A-６ EDIINT AS2 概要

A-６ EDIINT AS2 概要

A-６-１ AS2概要

ここでは、以下のドキュメントを参考資料として利用している。
「AS2 Transport Communications Guide for the EAN.UCC GDS（GLOBAL DATA SYNCHRONISATION）N Community」Version 1.1 2004 年 10 月版、「MIME-Based Secure Peer-to-Peer Business Data Interchange Using HTTP, Applicability Statement 2（AS2）」

現時点でも AS2 に関するドキュメントは完成したものではなく改訂が進められる可能性がある。ここでは主だった事項を記述し、詳細に関しては EDIINT AS2 に関する Internet Draft（RFC）を参照することとする。

（１）EDIINT AS2とは

EDIINT AS2 とは、HTTP POST と S/MIME、SSL/TLS を使って安全に EDI メッセージを交換する規定である。
詳細は、参照ドキュメントの記述を確認すること。

EDIINT AS2 にて実現される EDI メッセージの交換パターンは 12 種が想定される。GDS（Global Data Synchronisation）メッセージを交換するに当たっては、以下の要件を満たす範囲で使用する事を前提とする。

1) SSL（TLS）を利用する。
2) EDIINT AS2 メッセージで署名を利用する。
3) 同期あるいは非同期での MDN を利用する。
4) MDN への署名を利用する。

（2）メッセージ送信・メッセージ受信

EDIINT AS2 メッセージを作成し送信する際には、下記のようなヘッダを生成/付与した上で、HTTP 接続した対向システムへ送信する。

同様に、EDIINT AS2 メッセージを受信し解析する際には、下記のようなヘッダを解釈し、GDS（Global Data Synchronisation）メッセージを取り出す。

図表 A-6-1：メッセージヘッダ情報

	ヘッダ名	補 足 説 明
HTTP ヘッダ	Date	送信日時
	Message-ID	EDIINT AS2 メッセージ ID
	Disposition-notification-to	MDN 要求ヘッダ（要求時のみ設定） 【設定値】Disposition-notification-to: mdn@as2
	Disposition-notification -options	MDN 要求オプション（要求時のみ設定） 【設定値】 Disposition-notification-options:signed-receipt-protocol =optional, pkcs7-signature; signed-receipt-micalg=optional, sha1
	Content-Type	
	User-Agent	
	Host	ホスト名：ポート番号
	Accept	対応 MIME タイプ
	Connection	接続状態
	Content-Length	メッセージの長さ
EDIINT AS2 ヘッダ	AS2-Version	サポートする AS2 のバージョン
	AS2-From	送信元の AS2 対応識別子
	AS2-To	送信先の AS2 対応識別子

（３）MDN解析

EDIINT AS2 メッセージ受信側では、送付された GDS（Global Data Synchronisation）メッセージを取り出した後、状況に応じて MDN を生成する。送信側ではその MDN を解析し、EDIINT AS2 メッセージのステータス管理を行う。以下に MDN 解析の処理概要を記述する。

MDN を要求した場合、GDS（Global Data Synchronisation）メッセージ受信者から送られてくる応答 MDN を解析し、以下の点を確認する。

・ メッセージ受信者の認証
・ メッセージの完全性

1) MDN ヘッダ項目

MDN を受信後、以下のヘッダ項目のチェックを行う。

ヘッダ項目は MDN の同期・非同期の別により異なる可能性があるが、代表的なものを挙げている。

図表 A-６-2：MDN ヘッダ項目

情報格納場所	ヘッダ名	説　　明
HTTP ヘッダ	Message-Id	MDN に付与された Message-Id
	Date	MDN 送信日付
	Content-Type	Content-Type
	AS2-Version	EDIINT AS2 のバージョン
	AS2-From	MDN 送信元の AS2 対応識別子
	AS2-To	MDN 送信先の AS2 対応識別子
MIME ボディ	Reporting-UA	MDN 作成アプリケーション
	Final-Recipient	データ受信者
	Original-Message-Id	送信した MIME メッセージの Message-Id
	Disposition	処理結果
	Received-content-MIC	MIC 情報

以下、各ヘッダ項目について説明する。

【Disposition】

Disposition ヘッダには処理結果が記述されている。

メッセージ送信先の処理においてエラーが発生した場合、Disposition ヘッダにエラー情報が記述される。

Disposition ヘッダのエラー情報定義は以下の通り。

“AS2-disposition-type”は、”action-mode/sending-mode”で定義される。

“action-mode”は以下の値を取りうる。

- manual-action：手動で作成された場合
- automatic-action：自動で作成された場合

“sending-mode”は以下の値を取りうる。

- MDN-sent-manually：相手が意図して応答 MDN を送信してきた場合
- MDN-sent-automatically：自動設定で応答 MDN が送信されてきた場合

Disposition ヘッダのエラー情報の値により、該当 GDS（Global Data Synchronisation）メッセージの送受信に関するステータス情報を残す。

図表 A-6-3：エラー情報

処理エラー内容	AS2-disposition-type2	AS2-disposition-modifier-extension	エラータイプ
メッセージ受信エラー	Failed	failure	message-receive-failed
認証エラー	Processed	error	authentication-failed
伸長エラー	Processed	error	decompression-failed
復号エラー	Processed	error	decryption-failed
セキュリティ不十分	Processed	error	insufficient-message-security
改竄の可能性あり	Processed	error	integrity-check-failed
予期しないエラー	Processed	error	unexpected-processing-error
その他エラー	Processed	error	

【Original-Recipient、Final-Recipient】

Original-Recipient は省略されることがある。

Original-Recipient と Final-Recipient の値は異なってはいけない。

【Received-content-MIC】

署名 MDN を要求した場合は必須項目となる。非署名 MDN を要求した場合は、このヘッダは必要ではない。

2) EDIINT AS2 メッセージ受信者の確認

署名 MDN を要求した場合、EDIINT AS2 メッセージ受信者の確認を行う。

署名 MDN を要求した場合、MDN に署名が添付されていなければならない。

署名の確認は、EDIINT AS2 メッセージ受信者の公開鍵で行う。

メッセージ受信者の公開鍵が、CRL に登録されている場合はエラー終了する。

EDIINT AS2 メッセージ受信者側が MDN を作成する上で要求されたプロトコル・フォーマットおよび MIC アルゴリズムを

サポートしていない場合、非署名の MDN が返ってくることがある。この場合はエラー終了する。

3) EDIINT AS2 メッセージの完全性確認

メッセージの完全性を確認する方法は、以下の 2 通り存在する。

・ Message-Id チェック

・ MIC 比較

Message-Id チェックは、必ず行う必要がある。

MIC 比較は、署名 MDN を要求した場合に必ず行う必要がある。

【Message-Id チェック】

EDIINT AS2 メッセージ送信時にヘッダに含めた Message-Id と、受け取った応答 MDN のボディに含まれる Original-Message-Id を比較することで、メッセージの完全性をチェックする。

両者の値が等しい場合を正常とし、メッセージの完全性、及び送達の確認とする。

両者の値が異なる場合はエラー終了する。

【MIC 比較】

MIC とは、ハッシュ関数をもとに計算され、base64 にコード化されたダイジェストである。

Received-Content-MIC ヘッダに含まれる MIC と、EDIINT AS2 メッセージから作成したダイジェストを比較することで完全性をチェックする。

両者の値が等しい場合を正常とし、メッセージの完全性確認とする。

両者の値が異なる場合はエラー終了する。

ダイジェスト作成には、MIC 送信者が使用したハッシュ関数と同じものを使用する。

MIC 送信者が使用したハッシュ関数のアルゴリズムタイプは、MDN ヘッダの Content-Type に含まれている。メッセージ送信者が使用したアルゴリズムタイプと、MIC 送信者が使用したアルゴリズムタイプが異なる場合、エラー終了する。

MIC のデコードは、MDN 送信者の公開鍵で行う。

（4）通信プロトコル部で必要とする情報

通信プロトコル部にて取り扱う必要があると考えられる情報を以下に挙げる。これらの情報は設定ファイルあるいは設定情報を保管する DB、トランザクション情報を保管する DB にて管理される。

図表 A-6-4：通信プロトコルで必要とする情報

種別	項目	意味	補足
メッセージ識別	データ番号	EDIINT AS2 メッセージを識別する為に使用する一意の番号を付与する。	通信プロトコル呼び出し側にて、指定し識別するために利用する。
AS2 接続情報	AS2-FROM	EDIINT AS2 ヘッダ	
	AS2-TO	EDIINT AS2 ヘッダ	
	AS2-VERSION	EDIINT AS2 ヘッダ	
AS2 設定	送信先 URL	送信先 URL	
	MDN 要求有無	MDN 要求の有無	
	MDN 同期・非同期区別	応答 MDN 返送方法同期・非同期の区別	
	MDN 受信プロトコル	プロトコル判別 HTTP/SMTP	
	MDN 署名要求有無	MDN 署名の有無	
	CONTENT-TYPE	HTTP ヘッダに設定する値	
	署名アルゴリズム名	署名アルゴリズム	
署名	圧縮アルゴリズム名	圧縮アルゴリズム	
圧縮	暗号化アルゴリズム名	暗号化アルゴリズム	
暗号化			

（５）その他の詳細

詳細は、参考資料 AS2 20 版を参照のこと。

（６）EDIINT AS2での考慮事項

下表にて、EDIINT AS2 利用上の考慮事項をまとめる。

図表 A-６-5：考慮事項

考慮事項・パラメータ		対応事項
セキュリティ面の考慮事項		SSL（TLS）、S/MIME の利用上で考慮が必要である。 PKI を利用する上での証明書管理、運用、暗号化手法、暗号強度等については、セキュリティ要件を参照のこと。
証明書･鍵の管理と利用	秘密鍵	送信側・受信側の双方にてそれぞれに管理する。署名、復号に用いる。
	公開鍵	送信側・受信側の双方にてそれぞれ相手の公開鍵を使用する。署名確認、暗号化に用いる。
	証明書管理	秘密鍵は安全に保管され、複製を含む盗難を避けなければいけない。秘密鍵、公開鍵は通信システムから EDIINT AS2、SSL/（TLS）の実装プログラムからで必要に応じて参照できる必要がある。
下位プロトコル規約等	セッション層相当	HTTP を使用する。SSL（TLS）を合わせて使用し、通信路の安全を確保する。 必要に応じて HTTP Proxy/Reverse Proxy を用いたアクセス制御が行える。
	トランスポート層相当以下	対向システムから、固定の IP アドレスが割り当てられているものと認識できる。 必要に応じて FQDN や IP アドレスで対向システムを識別でき、ファイアウォールやルータにてアクセス制御が可能である。
	ポート番号	接続するシステム間で相互に合意したものを使用できるよう、設定可能とする。 例（欧米での実績）：4080、1443、443、80 等
AS2 ヘッダ	AS2 Version	圧縮をサポートするバージョンであるが、圧縮を使用するかどうかは相互合意で設定されるものとする。
	AS2-From	GLN を使用する。 EDIINT AS2 メッセージの送信元システムを示す。
	AS2-To	GLN を使用する。 EDIINT AS2 メッセージの受信先システムを示す。
GDS（GLOBAL DATA SYNCHRONISATION）メッセージ格納方法	MIME	application/xmlmessage/disposition-notification（MDN）
	S/MIME	署名を必須とする。 暗号化は選択可能とする。
	圧縮	行わない（必須としない）。 選択を可能とする。
Message Disposition Notification の利用		必須 但し、送信側の指定による選択となる。
	MDN への署名	必須

	同期・非同期	いずれかを選択する。 セッションタイムアウトへの対策として非同期を利用できる。

A-６-２参考

詳細設計と実装へ向けての参考情報を記述する。

（１）EDIINT AS2とebXML Messaging Service

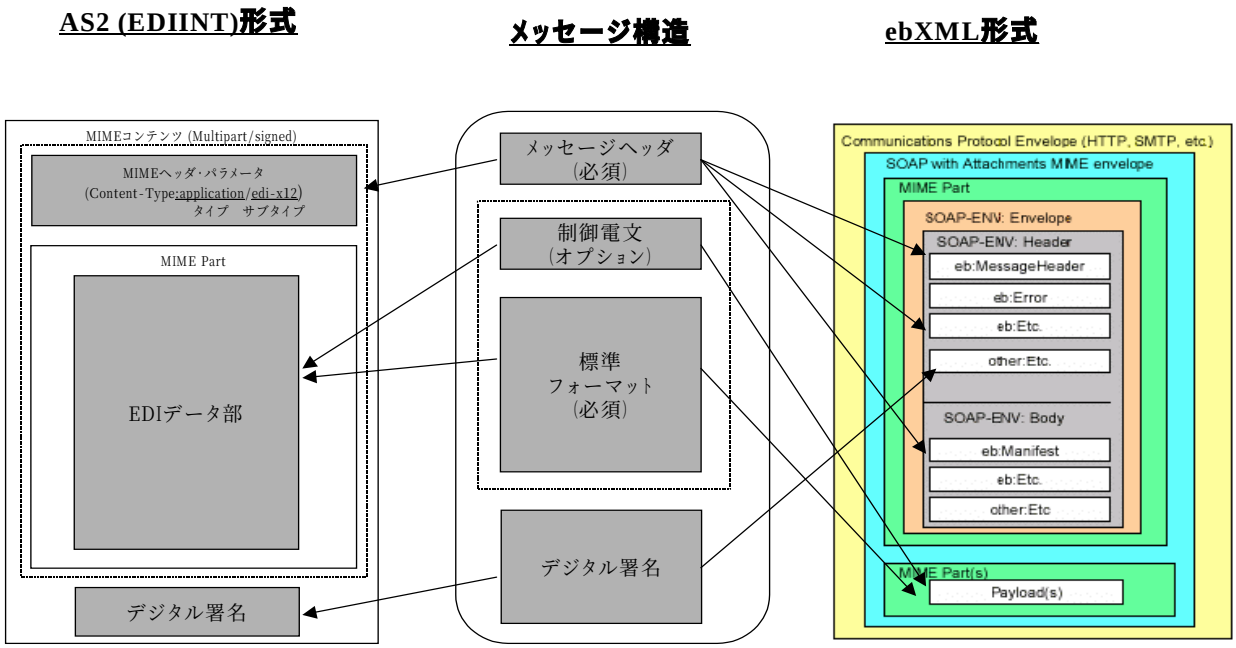
EDI ドキュメント(商品マスタ同期では GDS（Global Data Synchronisation）メッセージ)を安全に搬送する通信プロトコルの役割としては、EDIINT AS2 も ebXML Messaging Service も同様に取り扱うことが可能である。ここでは、実装上通信プロトコルの差異により上位のアプリケーションが影響を受ける事が無いように、通信プロトコル層へのイメージを正しく認識できるように、その差異を簡単に説明する。

EDIINT AS2 では、基本的に MIME で構造が定義されている。一方 ebXML Messaging Service では、XML でその構造を定義している。一般的な EDI ドキュメント搬送をインターネットで行う場合の最小限のメッセージ構造を定義すると、以下の項目に分けることができる。

1) メッセージヘッダ：EDI ドキュメントの送信元、送信先、解析方法等を示す情報が提供される部分。
2) EDI ドキュメント：EDI ドキュメントの本体である何らかの取り決めに従った「標準フォーマット」のデータ、
   およびそのデータの種別や一意識別情報等を示す「制御電文」
3) デジタル署名：改ざんを防止する為の署名情報。

これらの項目を EDIINT AS2 および ebXML Messaging Service と対比させると、下図の様になる。

図表 A-６-6：



この様に構造定義方法(MIME vs XML)、格納順等が異なるものの、搬送路として必要な格納先を持っているだけであり、通信プロトコル層では取り扱う EDI ドキュメントの内容や表現方法を規定するものではない。

① 圧縮方法について

圧縮を実装する場合は、下記の事項を考慮する。

現在、EDIINT AS2-Version 1.1 においては、RFC3274 の圧縮がサポートされる。S/MIME v3.1(RFC3851)をベースに検討が進められており、以下の名前で公開されている。

draft-ietf-ediint-compression-04

② 証明書交換方法

現在、EDIINT を使用する上での証明書の交換方法についての標準化活動も行われている。これは現在、Internet Draft として公開されている。

EDIINT AS2 で通信を行うシステム間で何らかの証明書交換手順を実装する必要がある場合、これらの標準化動向を確認する事が望まれる。ただし、継続的に検討が進む事を保証されているものではない。

現在は、以下の名前で公開されている。

draft-meadors-certificate-exchange-00

③ 参照 RFC

インターネットを使用した通信を行う為に AS2 を採用するに当たって、参照すべき RFC を以下に挙げる。

RFC 1123	Requirements for Internet Hosts
RFC 2045	MIME Format of Internet Message Bodies
RFC 2046	MIME Media Types
RFC 2049	MIME Conformance Criteria and Examples
RFC 1767	MIME Encapsulation of EDI Objects
RFC 1847	Security Multiparts for MIME
RFC 2298	An Extensible Message Format for Message Disposition Notifications
RFC 2311	S/MIME Version 2 Message Specification
RFC 2312	S/MIME Version 2 Certificate Handling
RFC 2616	Hypertext Transfer Protocol -- HTTP/1.1
RFC 2630	Cryptographic Message Syntax
RFC 2821	Simple Mail Transfer Protocol (SMTP)
RFC 2822	Standard for the Format of Internet Text Messages
RFC 3274	Compressed Data Content Type for Cryptographic Message Syntax（CMS）
RFC 3851	S/MIME Version 3.1 Message Specification

A-６-３通信プロトコル使用の考慮事項

（１）通信プロトコル層への依存度ついて

通信プロトコルに EDIINT AS2 を採用する場合、それ以外のものを採用する場合のいずれであっても、上位の規約で取り決めた事項に影響の無い様に、通信プロトコル層の利用方法に配慮する。

例：通信プロトコル層の挙動によって意味を持たせることの無い様にする。

× MDN の受信そのものを、上位規約の意味とマッピングする

○ MDN 受信結果を反映したステータス情報に対して、上位規約の意味とマッピングする。

（２）添付情報の取り扱い方法について

商品マスタ同期にあたって添付情報(例：画像データ等)を取り扱う必要が発生した場合、現在定められた搬送方法は無い。以下に通信プロトコル部分での添付情報の取り扱い方法について、EDIINT AS2 を使う上での案として例示する。いずれかの案の採用を指定するものではない。

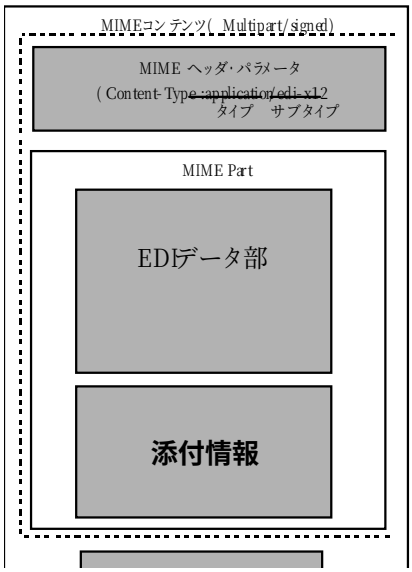
案１：EDIINT AS2 自体を拡張する

EDIINT AS2 では 1 回の接続で複数のファイルを取り扱うような仕様は定義されていない。これに対して、MIME の仕様としては拡張(複数の MIME ボディを持つ)事が可能である。
商品マスタ同期用に EDIINT AS2 を拡張し、添付ファイルを扱えるように複数 MIME ボディをサポートする。

懸念事項：この場合、既存の EDIINT AS2 対応パッケージ製品ではサポートできない形式となる可能性がある。

図表 A-６-7：

AS2 (EDIINT)形式

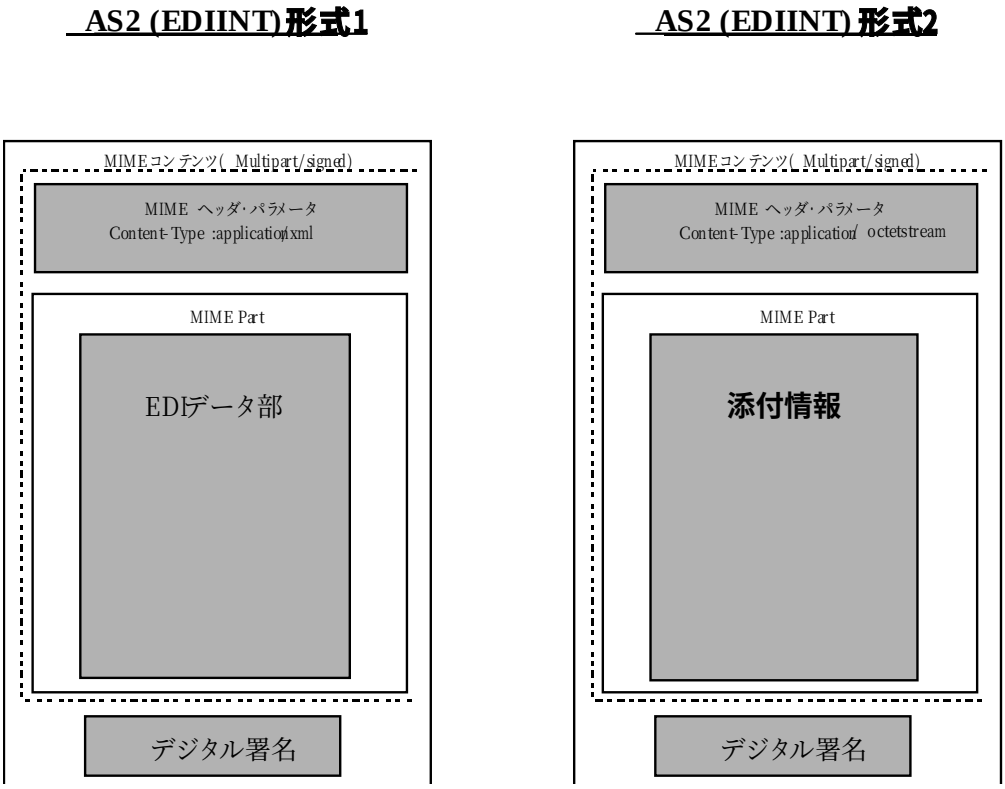


案２：EDIINT AS2 をそのまま使う

EDIINT AS2 では 1 回の接続で１つのファイルを取り扱う。この際に GDS（Global Data Synchronisation）メッセージ中には添付情報があることが分かるように記述し、直後に添付情報を送付する。
添付情報を送る際には先行する GDS（Global Data Synchronisation）メッセージが到達している事を確認してからの送信とする為、MDN による到達確認後の送付とする。
また、該当の GDS（Global Data Synchronisation）メッセージと添付情報を関連付け出来るように同一の AS2-From、AS2-To 組み合わせに関しての後続のデータを送信しない様に制御する機能を必要とする。

懸念事項：この場合、1 件の GDS（Global Data Synchronisation）メッセージの不通により、同一の AS2-From、AS2-To を持つ後続の GDS（Global Data Synchronisation）メッセージが滞る可能性がある。これは別途監視しオペレータへ警告する必要がある。

図表 A-６-8：



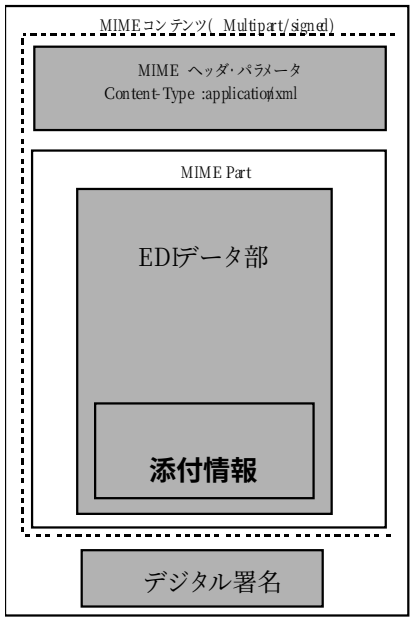
案３：添付ファイルは取り扱わない

通信プロトコル層で添付情報という情報の１つを特別に扱うことを止め、GDS（Global Data Synchronisation）メッセージのスキーマを拡張する形で実現する。
この場合、添付情報(添付ファイル)という考え方ではなく、画像情報も GDS（Global Data Synchronisation）メッセージ中に含まれる１項目の値となる。
この方法では通信プロトコル層に与える影響が最も少なく、実現性も高い。但し GDS（Global Data Synchronisation）メッセージ自体の拡張を伴う為、GDS（Global Data Synchronisation）メッセージのスキーマでの考慮が必要となる。

懸念事項：通信プロトコル層では、特になし。

図表 A-６-9：

AS2 (EDIINT) 形式



A-６-４設計考慮事項

（１）タイムスタンプの運用管理

① 関連システムとの時刻同期

通信プロトコルの稼働環境を含む関連システム全体で、適切に同期された時刻を持つことにより、システム全体の運用監視上の視認性を高め、正常運用時の通信記録の確認、障害発生時の解析、復旧対応に活用できるよう考慮する。

時刻同期の方法には、NTP 等の一般的に普及したコンピュータ間時刻同期の仕組みを用いるものとする。

（2）通信エラーの取り扱い

① TCP 以下のレイヤでのエラーについて

TCP より下層のプロトコルでのエラー発生については、エラー情報を上位アプリケーションへ返すとともに、エラー以前に通信プロトコルで取得した GDS（Global Data Synchronisation）メッセージを含む情報は、エラー情報の記録後に破棄される。

送信側は適切に初期化と再送信を試みる。再送信の最大回数は接続先別に設定できるものとする。

上位アプリケーションへエラー情報が通知された場合には、オペレータに通知すると共に、その記録を残し、再送等の自動的あるいは手動によるオペレーションに使える情報を残す。

オペレータへの通知はメール、SNMP 等が想定されるが、実際に導入する環境に合わせて適切なものを考慮する。

② SSL/TLS

SSL/TLS は通常の HTTP に対する様に動作を行う。

③ HTTP

Result コードでの一般的な通知を行う。

送信側にてエラーが通知された場合には、再送信を試みる。再送信の最大回数は接続先別に設定できる。

Result コード例：

正常時：200 番台

サーバエラー時：500 番台

エラーが通知された場合には、オペレータに通知すると共に、その記録を残し、再送等の自動的あるいは手動によるオペレーションに使える情報を残す。

オペレータへの通知はメール、SNMP 等が想定されるが、実際に導入する環境に合わせて適切なものを考慮する。

④ EDIINT AS2

MDN で通知される結果は記録する。これによりデータの到達を確認する為のステータスを管理する。

MDN でエラーが通知された場合には、自動的な再送処理は行わないことを前提とする。

MDN でエラーが通知された場合には、オペレータに通知すると共に、その記録を残し、再処理等の自動的あるいは手動によるオペレーションに使える情報を残す。

オペレータへの通知はメール、SNMP 等が想定されるが、実際に導入する環境に合わせて適切なものを考慮する。

（３）通信システムの多重化構成について

通信システムの多重化構成を可能とするためには、以下の事項を考慮するものとする。

① GDS（Global Data Synchronization）メッセージ処理部との受け渡し機能

通信システムを多重化構成とした場合に、GDS（Global Data Synchronisation）メッセージ処理部との受け渡しにおいて、多重化構成中の正常稼働中のシステムにより取り扱われるようにする為に以下のような実装を行う。

1) Relational Database(以下 RDB)を用いる方法

・ GDS（Global Data Synchronisation）メッセージを EDIINT AS2 にて受信時、GDS（Global Data Synchronisation）メッセージを取り出し RDB へ記録した後、EDIINT AS2 通信処理を完了する。GDS（Global Data Synchronisation）メッセージ処理部は RDB への新規書き込みを監視し RDB メッセージを読み出して処理する。

・ RDB へのアクセス（メッセージの記録、読出）に際しては、RDB テーブル上に処理タイムスタンプ、
ステータス項目を持ち、競合処理を防止する。

2) Message Queue(以下 MQ)を用いる方法

・ GDS（Global Data Synchronisation）メッセージを EDIINT AS2 にて受信時、GDS（Global Data Synchronisation）メッセージを取り出し MQ へ登録し、EDIINT AS2 通信処理を完了する。GDS（Global Data Synchronisation）メッセージ処理部は MQ を監視し、GDS（Global Data Synchronisation）メッセージを読み出して処理を行う。

・ MQ への登録と読み出しに関しては、別途ステータス管理を行い、競合することの無いような仕組みとする。

以上の様に祖結合の状態とすることで、多重化されたシステム間の同期をとる。
実装に際して必要なレベルで具体的構成を決定する。

② リバースプロキシ利用について

通信システムを多重化した場合に、各サーバへ直接アクセスを防止し、アクセス負荷を平準化する目的でリバースプロキシを利用する事が出来るものとする。

この場合、リバースプロキシの先の対向システムから見た立場での正しい情報を取り扱えるものとする。

例：

リバースプロキシはインターネット上に割り当てられた IP アドレス（＝グローバルアドレス）を使用している。各通信システムに割り当てられた IP アドレスは内部ネットワーク用の IP アドレス（プライベートアドレス）が設定され、使用している。

通信システムが HTTP ヘッダ内に含める IP アドレスや FQDN は、対向システムから見てグローバルアドレスとして解釈できるものでなくてはならない。

（４）障害に対する考慮事項

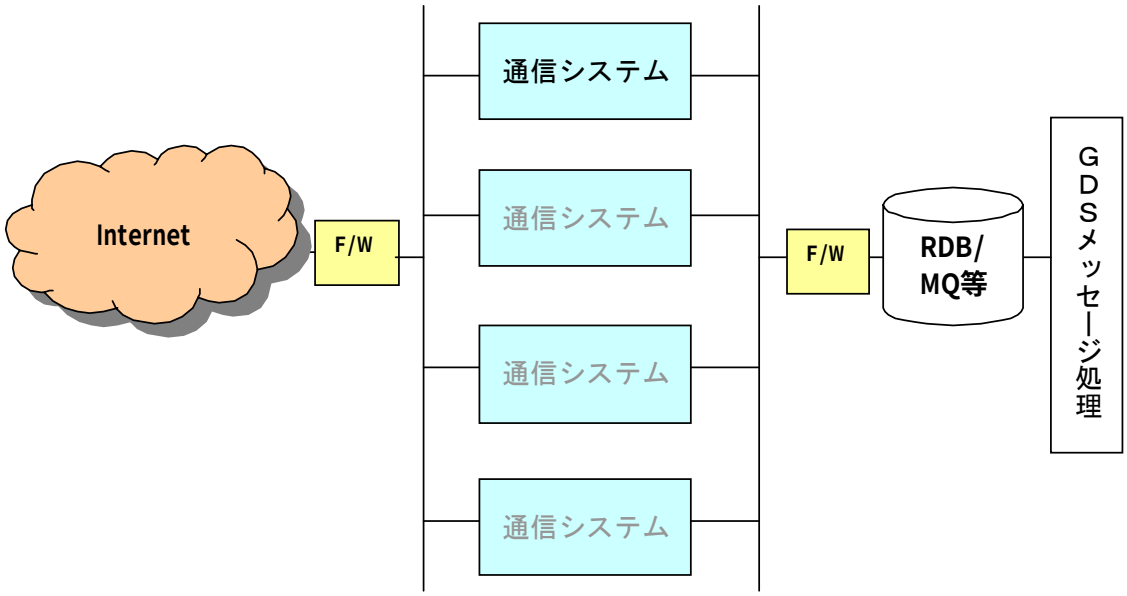
各種障害における影響を以降で検討する。

前提となる検討範囲は下図の「通信システム」部分とその周辺である。

インターネット側を「外部」、GDS（Global Data Synchronisation）メッセージ処理側を「内部」と呼ぶ。

① 想定されるハードウェア障害による影響

図表 A-６-10：



1) 通信システム

通信システムのハードウェア障害では、速やかに該当通信システムの停止を行う事で多重化構成された、他の通信システムへの引継ぎが可能となる。ハードウェア障害発生時点での通信内容は保証されない。使用する記憶装置は障害対策が施されているものとし、ここでは改めて考慮しない。

2) ファイアウォール(F/W)

ファイアウォールは二重化等の障害対策が施されているものとする。

3) GDS（Global Data Synchronisation）メッセージ処理―通信システム間

RDB/MQ 等のミドルウェアを使用して GDS（Global Data Synchronisation）メッセージを受け渡す事が望まれる。この場合、ミドルウェアに二重化等の障害対策が施されているものとする。

但し、簡易に行う場合にはGDS（Global Data Synchronisation）メッセージ処理を呼び出す中継プログラムのみで構成可能とする。

② 想定されるネットワーク設備障害による影響

1) 外部の設備障害について
- インターネット側の設備障害により、対向通信システムとの接続が不可能となる可能性がある。
- この場合、一定回数の再送を繰り返すと共に、オペレータへの通知が行われるものとする。
- オペレータへの通知はメール、SNMP 等が想定されるが、実際に導入する環境に合わせて適切なものを
  考慮する。

2) 内部の設備障害について
- 内部設備障害については、障害機器の差し替えなどにより復旧するか、事前にバックアップルートを
  持つ構成とする。
- 障害の発生は迅速にオペレータへの通知が行われるものとする。
- オペレータへの通知はメール、SNMP 等が想定されるが、実際に導入する環境に合わせて適切なものを考慮する。

③ 想定されるソフトウェア障害等による影響
- ソフトウェアバグ等による障害を含め、意図しない動作となってしまった場合には何らかの再処理ループ、ハングアップが発生する可能性がある。
- ソフトウェア障害を検出する為に、テスト用の GDS（Global Data Synchronisation）メッセージを送受信する機能を実装できるようにする。各所にてテスト用 GDS（Global Data Synchronisation）メッセージを検出し、所定の閾値での異常検出を行い、通知するものとする。
- オペレータへの通知はメール、SNMP 等が想定されるが、実際に導入する環境に合わせて適切なものを考慮する。

④ ウィルスなどに対する考慮事項について
- 暗号化等の実装により、ウィルス対策の為の検索機能などが機能する為には、外部との接続箇所ではなく通信システムより内部側で検索を行う必要がある。
- 送受信される GDS（Global Data Synchronisation）メッセージに対して、ウィルス対策ソフトウェアによるチェックを行う場合には、受信時には通信システムにて GDS（Global Data Synchronisation）メッセージを取り出した後、送信時には通信システムへ GDS（Global Data Synchronisation）メッセージを引き渡す前に、検索を行うものとする。

A-６ 参考資料：EDIINT AS2

INTERNET-DRAFT
draft-ietf-ediint-as2-20.txt
Category: Standards Track
Expires: May 2005

D. Moberg
R. Drummond
21 December 2004

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

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, or will be disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

IPR Statement

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, or will be disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Abstract

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

Feedback Instructions:

NOTE TO RFC EDITOR: This section should be removed by the RFC editor prior to publication.

If you want to provide feedback on this draft, follow these guidelines:

-Send feedback via e-mail to the ietf-ediint list for discussion, with "AS#2" in the Subject field. To enter or follow the discussion, you need to subscribe to ietf-ediint@imc.org.

-Be specific as to what section you are referring to, preferably quoting the portion that needs modification, after which you state your comments.

-If you are recommending some text to be replaced with your suggested text, again, quote the section to be replaced, and be clear on the section in question.

Table of Contents

1.0 Introduction

1.1 Applicable RFCs

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

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

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

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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [13].

1.2 Terms

AS2 - Applicability Statement 2 (this document)

EDI - Electronic Data Interchange

EC - Business to Business Electronic Commerce

B2B - Business to Business

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

Signed Receipt - A receipt with a digital signature.

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

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

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

Non-repudiation of receipt (NRR) - NRR is 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 to be 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.

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

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

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

MIC - The message integrity check (MIC), also called the message digest, is the digest output of the hash algorithm used by the

digital signature. The digital signature is computed over the MIC.

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

2.0 Overview

2.1 Overall Operation

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

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

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

2.2 Purpose of a security guideline for MIME EDI

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

2.3 Definitions

2.3.1 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.

The "secure transmission loop" for EDI/EC involves one organization sending a signed and encrypted EDI/EC interchange to another organization, requesting a signed receipt, followed later by the receiving organization sending this signed receipt back to the sending organization. In other words, the following transpires:

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

To support NRR, the original sender retains records of the message, message-ID, and digest (MIC) value.

- o The receiving organization decrypts the message and verifies the signature, resulting in verified integrity of the data and authenticity of the sender.
- o 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 which, 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 term is used if the acknowledgment is for an interchange resulting in a receipt which is NOT signed. The second term is used if the acknowledgment is for an interchange resulting in a receipt which IS signed.

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

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

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

2.4 Assumptions

2.4.1 EDI/EC process assumptions

o Encrypted object is an EDI/EC Interchange

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

Specifically, in EDI ANSI X12, this means that anything between, and including segments ISA and IEA, is secured. In EDIFACT, this means 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.

o EDI envelope headers are encrypted

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

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

o 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

o Encrypted or unencrypted data

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

o Signed or unsigned data

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

o Use of receipt or not

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

o Use of synchronous or asynchronous receipts

This specification allows in addition to a receipt request the

specification of the type of receipt that should be returned. It supports synchronous or asynchronous receipts in the MDN format specified in section 7 of this document.

o Security Formatting

This specification relies on the guidelines set forth in RFC 3851/3852 [8] "S/MIME Version 3.1 Message Specification; Cryptographic Message Syntax".

o Hash function, message digest choices

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

o Permutation Summary

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

1. Sender sends un-encrypted data, does NOT request a receipt.
2. Sender sends un-encrypted data, requests an unsigned receipt. The receiver sends back the unsigned receipt.
3. Sender sends un-encrypted data, requests a signed receipt. The receiver sends back the signed receipt.
4. Sender sends encrypted data, does NOT request a receipt.
5. Sender sends encrypted data, requests an unsigned receipt. The receiver sends back the unsigned receipt.
6. Sender sends encrypted data, requests a signed receipt. The receiver sends back the signed receipt.
7. Sender sends signed data, does NOT request a signed or unsigned receipt.
8. Sender sends signed data, requests an unsigned receipt. Receiver sends back the unsigned receipt.
9. Sender sends signed data, requests a signed receipt. Receiver sends back the signed receipt.
10. Sender sends encrypted and signed data, does NOT request a signed or unsigned receipt.
11. Sender sends encrypted and signed data, requests an unsigned receipt. Receiver sends back the unsigned receipt.
12. Sender sends encrypted and signed data, requests a signed receipt. Receiver sends back the signed receipt.

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

Additionally, the receipts discussed above may be either synchronous or asynchronous in nature depending on the type requested. The use of

either the synchronous or asynchronous receipts does not change the nature of the "Secure transmission loop" in support of NRR.

3.0 Referenced RFC's and their contribution

3.1 RFC 2616 HTTP v1.1 [3]

This document specifies how data is transferred using HTTP.

3.2 RFC 1847 MIME Security Multiparts [6]

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

3.3 RFC 3462 Multipart/report [9]

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

3.4 RFC 1767 EDI Content [2]

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

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

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

3.6 RFC 3798 Message Disposition Notification [5]

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

3.7 RFC 3851 and 3852 S/MIME Version 3.1 Message Specifications and
Cryptographic Message Syntax (CMS) [8]

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

3.8 RFC 3023 XML Media Types [12]

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

4.0 Structure of an AS2 message

4.1 Introduction

The basic structure of an AS2 messages 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 RFC's
are applied to form the specific structure. For details of how to
code in compliance with all RFC's involved, turn directly to the
RFC's referenced. Any difference between AS2 implantations and RFCs
are mentioned specifically in the sections below.

4.2 Structure of an Internet EDI MIME message

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

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

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

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

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

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

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

While all MIME content types SHOULD be supported. The
following MIME content types MUST be supported:

Content-type: multipart/signed
Content-Type: multipart/report
Content-type: message/disposition-notification
Content-Type: application/PKCS7-signature
Content-Type: application/PKCS7-mime
Content-Type: application/EDI-X12
Content-Type: application/EDIFACT
Content-Type: application/edi-consent
Content-Type: application/XML

5.0 HTTP Considerations

5.1 Sending EDI in HTTP POST Requests

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

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

When using Transport Layer Security [10] or SSLv3, the request-]URI SHOULD indicate the appropriate scheme value, HTTPS. Usually only a multipart/signed message body would be sent using TLS, as encrypted message bodies would be redundant. However, encrypted message bodies are not prohibited.

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

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

5.2 Unused MIME Headers and Operations

5.2.1 Content-Transfer-Encoding not used in HTTP transport

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

encoding of MIME bodyparts within the AS2 message body is also allowed.

5.2.2 Message bodies

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

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

5.3 Modification of MIME or other headers or parameters used

5.3.1 Content-Length

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

5.3.2 Final Recipient and Original Recipient

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

5.3.3 Message-Id and Original-Message-Id

Message-Id and Original-Message-Id is formatted as defined in RFC2822:

"<" id-left "@" id-right ">" (RFC2822 3.6.4)

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

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

5.3.4 Host header

The host request header field MUST be included in the POST request made when sending business data. This field is to allow one server IP address to service multiple hostnames, and potentially conserve IP

addresses. See [3], sections 14.23 and 19.5.1.

5.4 HTTP Response Status Codes

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

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

5.5 HTTP Error Recovery

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

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

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

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

6.0 Additional AS2 Specific HTTP Headers

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

6.1 AS2 Version Header

To promote backward compatibility AS2 includes a version:

AS2-Version: 1.0 - Used in all implementations implementing this specification. 1.x will be interpreted as 1.0 by all implementation implemented with the AS2 Version: 1.0 header. That is only the most significant digit is used as the version identifier for those not implementing additional

non-AS2 specified functionality.
AS2-Version: 1.0 through 1.9 MAY be used
All implementations MUST interpret "1.0 through 1.9" as implementing this specification. However Implementation MAY extend this specification with additional functionality by specifying versions 1.1 through 1.9. If this mechanism is used the additional functionality MUST be completely transparent to implementations with AS2-Version: 1.0 designation.

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

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

6.2 AS2 System Identifiers

To aid the receiving system in identifying the sending system,

AS2-From and AS2-To headers are used.

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

These AS2 headers contain textual values, as described below, identifying the sender/receiver of a data exchange. Their values may be company specific, such as DUNS number, or it may be simply an identification string 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.

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

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

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

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

There is no required response to a client request containing invalid

or unknown AS2-From or AS2-To header values. The receiving AS2 system MAY return an unsigned MDN with an explanation of the error, if the sending system requested an MDN.

7.0 Structure and Processing of an MDN Message

7.1 Introduction

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

The following support for signed receipts is REQUIRED:

1. The ability to create a multipart/report; where the report-type = disposition-notification.
2. The ability to calculate a message integrity check (MIC) on the received message. The calculated MIC value will be returned to the sender of the message inside the signed receipt.
3. The ability to create a multipart/signed content with the message disposition notification as the first body part, and the signature as the second body part.
4. The ability to return the signed receipt to the sending

trading partner.
5. The ability to return either a synchronous or 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 or not, the receiving trading partner's UA MUST provide the following basic processing:

1. If the sent EDI/EC Interchange is encrypted, then the encrypted symmetric key and initialization vector (if applicable) is decrypted using the receiver's private key.
2. The decrypted symmetric encryption key is then used to decrypt the EDI/EC Interchange.
3. The receiving trading partner authenticates signatures in a message using the sender's public key. The authentication algorithm performs the following:

a. The message integrity check (MIC or Message Digest), is decrypted using the sender's public key.
b. A MIC on the signed contents (the MIME header and encoded EDI object, as per RFC 1767) in the message received is calculated using the same oneway 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 oneway hash function that the sending trading partner used is compared for equality.

4. The receiving trading partner formats the MDN and sets the calculated MIC into the "Received-content-MIC" extension field.
5. The receiving trading partner creates a multipart/signed MIME message according to RFC 1847.
6. The MDN is the first part of the multipart/signed message, and the digital signature is created over this MDN, including its MIME headers.
7. The second part of the multipart/signed message contains the digital signature. The "protocol" option specified in the second part of the multipart/signed is as follows:

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

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

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

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

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

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

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

message.

7.2 Synchronous and Asynchronous MDNs

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

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

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

When an asynchronous AS2-MDN is requested by the sender of an AS2

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

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

Synchronous AS2-MDN

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

Asynchronous AS2-MDN

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

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

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

The advantage of the synchronous MDN is that it can provide the sender of the AS2 Message with a verifiable confirmation of message delivery within a synchronous logic flow. However, if the message is relatively large, the time required to process this message and

return an AS2-MDN to the sender on the same TCP/IP connection may exceed the maximum configured time permitted for an IP connection.

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

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

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

7.3 Requesting a Signed Receipt

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

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

Example requesting a MDN:

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

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

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

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

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

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

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

Here is an example requesting the MDN to be asynchronous

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

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

The "receipt-delivery-option: return-url" string indicates the URL

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

where
importance = "required" | "optional"

So the Disposition-notification-options string could be:

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

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

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

Algorithm	Value Used
SHA-1	sha1
MD5	md5

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

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

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

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

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

2. The "importance" attribute of "Optional" is defined in the 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 a 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 as well as why the MDN could not be signed. See the Disposition field in section 7.5 for more information.

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

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

7.3.1 Signed receipt considerations

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

The "rule" is:

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 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, that the UA send back at a minimum, an unsigned receipt. If a signed receipt however 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 for 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:

- o 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.

- o 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.

- o For unsigned, unencrypted messages, the MIC MUST be calculated over the message contents without the MIME or any other RFC 822 headers, since these are sometimes altered or reordered by MTAs.

7.4 MDN Format and Values

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

7.4.1 AS2-MDN general formats

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

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

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

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

AS2-async-http-MDN =
```

```
        Request-Line
        *(( general-header | request-header | entity-header )
        CRLF )
Moberg, Drummond         Expires - May 2005                 [Page 24]
Internet-Draft         MIME-based Secure Peer-to-Peer      December 2004

        CRLF
        AS2-MDN-body

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

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

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

7.4.2 AS2-MDN construction

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

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

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

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

7.4.3 AS2-MDN fields

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

AS2-MDN field names (e.g. "Disposition:", "Final-Recipient:") are case-insensitive (cf. RFC 3798, 3.1.1). AS2-MDN action-modes, sending-modes, AS2-disposition-types, and AS2-disposition-modifier values that 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, AS2-MDN-failure-description nor for the values of any (optional) error, warning, or failure fields.

7.4.4 Additional AS2-MDN programming notes

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

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

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

- Use an action-mode of "automatic-action" when the disposition described by the disposition type was a result of an automatic action, rather than 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 ONLY meaningful with "manual-action", not with "automatic-action".

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

7.5 Disposition Mode, Type, and Modifier

7.5.1 Disposition mode overview

This section would provide a brief overview of how processed, error, failure, and warnings are used.

7.5.2 Successful processing status indication

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

The second part of the disposition-mode is set to "MDN-sent-manually" if the user gave explicit permission for the MDN to be sent. Again, the user MUST not be allowed to explicitly refuse to send a signed receipt when the sender requests one. The second part

of the "disposition-mode" is set to "MDN-sent-automatically" whenever the EDI UA sends the MDN automatically, regardless of whether the sending was under the control of a user, administrator, or software.

Since EDI content is generally handled automatically by the EDI UA, a request for a receipt or signed receipt will generally return the

following in the "disposition-field":

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

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

7.5.3 Unsuccessful processed content

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

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

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

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

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

"Failure: unsupported format"

"Failure: unsupported MIC-algorithms"

7.5.4 Unsuccessful non-content processing

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

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

information may be contained in an error-field.

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

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

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

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

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

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

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

Example

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

7.5.5 Processing warnings

Situations arise in EDI where even if a trading partner cannot be authenticated correctly, the trading partners still agree to continue processing the EDI transactions. Transaction reconciliation is done between the trading partners at a later time. In the content processing warning situations as described above, the "disposition-field' MUST be set to the "processed" disposition-type value, and the "warning" "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 an 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 other than content processing warnings are detected is as follows:

Example:

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

7.5.6 Backwards compatibility with disposition type, modifier and extension

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

multiple error conditions.

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

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

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

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

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

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

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

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

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

7.6 Receipt Reply Considerations in a 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 no errors accessing the request-URI specified processing, the status line in the Response

to the POST request SHOULD be in the 200 range. Status codes in the 200 range SHOULD also be used when an entity is returned (a signed receipt in a multipart/signed content type or an unsigned receipt in a multipart/report). Even when the disposition of the data was an error condition at the authentication, decryption or other higher level, the HTTP status code SHOULD indicate success at the HTTP level.

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

Message Disposition Notifications, when used in the HTTP reply context, will closely parallel a SMTP MDN. For example, the disposition field is a required element in the machine readable

second part of a multipart/report for a MDN. The final-recipient-field ([5] section 3.1) value SHOULD be derived from the entity headers of the request.

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

8.0 Public Key Certificate Handling

In the near term, the exchange of public keys and certification of these keys MUST be handled as part of the process of establishing a trading partnership. The UA and/or EDI application interface must maintain a database of public keys used for encryption or signatures, in addition to the mapping between EDI trading partner ID and RFC 822 [11] email address and http URL/URI. The procedures for establishing a trading partnership and configuring the secure EDI messaging system might vary among trading partners and software packages.

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

It is RECOMMENDED that when trading partners are using S/MIME, that

they also exchange public key certificates considering advice provided in [15].

The message formats useful for certificate exchange are found in [8] and [16].

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

9.0 Security Considerations

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

Extracted from RFC 3851 [8]:
40-bit encryption is considered weak by most cryptographers. Using weak cryptography in S/MIME offers little actual security over sending plaintext. However, other features of S/MIME, such as the

specification of tripleDES and the ability to announce stronger cryptographic capabilities to parties with whom you communicate, allow senders to create messages that use strong encryption. Using weak cryptography is never recommended unless the only alternative is no cryptography. When feasible, sending and receiving agents SHOULD inform senders and recipients of the relative cryptographic strength of messages.

Extracted from RFC 3850 [15]:
When processing certificates, there are many situations where the processing might fail. Because the processing may be done by a user agent, a security gateway, or other program, there is no single way to handle such failures. Just because the methods to handle the failures have not been listed, however, the reader should not assume that they are not important. The opposite is true: if a certificate is not provably valid and associated with the message, the processing software should take immediate and noticeable steps to inform the end user about it.

Some of the many places where signature and certificate checking might fail include:

- o no certificate chain leads to a trusted CA
- o no ability to check the CRL for a certificate
- o an invalid CRL was received
- o the CRL being checked is expired
- o the certificate is expired
- o the certificate has been revoked

There are certainly other instances where a certificate may be

invalid, and it is the responsibility of the processing software to check them all thoroughly, and to decide what to do if the check fails. See RFC 3280 for additional information on certificate path validation.

The following are additional security considerations to those listed in [8] and [15].

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

analyst 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 obtains, 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 to digitally sign 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 be from using client-side SSL, authentication via IPSEC, or use of 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.

So, 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 have been agreed to as 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.

HTTPS Remark

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

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

The URL, which matches the source server identity, SHOULD be carried in the certificate. However, it is not required to make DNS checks or reverse lookups to vouch for the accuracy of the URL or server value. Since server side certificates are to be exchanged and also trust established during the configuration of the trading partner relationship, runtime checks are not required by implementations of this specification.

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

Replay Remark

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

10.0 IANA Considerations

RFC 3335 registered two Disposition-Notification-Options parameters

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

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

RFC 3335 also registered on MDN Extension field name

Extension field name: Received-content-MIC

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

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

10.1.1 Disposition modifier 'warning'

Parameter-name: warning
Semantics: (see sections 7.4.3 and 7.5.5 of this document)

11.0 Acknowledgements

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

12.0 References

12.1 Normative References

[1] N. Borenstein, N.Freed, "Multipurpose Internet Mail Extensions (MIME)

Part One: Format of Internet Message Bodies", RFC 2045, December 02, 1996.

N. Borenstein, N.Freed, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, December 02, 1996.

N. Borenstein, N.Freed, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, December 02, 1996.

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

[3] R. Fielding, J.Gettys, J. Mogul, H. Frystyk, T. Berners- Lee, "Hypertext Transfer Protocol--HTTP/1.1", RFC 2616, March 1997.

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

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

[6] J. Galvin, S. Murphy, S. Crocker, N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, Oct. 3, 1995

[8] B. Ramsdell "S/MIME Version 3.1 Message Specification, RFC 3851, July 2004.

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

[11] D. Crocker, "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August 13, 1982.

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

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

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

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

[16] R. Housley "Cryptographic Message Syntax CMS", RFC 3852, July 2004.

12.2 Informative References

[7] J. Postel, "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1, 1982.

[10] T. Dierks,C. Allen, "The TLS Protocol Version 1.0" RFC 2246, March 1999.

[17] D. Crocker, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November, 1997.

13.0 Authors' Addresses

Dale Moberg
dmoberg@cyclonecommerce.com
Cyclone Commerce
8388 E. Hartford Drive, Suite 100

Scottsdale, AZ 85255 USA

Rik Drummond
rvd2@drummondgroup.com
Drummond Group Inc.
4700 Bryant Irvin Court, Suite 303
Fort Worth, TX 76107 USA

Appendices

A. Message Examples

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

A.1 Signed message requesting a signed, synchronous receipt

```
POST /receive HTTP/1.0
Host: 10.234.160.12:80
User-Agent: AS2 Company Server
```

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



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

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

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

A.2 MDN for Message A.1 Above

```
   HTTP/1.0 200 OK
   AS2-From: 0123456780000
   AS2-To: "¥"  as2Name  ¥""
   AS2-Version: 1.1
   Message-ID: <709700825.1028122454671.JavaMail@ediXchange>
   Content-Type: multipart/signed; micalg=sha1;
        protocol="application/pkcs7-signature";
        boundary="----=_Part_57_648441049.1028122454671"
   Connection: Close
   Content-Length: 1980

   ------=_Part_57_648441049.1028122454671

   & Content-Type: multipart/report;
   & Report-Type=disposition-notification;
   &    boundary="----=_Part_56_1672293592.1028122454656"
   &
   &------=_Part_56_1672293592.1028122454656
   &Content-Type: text/plain
   &Content-Transfer-Encoding: 7bit
   &
   &MDN for -
   & Message ID: <200207310834482A70BF63@¥"~~foo~~¥">
   &  From: "¥"  as2Name  ¥""
   &  To: "0123456780000"
   &  Received on: 2002-07-31 at 09:34:14 (EDT)
```

```
   & Status: processed
   & Comment: This is not a guarantee that the message has
   &  been completely processed or &understood by the receiving
   &  translator
   &
   &------=_Part_56_1672293592.1028122454656
   &Content-Type: message/disposition-notification
   &Content-Transfer-Encoding: 7bit
   &
   &Reporting-UA: AS2 Server
   &Original-Recipient: rfc822; 0123456780000
```



```
   &Final-Recipient: rfc822; 0123456780000
   &Original-Message-ID: <200207310834482A70BF63@¥"~~foo~~¥">
   &Received-content-MIC: 7v7F++fQaNB1sVLFtMRp+dF+eG4=, sha1
   &Disposition: automatic-action/MDN-sent-automatically;
   &  processed
   &
   &------=_Part_56_1672293592.1028122454656--

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

   MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQ
   cp24hMJNbxDKHnlB9jTiQzLwSwo+/9OPc87x+Sc6EpFSUYWGAAAAAAAA
   ------=_Part_57_648441049.1028122454671--
```

Notes:

1. The lines proceeded with "&" is 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 [9], returning the original or portions of the original message in the third body part of the multipart/report is not required. This is an optional body part. However, it is RECOMMENDED that this body part be omitted or left blank.

A.3 Signed, encrypted message requesting a signed, asynchronous receipt

```
   Message-ID: <#as2_company#01#a4260as2_companyout#>
   Date: Thu, 19 Dec 2002 15:04:18 GMT
   From: me@example.com
   Subject: Async MDN request
```

```
Mime-Version: 1.0
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename=smime.p7m
Recipient-Address: 10.240.1.2//
Disposition-Notification-To:
```



```
  http://10.240.1.2:8201/exchange/as2_company
Disposition-Notification-Options: signed-receipt-protocol=optional,
 pkcs7-signature; signed-receipt-micalg=optional,sha1
Receipt-Delivery-Option:
  http://10.240.1.2:8201/exchange/as2_company
AS2-From: as2_company
AS2-To: "AS2 Test"
AS2-Version: 1.1
Host: 10.240.1.2:8101
Connection: close
Content-Length: 3428

  [omitted binary encrypted data]
```

A.4 Asynchronous MDN for Message A.3 Above

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

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

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

The message <x12.edi> sent to Recipient <AS2 Test> on Thu, 19 Dec 2002 15:04:18 GMT with Subject <async MDN request> has been received, the EDI Interchange was successfully decrypted and its integrity was verified. In addition, the sender of the message, Sender

<as2_company> at Location http://10.240.1.2:8201/exchange/as2_company was authenticated as the originator of the message. There is no guarantee however that the EDI interchange was syntactically correct, or 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+vlxlYxmvsA+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/HOzpnEqLqbaBh29y2v82b8bdeGw8pipBQWmf53hlcqHGM
4ZBF3CHw5Wrf1JlE+8TwOzdbal3OzeChw88WfRfD7c/j1flA8sxsujvf2d9j
UxCUga8BVdVB9kHOGeexytytOKvWQXfaEEcgZGUAAAAAAAA=

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

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.