "sha-1" | "sha1"
```

```
                    ; The "sha1" is a legacy representation of the "sha-1"
                    ; defined hash in the IANA Textual Names Registry.
                    ; It should be maintained for backwards compatibility;
                    ; sha1 | sha-1 should only be used by legacy deployments
                    ; that cannot support sha-256 or greater
```

To improve error reporting and interoperability, this specification introduces additional standardized disposition modifiers beyond those defined in [RFC4130] and [RFC8098].

These modifiers are used to indicate specific failure conditions that cannot be adequately represented by the existing error codes and may not be compatible with earlier implementations of AS2. Implementations MUST include a human-readable explanation in the MDN Explanation field when returning these modifiers.

Future modifiers may be registered through the IANA registry for AS2 Disposition Values and Modifiers (see Section 11).

The "Received-content-MIC" extension field is set when the integrity of the received message is verified. The MIC value is the base64-encoded message-digest computed over the received message using 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 8.3.1 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-256 algorithm SHOULD be used. SHA-1 MAY be used only for legacy interoperability (see Section 1.2.1). This field is set only when the content of the message is 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.

8.4.4. AS2-MDN Field Requirements

The following fields have clarified requirements for interoperability:

- **Final-Recipient** — This field **MUST** always be present in an MDN and MUST identify the AS2-To value of the original message.
- **Original-Message-ID** — This field is **REQUIRED** and MUST exactly match the ‘Message-ID‘ of the original message as transmitted. ‘Message-ID‘ in the MDN itself is optional.
- **Disposition-Notification-To** — Implementations **MAY** include this field using an email address, URL, hostname, or other identifier as appropriate to the system. However, as specified in [RFC4130], receiving applications **MUST** ignore this field and **MUST NOT** reject a message due to syntax or address format violations. The field is retained for compatibility with prior implementations.

8.4.5. Additional AS2-MDN Programming Notes

- 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.

8.5. Disposition Mode, Type, and Modifier

8.5.1. Disposition Mode Overview

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

8.5.2. 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 to disallow sending of 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 the 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.

8.5.3. 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 content. 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"

8.5.4. 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:

- "Error: authentication-failed" - the receiver could not authenticate the sender.
- "Error: decompression-failed" - the receiver could not decompress the message contents.
- "Error: decryption-failed" - the receiver could not decrypt the message contents.
- "Error: duplicate-filename" - the message payload contained a filename already received by the backend server.
- "Error: illegal-filename" - the message payload contained a filename that could nor be processed by the backend server.
- "Error: insufficient-message-security" - the content of the message was not appropriately enveloped according to the agreed-upon message security.
- "Error: integrity-check-failed" - the receiver could not verify content integrity.
- "Error: invalid-message-id" - the receiver could not parse the value of the Message-ID header because it was not syntactically correct.
- "Error: unexpected-processing-error" - a catch-all for any additional processing errors.
- "Error: unknown-trading-relationship" - the receiver could not correlate the AS2-To/AS2-From header values to values known to the system.

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
```

8.5.5. 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

8.5.6. 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 modulu
s.

     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 modulu
s.

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

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

8.6. 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 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, follow the same semantics as those defined in [RFC3798]. 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 ([RFC3798], 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 portion) 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 portion). 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 (this was historically used to return the original message or its headers within the SMTP context).

9. Public Key Certificate Handling

The initial exchange and certification of public keys are essential steps in establishing a secure trading partnership. This process MAY occur manually during partner onboarding or automatically through supported mechanisms such as the *AS2 GET* method (see Section 9.2). Implementations MUST maintain a database of public keys used for encryption and signature verification, together with the mapping between the EDI trading partner identifier and its associated RFC 5322 [RFC5322] email address and HTTP URL/URI. The exact procedures for establishing and configuring secure AS2 messaging can vary among trading partners and software implementations.

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 (CA) and the use of a CA 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 the advice provided in [RFC3850].

The message formats useful for certificate exchange are found in [RFC5751] and [RFC5652].

9.1. Certificate Roles and Requirements

While TLS certificates and AS2 message-signing certificates both use the X.509 standard, they serve distinct purposes and MUST be managed separately:

- **TLS Certificates** are used solely to secure the HTTPS transport channel. They establish session-level confidentiality and integrity and SHOULD be issued by a trusted certification authority (CA) in production environments. For public-facing servers, TLS certificates SHOULD comply with the CA/Browser Forum Baseline Requirements (https://cabforum.org/baseline-requirements-documents/). Self-signed TLS certificates MAY be used for testing or by explicit agreement between trading partners, provided they include a **Subject Alternative Name (SAN)** extension containing the DNS name and/or IP address. The SAN extension MUST be marked as non-critical.

- **AS2 Certificates** are used for signing and encrypting AS2 messages and MUST NOT be the same as the TLS certificate. Separation ensures that a compromise of the transport layer does not affect message-level security, and vice versa. Using the same certificate for both purposes creates security dependencies and operational risks that MUST be avoided. AS2 certificates MAY be CA-issued or self-signed, depending on organizational policy and trading-partner agreements.

  AS2 certificates MUST use a key length of at least **2048 bits for RSA keys**. For elliptic-curve certificates, the selected curve MUST provide equivalent or stronger security (e.g., P-256 or higher).

Although short certificate lifetimes are now common in the TLS ecosystem due to CA/Browser Forum requirements and industry regulations, AS2 certificates generally do not require the same frequency of renewal. AS2 systems handle far fewer encrypted transactions than high-volume web servers, and certificate rollover can be operationally complex. Implementations SHOULD allow independent lifetime policies for AS2 and TLS certificates.

9.2. Certificate Exchange and Renewal

Automated certificate management significantly reduces operational risk. Implementations SHOULD support *Certificate Exchange Messaging (CEM)* [I-D.draft-meadors-certificate-exchange-14] to enable secure, automated exchange of AS2 certificates between trading partners. When CEM is not available, manual exchange processes MUST ensure integrity and authentication of keys prior to activation.

The *AS2 GET* method MAY be used for initial retrieval of partner certificates. Implementations using AS2 GET MUST ensure that certificate retrieval is authenticated (to verify the requester’s identity) and authorized (to ensure only legitimate trading partners can access certificates). While certificates themselves are digitally signed by their issuer and thus tamper-evident, authentication and authorization are required to prevent unauthorized parties from obtaining certificates and using them to identify

legitimate trading partners or map relationships. For self-signed certificates, additional out-of-band verification (such as fingerprint confirmation via secure channel) is REQUIRED to establish initial trust before use in production.

9.3. Operational Guidance

- TLS and AS2 certificates MUST be managed separately and MUST NOT be the same certificate.
- CEM SHOULD be supported to reduce manual errors and configuration drift.
- Self-signed certificates SHOULD include SAN extensions for clarity and validation consistency.
- Implementations SHOULD support configurable expiration and notification mechanisms for certificate renewal.
- Administrators MUST NOT reuse TLS certificates as AS2 certificates to maintain separation of security domains.

For security and algorithm lifecycle considerations, see Section 7 and Section 10.

10. Security Considerations

This entire document is concerned with secure transport of business data, covering confidentiality, authentication, and non-repudiation.

Cryptographic algorithms used for signatures, MIC calculations, and encryption are subject to modernization and deprecation guidance. The definitive algorithm requirements (for hash functions and encryption) are specified in Section 7. This section provides security rationale and additional guidance.

Legacy algorithms such as SHA-1, MD5, 3DES, and RC2 are deprecated due to known weaknesses. Implementations SHOULD NOT generate these algorithms in modern implementations. However, legacy use cases may still be encountered when interoperating with older systems conforming to [RFC4130]. See Section 1.2.1 for clarifications on legacy interoperability.

Modern implementations are expected to support strong algorithms such as SHA-256 or stronger for MIC calculations, and AES (128-bit or greater) for encryption. Implementations SHOULD also support advanced AES modes such as AES-GCM and AES-CCM for improved efficiency and authenticated encryption. See [RFC8551] for details.

Implementations must ensure robust handling of all cryptographic failures. Administrators are encouraged to monitor IETF and NIST publications for algorithm lifecycle updates and to update deployed systems accordingly. These compatibility allowances are described in more detail in Section 1.2 and Section 1.2.1.

When processing certificates, failures such as expired, revoked, or untrusted certificates MUST result in immediate and noticeable error reporting. See [RFC3280] and [RFC8551] for guidance on certificate path validation. For guidance on certificate management, key exchange, and renewal, including use of Certificate Exchange Messaging (CEM) and AS2 GET, see Section 9 and Section 9.2.

10.1. HTTPS and TLS Requirements

Consensus Update: Implementations *MUST* support TLS 1.2 or higher and *SHOULD* support TLS 1.3. TLS 1.3 is strongly encouraged as the default where platform support allows, but TLS 1.2 remains the mandatory baseline to ensure interoperability with older environments (e.g., legacy JVMs). Products SHOULD allow administrators to configure which TLS versions are enabled, so that TLS 1.3 can be mandated by policy where required. Future revisions of this specification are expected to mandate TLS 1.3 as the minimum required version once deployment is widespread.

Administrators SHOULD use only cipher suites listed as “Recommended (Y)” in the IANA TLS Parameters (https://www.iana.org/assignments/tls-parameters) registry. Implementations SHOULD provide configurable cipher selection rather than hardcoding cipher lists.

New implementations of AS2 *SHOULD* use HTTPS as the default transport protocol to provide confidentiality and integrity in transit. Plain HTTP remains permitted to support message-level encryption and backward compatibility with existing deployments.

This guidance promotes strong encryption, aligns with current best practices, and ensures that AS2 remains interoperable with existing deployments while allowing administrators to phase out weaker protocols and cipher suites over time.

Future Considerations Per BCP 195/RFC 7525bis, new IETF protocols and major revisions are expected to mandate TLS 1.3. AS2-bis is a backward-compatible revision, and therefore specifies MUST support of TLS 1.2 and SHOULD support of TLS 1.3 to preserve interoperability with existing implementations. At the same time, implementers are strongly encouraged to deploy TLS 1.3 as the default for all new products and configurations. Future versions of AS2 (for example, a potential AS2-Version 1.3 or subsequent update) are expected to raise

the baseline to MUST support TLS 1.3 or higher, and may deprecate TLS 1.2 once widespread deployment makes this feasible. Implementers are encouraged to prepare for a future revision of this specification that will mandate TLS 1.3 as the minimum required version.

10.2. TLS Server Certificates

The following certificate types MUST be supported for TLS server certificates:

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

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.

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.

Because server certificates are exchanged, and also trust is established during the configuration of the trading partner relationship, runtime validation (including hostname matching and certificate path validation) SHOULD be performed unless an out-of-band trust model has been explicitly agreed upon by trading partners. If a self-signed TLS certificate is used, it SHOULD contain a Subject Alternative Name (SAN) extension that includes the DNS name and/or IP address of the sender. If included, this certificate extension MUST be marked as non-critical.

Note: Although not restricted by this specification, self-signed TLS certificates should be used with great care, especially in production environments.

10.3. 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.

If TLS is used for transport, the guidance in Section 10.1 applies.

10.4. 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.

11. IANA Considerations

IANA is requested to update the following registries:

- MDN Disposition Modifier Names
  https://www.iana.org/assignments/mdn/mdn.xhtml#disposition-modifier

- Message Disposition Notification Parameters
  https://www.iana.org/assignments/mdn/mdn.xhtml#parameters

11.1. Registration

RFC 4130 originally defined 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".

This document updates that definition, and IANA is requested to replace RFC 4130 with this document as the reference for the MDN Disposition Modifier Names registry.

11.1.1. Disposition Modifier ’warning’

Parameter-name: warning
Semantics: See (#as2-mdn-fields) and (#processing-warnings) in this document.

12. Acknowledgments

Carl Hage, Karen Rosenfeld, Chuck Fenton, Russ Housley, Marc Blanchet, Erik Wrammer, and many others provided valuable suggestions during both the initial review of RFC 4130 that improved that applicability statement and this bis specification. The authors would also like to thank the past and current vendors who have participated in the Drummond AS2 interoperability testing. Their contributions have ultimately led to great improvement in the clarity of this document.

13. References

13.1. Normative References

[RFC1767] Crocker, D., "MIME Encapsulation of EDI Objects", RFC 1767, DOI 10.17487/RFC1767, March 1995, <https://www.rfc-editor.org/rfc/rfc1767>.

[RFC1847] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, DOI 10.17487/RFC1847, October 1995, <https://www.rfc-editor.org/rfc/rfc1847>.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/rfc/rfc2026>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/rfc/rfc2045>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/rfc/rfc2046>.

[RFC2049] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, DOI 10.17487/RFC2049, November 1996, <https://www.rfc-editor.org/rfc/rfc2049>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <https://www.rfc-editor.org/rfc/rfc2234>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/rfc/rfc2616>.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, DOI 10.17487/RFC3023, January 2001, <https://www.rfc-editor.org/rfc/rfc3023>.

[RFC3274] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, DOI 10.17487/RFC3274, June 2002, <https://www.rfc-editor.org/rfc/rfc3274>.

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, DOI 10.17487/RFC3280, April 2002, <https://www.rfc-editor.org/rfc/rfc3280>.

[RFC3335] Harding, T., Drummond, R., and C. Shih, "MIME-based Secure Peer-to-Peer Business Data Interchange over the Internet", RFC 3335, DOI 10.17487/RFC3335, September 2002, <https://www.rfc-editor.org/rfc/rfc3335>.

[RFC3462] Vaudreuil, G., "The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages", RFC 3462, DOI 10.17487/RFC3462, January 2003, <https://www.rfc-editor.org/rfc/rfc3462>.

[RFC3798] Hansen, T., Ed. and G. Vaudreuil, Ed., "Message Disposition Notification", RFC 3798, DOI 10.17487/RFC3798, May 2004, <https://www.rfc-editor.org/rfc/rfc3798>.

[RFC3850] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling", RFC 3850, DOI 10.17487/RFC3850, July 2004, <https://www.rfc-editor.org/rfc/rfc3850>.

[RFC4130] Moberg, D. and R. Drummond, "MIME-Based Secure Peer-to-Peer Business Data Interchange Using HTTP, Applicability Statement 2 (AS2)", RFC 4130, DOI 10.17487/RFC4130, July 2005, <https://www.rfc-editor.org/rfc/rfc4130>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/rfc/rfc5322>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/rfc/rfc5652>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <https://www.rfc-editor.org/rfc/rfc5751>.

[RFC6017] Meadors, K., Ed., "Electronic Data Interchange - Internet Integration (EDIINT) Features Header Field", RFC 6017, DOI 10.17487/RFC6017, September 2010, <https://www.rfc-editor.org/rfc/rfc6017>.

[RFC8098] Hansen, T., Ed. and A. Melnikov, Ed., "Message Disposition Notification", STD 85, RFC 8098, DOI 10.17487/RFC8098, February 2017, <https://www.rfc-editor.org/rfc/rfc8098>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/rfc/rfc8551>.

13.2. Informative References

[I-D.draft-duker-as2-reliability-16]
Duker, J. and dale.moberg@gmail.com, "Operational Reliability for EDIINT AS2", Work in Progress, Internet-Draft, draft-duker-as2-reliability-16, 21 October 2014, <https://datatracker.ietf.org/doc/html/draft-duker-as2-reliability-16>.

[I-D.draft-harding-as2-restart-02]
Harding, T., "AS2 Restart for Very Large Messages", Work in Progress, Internet-Draft, draft-harding-as2-restart-02, 26 January 2011, <https://datatracker.ietf.org/doc/html/draft-harding-as2-restart-02>.

[I-D.draft-meadors-certificate-exchange-14]
Meadors, K. and D. Moberg, "Certificate Exchange Messaging for EDIINT draft-meadors-certificate-exchange-14.txt Abstract", Work in Progress, Internet-Draft, draft-meadors-certificate-exchange-14, 22 December 2011, <https://datatracker.ietf.org/doc/html/draft-meadors-certificate-exchange-14>.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <https://www.rfc-editor.org/rfc/rfc2246>.

[RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, DOI 10.17487/RFC4918, June 2007, <https://www.rfc-editor.org/rfc/rfc4918>.

[RFC5753] Turner, S. and D. Brown, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 5753, DOI 10.17487/RFC5753, January 2010, <https://www.rfc-editor.org/rfc/rfc5753>.

Appendix A. Message Examples

Note to Readers: All examples are provided for illustration only, and are not part of the protocol specification. If an example conflicts with the protocol definitions, the example is wrong. Email addresses in the examples (e.g., in the Disposition-Notification-To field) reflect one valid option but are not required. In AS2, this field may also contain a URL, a fully qualified host name, an AS2 identifier, or another implementation-specific string, as described in Section 8.3.

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

```
POST /receive HTTP/1.1
Host: 10.234.160.12:80
User-Agent: AS2 Company Server
Date: Wed, 31 Jul 2025 13:34:50 GMT
AS2-Version: 1.3
AS2-From: "as2 Name"
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,sha-256
Content-Type: multipart/signed; boundary="as2BouNdary1as2";
     protocol="application/pkcs7-signature"; micalg=sha-256
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 in A.1, Above

```
HTTP/1.1 200 OK
AS2-From: 0123456780000
AS2-To: "as2 Name"
AS2-Version: 1.3
Message-ID: <709700825.1028122454671.JavaMail@ediXchange>
Content-Type: multipart/signed; micalg=sha-256;
     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: "as2 Name"
&  To: 0123456780000
&  Received on: 2025-07-31 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: 43d9tGY3gNSGuFaut4PAGvuc+48VgW6USgXLDPTxsBU=, sha-256
&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 [RFC3462], 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

```
POST /trading_partner HTTP/1.1
Host: 10.240.1.2:58101
User-Agent: AS2 Company Server
Message-ID: <#as2_company#01#a4260as2_companyout#>
Date: Thu, 19 Dec 2024 15:04:18 GMT
Subject: Signed and encrypted message with 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://10.240.1.2:58201/exchange/as2_company
Disposition-Notification-Options: signed-receipt-protocol=optional,
     pkcs7-signature; signed-receipt-micalg=optional,sha-256
Receipt-Delivery-Option: http://10.240.1.2:58201/exchange/as2_company
AS2-From: as2_company
AS2-To: "AS2 Test"
AS2-Version: 1.3
Connection: close
Content-Length: 3428

  {omitted binary encrypted data}
```

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

```
POST / HTTP/1.1
Host: 10.234.160.12:80
Connection: close, TE
TE: trailers, deflate, gzip, compress
User-Agent: AS2 Company Server
Date: Thu, 19 Dec 2024 15:05:38 GMT
Message-ID: <AS2-20021219_030338@as2_company.dgi_th>
AS2-Version: 1.3
Mime-Version: 1.0
Recipient-Address: http://10.240.1.2:58201/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=sha-256;
     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
2024 15:04:18 GMT with Subject <Signed and encrypted message with 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://10.240.1.2:58201/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-

Appendix B. Change Log (Non-Normative)

This appendix records the substantive changes made during the revision from the original RFC 4130 draft toward the current version of this document.

B.1. General

* Removed all references to AS1/SMTP throughout the draft.
* Included descriptions and references for compressed content, which was previously supported since AS2 version 1.1.
* This draft explicitly allows negotiation of newer RFCs while retaining legacy support, where necessary.
* Updated formatting and cross-references so that all internal "Section X.Y" references are properly linked in HTML and XML output.
* Moved legacy interoperability clarifications to Section 1.2.1 to avoid confusion with normative text.
* Clarified that appendices are non-normative unless otherwise indicated.
* Replaced all references to RFC 3851 with RFC 5751.
* Replaced all references to RFC 3852 with RFC 5652.

B.2. Changes affecting Section 1.2 - Backward Compatibility and Interoperability

* Expanded to explicitly state the dual-reference policy:
  - Implementations MAY interoperate with S/MIME v3.2 (RFC 5751, which obsoletes RFC 3851/3852).
  - Conformant implementations MUST also support S/MIME v4.0 (RFC 8551, which obsoletes RFC 5751).

* *UPDATED (Technical Review)*: Clarified that S/MIME 4.0 is the baseline for conformant implementations, with explicit guidance on when to use AuthEnvelopedData (S/MIME 4.0 with AES-GCM/AES-CCM) versus EnvelopedData (S/MIME 3.2 with AES-CBC or for backward compatibility).

* Added explicit pointer to Section 1.2.1 for legacy interoperability clarifications.

* Clarified that RFC 8551 forms the baseline for new implementations.

* Added new *Section 1.2.1*: Legacy Interoperability (non-normative clarifications):

  - Captures behavior when interoperating with RFC 4130 systems.

  - Explicit note: 3DES withdrawn by NIST in 2024; MAY only be accepted for backward compatibility, SHOULD NOT be generated.

B.3. Changes affecting Section 5.1 - AS2-Version Header

* Retained definitions for AS2-Version 1.0, 1.1, and included the previously supported version 1.2.

* Added *AS2-Version: 1.3*:

  - Defines modernization requirements (SHA-256, AES baseline per RFC 8551).

  - Aligns MDN behavior with RFC 8098.

  - Requires support for multiple-recipient encryption (Section 7.2).

  - States weak algorithms (SHA-1, MD5, 3DES, RC2) SHOULD NOT be generated by conformant 1.3 implementations.

  - Points to Section 1.2.1 for legacy interoperability when communicating with 1.2 or earlier partners.

* Added note about future versioning: minor versions (1.x) remain backward compatible; major version (2.0+) would indicate non-backward-compatible revisions.

B.4. Changes affecting Section 5.3.3 — Message-Id and Original-Message-Id

* Prohibit spaces and control characters in newly generated Message-Id values; recommend removal (not substitution) when constructing from other attributes.
* Clarify receiver behavior: implementations are not required to accept malformed Message-Id values; they SHOULD return an MDN with processed/error and a human-readable explanation (per [RFC8098]).
* Keep angle-bracket guidance; brackets are required on send and are not part of the identifier value. Receivers SHOULD NOT reject solely for missing brackets.
* Update normative reference to [RFC5322].

B.5. Changes affecting Sections 5.4 and 5.5 — Reliability and Restart

* Expanded Section 5.4 to clarify that HTTP 102 (Processing) MAY be used under HTTP/1.1 for progress indication but MUST NOT be used under HTTP/2 or HTTP/3.
* Changed previous requirement to close connections before retry to optional behavior; clarified that 102 is deprecated but still permitted for backward compatibility.
* Expanded Section 5.5 to reference the AS2 Reliability ([I-D.draft-duker-as2-reliability-16]) and AS2 Restart ([I-D.draft-harding-as2-restart-02]) drafts.
* Clarified that implementations SHOULD support configurable retry logic and MAY implement restart for large or interrupted transfers.
* Added explicit normative language about duplicate detection using Message-ID.

B.6. Changes affecting Section 6 — Additional AS2-Specific HTTP Headers

* Added new AS2-Product header to identify the sending product and version.
* Defined header format as <product-name>:<major.minor[.patch]>.
* Required inclusion for AS2-Version 1.3 and possibly 2.0 later.

* Clarified that the header enables interoperability diagnostics and implementation-specific workarounds, not capability negotiation.

* Explicitly stated that arbitrary or misleading product names MUST NOT be used.

B.7. Changes affecting Section 7 - Algorithm Requirements

* Introduced a new dedicated section consolidating algorithm requirements.

* *Hash Algorithms*:

  - MUST support SHA-256.

  - SHOULD support SHA-384 or stronger.

  - SHA-1 and MD5 deprecated; SHOULD NOT be generated by conformant implementations.

* *Encryption Algorithms*:

  - MUST support AES-128-CBC; SHOULD support AES-256-CBC.

  - RECOMMENDED to support AES-GCM and AES-CCM modes.

  - 3DES and RC2 deprecated; SHOULD NOT be generated.

  - SHOULD support multiple-recipient encryption (per RFC 8551 § 3.3).

* *UPDATED (Technical Review)*: Added comprehensive key management algorithm requirements:

  - MUST support RSA with minimum 2048-bit key length

  - MAY support ECDH and Diffie-Hellman (RFC 5753)

  - For elliptic curves, SHOULD support NIST P-256 or stronger

* *UPDATED (Technical Review)*: Added new Section 7.2 subsections:

  - "EnvelopedData vs AuthEnvelopedData" - Explicit rules for when to use each:

    o AuthEnvelopedData MUST be used with AES-GCM and AES-CCM

    - o EnvelopedData MUST be used with AES-CBC and for S/MIME 3.2 compatibility

    - o Single content encryption algorithm MUST be used for all recipients

  - "Multiple-Recipient Encryption" - Explains support for multiple recipients of the same content-encryption key

* Added explicit cross-references to Section 1.2.1 for legacy interoperability.

B.8. Changes affecting Section 8 - MDN Processing

* Updated to align MDN behavior with RFC 8098 (superseding RFC 3798).

* Clarified semantics for signed-receipt-micalg:

  - Allow multiple algorithms in header for backward compatibility.

  - Conformance requires selecting one algorithm in Received-content-MIC.

  - SHA-256 set as the default minimum; SHA-1 only permitted for legacy use.

* Relaxed constraints on the content of the Disposition-notification-to header.

* Definitions have been included for additional supported error dispositions.0

* Clarified required MDN fields:

  - Final-Recipient — MUST always be present.

  - Original-Message-ID — REQUIRED and must match the original message exactly.

  - Message-ID in the MDN — optional.

  - Disposition-Notification-To — MAY use any format (email, URL, hostname); receiving systems MUST ignore syntax issues per RFC 4130.

* Updated asynchronous MDN handling to reflect practical implementation realities:

  - HTTP 200-level responses SHOULD be sent immediately after receiving the last byte, before full decryption or validation, to minimize timeout risk.

  - Persistent (keep-alive) connections MAY be used; closing is optional and implementation-dependent.

  - Receipt of a 200-level response only acknowledges receipt, not successful processing.

* Emphasized that asynchronous MDNs are sent as independent HTTP messages per Section 7.3, and clarified that connection handling should not be mandated at the application layer.

* Added standardized disposition-modifier extensions to improve error reporting.

* New recommended modifiers:

  - error: decompression-failed

  - error/duplicate-filename

  - error/illegal-filename

  - error: insufficient-message-security

  - error/invalid-message-id

  - error/unknown-trading-relationship

* Clarified that implementations returning these modifiers MUST include a human-readable explanation in the MDN Explanation field.

B.9. Changes affecting Section 9 — Public Key Certificate Handling

* Revised the opening paragraph to reference the optional *AS2 GET* method, in addition to manual partner onboarding.

* Expanded the section to clarify separate roles and lifecycle management for *TLS certificates* (transport security) and *AS2 certificates* (message signing and encryption).

* *UPDATED (Technical Review)*: Strengthened requirement that TLS and AS2 certificates *MUST NOT* be the same certificate (changed from SHOULD NOT). Using the same certificate for both purposes creates security dependencies and operational risks that must be avoided.

* Required a *minimum RSA key length of 2048 bits* (or equivalent elliptic-curve strength such as P-256).

* Clarified that:

  - *TLS certificates* SHOULD be CA-signed in production; self-signed certificates MAY be used in test environments or by partner agreement, provided they include a *Subject Alternative Name (SAN)* extension with hostname and/or IP address.

  - *UPDATED (Technical Review)*: Added reference to CA/Browser Forum Baseline Requirements (https://cabforum.org/baseline-requirements-documents/) for public-facing TLS certificates.

  - *AS2 certificates* MAY be CA-issued or self-signed per partner policy.

* Added guidance that *AS2 certificate lifetimes* need not mirror the short renewal cycles of TLS certificates; renewal policies SHOULD be independent.

* Recommended *CEM* for automated certificate exchange between partners to reduce manual errors and downtime.

* *UPDATED (Technical Review)*: Clarified AS2 GET certificate retrieval requirements to focus on authentication (verify requester identity) and authorization (ensure only legitimate partners can access certificates) rather than just integrity protection. While certificates are digitally signed and thus tamper-evident, authentication and authorization prevent unauthorized parties from obtaining certificates and mapping trading partner relationships. For self-signed certificates, added requirement for out-of-band fingerprint verification before production use.

* Added operational recommendations:

  - Maintain separate TLS / AS2 certificates.

  - Include SAN extensions in all self-signed certificates.

  - Support configurable expiry-notification mechanisms.

  - *UPDATED (Technical Review)*: Administrators MUST NOT (strengthened from SHOULD NOT) reuse TLS certificates for AS2 message security.

B.10. Changes affecting Section 10 - Security Considerations

* Provided guidance for the usgae of HTTPS and the minimum and recommended usage of TLS versions.
* Expanded discussion of algorithm lifecycle:
  - SHA-1 and MD5 deprecated; 3DES formally withdrawn by NIST (2024).
  - Implementations SHOULD NOT generate deprecated algorithms.
  - Migration guidance provided for interoperability.
* Added explicit references back to Section 7 (Algorithm Requirements) and Section 1.2.1 (Legacy Interoperability).

B.11. Changes affecting Section 11 - IANA Considerations

* Clarified that IANA must update existing MDN registries to reference this specification (replacing RFC 4130).
* Added direct links to IANA registry pages for clarity.

B.12. Updated Message Examples

* *Appendix A*: Updated Message Examples with newer algorithms.

B.13. Formatting and Editorial Updates (Technical Review)

The following formatting and editorial improvements were made based on technical review feedback:

* *Section 1.1 (Applicable RFCs)*:
  - Updated RFC references to reflect proper obsolescence chain (RFC 3851 -> RFC 5751 -> RFC 8551)
  - Added RFC 5751 and RFC 5652 to normative references
  - Added explicit explanation of when to use AuthEnvelopedData vs EnvelopedData based on encryption algorithm choice
* *Section 1.3 (Algorithm Coverage)*:
  - Expanded hash function section to include encryption algorithms
  - Added key management algorithm requirements for ECDH

- Added RFC 5753 to informative references

* *Section 6 (AS2-Specific HTTP Headers)*:

  - Reformatted AS2-Version header descriptions to comply with RFC formatting requirements (72-character line limit)

  - Removed markdown artifacts {:format="none"} from all cross-references throughout the document (24 instances)

* *RFC Reference Sections*:

  - Moved RFC citations from section titles to first sentence of each section (9 sections in "Referenced RFCs and Their Contributions")

  - Example: "## RFC 2616 HTTP v1.1 [RFC2616]" became "## RFC 2616 HTTP v1.1" with "[RFC2616] specifies..." in text

* *Capitalization*:

  - Corrected "internet" to "Internet" (2 instances)

  - Ensured consistent capitalization throughout document

* *Cross-References*:

  - Standardized all internal section references to use plain markdown format without formatting directives

* *Bullet Formatting*:

  - Standardized all sections requiring bullets to use the same type and and same spacing and margins.

These updates improve RFC formatting compliance and document clarity while maintaining all technical content and backward compatibility requirements.

Author’s Address

Debra Petta
Drummond Group, LLC
Email: debrap@drummondgroup.com