ebXML による次世代 EDI 促進報告書

平成１８年　３月

次世代電子商取引推進協議会

はじめに

UN/CEFACT と OASIS が始めた ebXML 規格群は、昨年度までに主要な規格整備が進められ、現時点で企業間業務プロセス改善のために充分に実装可能な状態になったと言える。しかしながら、当該技術は大企業・中堅企業を中心に普及しつつあるも、中小企業を含めての普及は遅れている。中小企業への普及には、パソコンを利用したクライアント型のシステムの開発と普及が鍵となっており、このためには Pull 型電文搬送を用いたシステムと通信の共通化が必須である。

ebXML による次世代 EDI 推進 WG では、中小企業でも容易に導入・運用が可能なクライアント対応の要件を元に、Pull 型の ebXML 電文搬送サービス仕様を検討し、国際標準団体 OASIS (Organization for the Advancement of Structured Information Standards)に提案・仕様策定活動に参画した。その結果、この機能は ebXML 電文搬送サービス（ebXML MS) V3.0 仕様として技術委員会ドラフトの一部として標準化する見通しが立った。

本報告書は、上記活動成果である Pull 型仕様の要件と作成した標準規約につき報告するとともに、それを含め、現在 OASIS にて審議中の ebXML 電文搬送サービス（ebXML MS) V3.0 仕様技術委員会ドラフトにつき解説している。

本報告書は、第１編「クライアント対応 ebXML 電文搬送サービス仕様開発」と第２編「ebXML 電文搬送サービス第３版（ドラフト)」の２部から構成されている。

第１編は、ユーザー企業のシステム部門及び IT ベンダー/サービスベンダーの一般技術者と管理者を読者として想定しており、本編を理解するにあたっては EDI 及び ebXML の基本的な知識を有することが望ましい。

第２編は、主に IT ベンダー/サービスベンダーのソリューション開発者を対象としており、本編を理解するにあたっては、ebXML 電文搬送サービス第２版および Web サービスの技術仕様（SOAP、WS-Reliability、WS-Security 等）についての知識を前提としている。

本報告書の作成に当っては、次世代電子商取引推進協議会会員を中心とした ebXML による次世代 EDI 推進 WG 委員の方々のご協力を得て作成しました。関係者各位のご理解・ご協力に対して厚く御礼申し上げます。

平成１８年３月

次世代電子商取引推進協議会

ebXML による次世代 EDI 促進 WG 委員名簿

（主査）

成田 雅彦	富士通株式会社

（委員）

大久保　秀典	武蔵工業大学講師
森田　勝弘	県立広島大学
武山　一史	鉄道情報システム株式会社
岩佐 和典	富士通株式会社
黛　　崇	株式会社アルゴ２１
東　　保行	花王インフォネットワーク株式会社
鈴木　俊宏	日本オラクル株式会社
島村　　栄	日本電気株式会社
齋藤　　勉	日本電気株式会社
大沼　保夫	日本ユニシス株式会社
小池　　博	株式会社日立製作所
藤井 慶三	株式会社日立製作所
笠井　利一	富士通株式会社
斉藤　幸則	富士電機ホールディングス株式会社
細田　直正	NEC ソフト株式会社
大林 正晴	株式会社管理工学研究所
長瀬　嘉秀	株式会社テクノロジックアート
梶原　智	株式会社エス・エフ・アイ
坂本　真人	財団法人流通システム開発センター

（オブザーバ）

小林　秀司	経済産業省

（事務局）

菅又　久直	次世代電子商取引推進協議会　主席研究員

目　次

第１編

クライアント対応 ebXML 電文搬送サービス仕様の開発

1. 活動の目的

UN/CEFACT (United Nations Center for Trade Facilitation and Electronic Business)とOASIS (Organization for the Advancement of Structured Information Standard)が始めたebXML 規格群は、平成１６年度までに主要な規格整備が進められ、現時点で企業間業務プロセス改善のために充分に実装可能な状態になったと言える。しかしながら、当該標準に基づく欧米およびアジア地域においてebXMLの導入が着々と進んでいるにもかかわらず、国内産業におけるebXMLの導入促進は期待通りに進んではいない。

ebXML による次世代 EDI 促進 WG は、中小企業も含めた国内産業界への EDI 国際標準ebXMLの普及を図ることを目的に、産業界のニーズを基に、より簡便・安価に導入し運用できるクライアント対応ソリューションの技術仕様を策定し、UN/CEFACT および OASIS を通じて当該標準の整備を促進することを目的とした。

2. 企業間電子取引の状況

2.1 EDI 技術の動向

インターネットとXMLは次世代EDI基盤の重要技術と見られ、その誕生時点から期待されてきた。しかしながら、その実装においては従来からのEDI技術のしがらみと、新しいオブジェクト指向技術の業務プロセスへの適用の葛藤において、幾多の互換性の無い提案がなされてきた。その混乱の中で、UN/CEFACT 傘下にある国と国際業界団体、そしてOASIS会員のITベンダーとユーザーコンソーシアムが協力し、生まれた組織がebXMLイニシャチブであり、そこで開発され2001年5月に合意された仕様として発表されたのがebXML技術仕様群第１版である。当該技術仕様群の一部は、その後ISOに提案され、2005年12月時点、次の5つの技術仕様（CPPA、ebXML MS、RIM、RS、CCTS)がISOの技術仕様（ISO/TS15000)として発行されている。

① ISO 15000-1 TS　CPPA : ebXML コラボレーション規定
② ISO 15000-2 TS　ebMS : ebXML 電文搬送サービス
③ ISO 15000-3 TS　RIM : ebXML レジストリ情報モデル
④ ISO 15000-4 TS　RS : ebXML レジストリサービス
⑤ ISO 15000-5 TS　CCTS : ebXML コア構成要素技術仕様

ところで、企業間で電子的にビジネスを遂行しようとするとき、取引に関与する企業及び企業のコンピュータシステムは、業務の側面とコンピュータ実装の側面の二面で整合をとる必要がある。

業務の側面では、取引企業間の業務遂行の流れについての約束事を両者で明示的に取り決め、それぞれの業務遂行の流れに付随する情報についての定義と記述方式(XML スキーマ等)に関する合意が必要である。

コンピュータ実装の側面では、情報伝達を行うネットワークの通信方式、及び情報伝送における信頼性と安全性確保のための方式について相互のシステムで取り決めておく必要がある。

ebXML は業務の側面の定義（「業務プロセスモデル化標準」「情報モデル化標準」）とコンピュータ実装の側面の取り決め（「コラボレーション規定(CPPA)」「電文搬送サービス標準(電文サー

ビス)」）及びその両面を支援する情報格納庫（「レジストリ・リポジトリ」）の技術仕様群を提供する、インターネットと XML を有効活用した企業間電子取引のためのフレームワークである。

このうちコンピュータ間における情報の伝達のための規格として、ebXML では、インターネット上の通信の信頼性と安全性を高めるために、「ebXML 電文搬送サービス技術仕様(ebXML MS)」(ISO TS15000-2)」を規定している。この仕様は、(財) 流通システム開発センターや (社) 電子情報技術産業協会などで検討・開発されている EDI システムで採用されている。

ebXML 電文搬送サービス(ebXML MS)における信頼性では、情報の到達確認、情報の順序保証及び情報の重複回避の機能を提供している。情報の到達確認手法は、送信した情報に対する受信確認を受けとるまで何度も一定間隔で同じ情報を送り続け、両者で合意した送信繰り返し限度まで行う仕組みである。また、ebXML 電文搬送サービス(ebXML MS)における安全性の確保においては、通信経路の暗号化による保護、及び電子署名による発信人確認と改ざん防止機能を提供している。

2.2 EDI の普及状況

日本の企業全体の EDI 導入率は 8。1%であり低いレベルである。このうち大企業(資本金 3 億円以上)の EDI の導入率は 23。8%であり、必ずしも高くない。中小企業(資本金 3 億円未満)の EDI の導入率は 7。9%と低い（表 1 参照)。

表 1 日本企業全体の EDI の導入率

企業分類(資本金)	企業総数		B2B-EC 導入	
	企業数	%	企業数	%
大企業(3 億円以上)	14、705	0。9	3、499	23。8
中小企業(3 億円未満)	1、602、830	99。1	127、125	7。9
全体	1、617、535	100。0	131、020	8。1

出典：平成 13 年度事業所・企業統計調査（総務省統計局）

日本の企業総数は約 162 万社であり、この内中小企業が 99。1%を占める。今後、EDI の導入を拡大するには中小企業への EDI 導入が不可欠であるあることがわかる。

IT 戦略本部は、2006 年 1 月 19 日に公開した IT 新改革戦略の中で、企業が電子商取引のための汎用的な共通基盤を開発し、2010 年度までに、電子商取引を実施する企業の割合を 60%以上とするだけでなく、中小企業の取引先のうち電子商取引を実施する企業の割合を 50%以上とする中小企業を含めた電子取引の普及の実現を目指している。このことからも、中小企業への EDI 普及が重要なテーマであることがわかる。

3. 中小企業向け EDI の要件

産業全体の電子取引システムは、現状、図 1 のような構造を持っている。

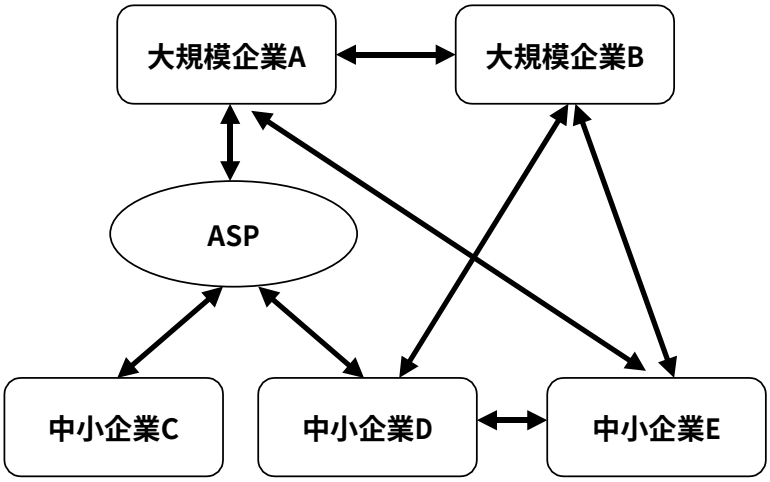

図 1 産業全体の電子取引システムの構造

大企業・中堅企業は、自社の調達システム・販売システムや、ASP を利用して受発注処理を行っている。中小企業は WebEDI や FAX を利用した直接大企業・中堅企業との直接接続や、ASP（アプリケーション・サービス・プロバイダー）経由で取引情報のやり取りを行っている。

したがって、EDI 普及の要件の中で、中小企業に特有なものは以下の通りである。

① 汎用パソコンを使用した EDI システムで受注・発注もできること。

中小企業の EDI 導入に関する問題点・課題の一つとしてユーザー体制・能力の問題がある。中小企業経営者の電子商取引導入に対する意識は高いものの、「電子商取引を行う人的環境が整っていない」、「システム構築に専門知識を要するのでシステム構築できない」、及び「セキュリティ対策が十分に構築できない」のような指摘がある。パソコン利用の EDI システムといっても、サーバの運用・メンテナンスは困難である。情報システムの導入や運用、メンテナンスに費やせるリソースは限られており、企業間取引のためにサーバを立ち上げ、ファイアウォールなどのセキュリティを確保し、運用・メンテナンスを適切に行うことは人的リソースの面でも、技術的にみても困難であるからである。

② 運用が容易であること

FAX や電子メールで Excel の発注書を送受信する程度の手軽さで使えることが必須である。さらに、図 1 の環境下で複数の取引先と単一のインタフェース（通信プロトコル、セキュリティを含む）で利用できることも重要である。これは、EDI 送受信機能、EDI 標準電文、及び画面・帳票レイアウトなどが該当する。

4. 課題の解決

4.1 Pull 型電文搬送による解決

従来のEDIシステムはサーバの導入とそれを用いたebXML MS V2.0などの電文搬送方式が前提となっていた。これは、送信者から EDI データを、受信者に送付する方式(Push 型)である。これは、事象の発生と同時に EDI データが受け取れるという速報性に優れている。しかしながら、常に受信待ちをするため、サーバの導入、サーバの 24 時間運用、インターネットの常時接続、固定 IP アドレスの取得、ファイアウォールなどの強固なセキュリティが要求される。

一般に中小企業はこのようなサーバ設置にまつわる運用上の要件を満たしにくい。この問題は、受信者がパソコンをクライアントとして用いて、送信者のサーバコンピュータに、EDI データを取りに行くことで解決できる。このときに利用するのが Pull 型電文搬送(図 2)である。

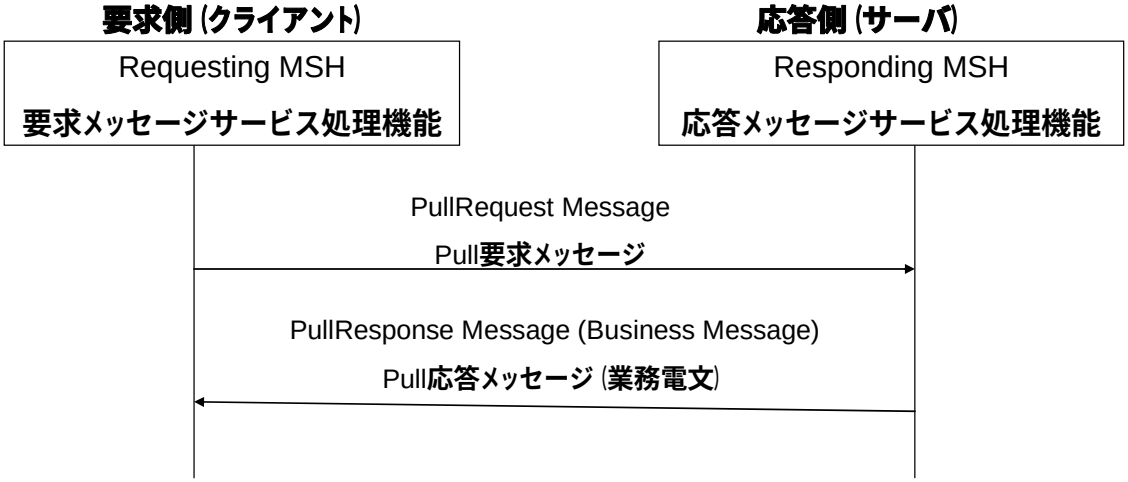

図 2 Pull 型電文搬送の原理

ここでは、図2において、左側のクライアントがHTTPベースの電文を受信する例を説明する。

①右側のサーバがクライアントに電文を送信する際に、直接送信せずに、受信者が電文を取りに来るまで保管する。

②クライアントが電文を受信したいときに、サーバ側に自分あての電文があるか問い合わせる(PullRequest)。通常この電文は HTTP Request で送信される。

③サーバが、クライアントから PullRequest 電文を受信したら、そのクライアント宛ての電文が保存されているか確認する。そのクライアント宛ての電文がある場合には HTTP Response に電文を載せて送る。

Pull 型電文搬送により、サーバからクライアントに電文を送信しなくても、電文を受け渡すことができる。したがって、事象の発生に対する即時性には劣るが、クライアントの起動は、電文の処理の時に限られるので 24 時間運転は不要、インターネットへの接続はダイアルアップでもよく、固定 IP アドレスが不要という利点がある。クライアント型システムを用いれば、中小企業の要件を満たし、安価な EDI 導入と簡単な運用管理が実現する。

4.2 Pull 型電文に必要とされる機能

Pull 型電文を実際に利用するには、前節で述べた基本的な電文の転送機能の他に、以下もあわせて必要とされる。

① クライアントからの接続時の簡易認証
② クライアント側から電文取得するための業務手続。
③ 緊急の電文を通常の電文に優先して取り出すための、電文アクセスの優先度制御
④ 電文の再送・システムダウン時の回復処理を行なうアプリケーションによらないリライアブル電文機能のサポートも望ましい。

①は、簡易認証、②は、例えば、受信者が一日何回か定期的にサーバに接続し、電文の転送要求を出す。電文があれば、その戻りとして電文が転送するという、業務手続をアプリケーションで設定することで解決する。③は、用途別の電文 BOX 機能を Pull 型電文搬送の機能として用意することで、④は、既存のリライアブル電文搬送機能を取り込むことで解決できる。

4.3 プロトコルの標準化

一つの中小企業は一般に複数の大手企業と取引関係を持っているが、受注のために複数の異なる受注システムを保持することは、コスト・運用の容易性に反する。したがって、異なる企業のサーバで同じプロトコルをサポートするのが必須である。また、プロトコルの統一は、複数の ASP に接続して利用する場合にも、運用の簡易化に貢献できる。

プロトコルの統一には、プロトコルの標準化が必要である。現在、OASIS で標準化された電文搬送のプロトコルは、ebXML MS V2.0 、WS-Reliablity 1.1 があるが、すべて Push 型であるので、新たに、Pull 型電文搬送仕様の標準化が必要になる。

Pull 型電文搬送を利用する際、サーバ側は、クライアントサポートのための Pull 型電文搬送と既存のサーバ間通信の電文搬送の２つのサーバを設置することになるので、コスト・メンテナンス性が問題になる。既存の仕様と独立の仕様ではなく、サーバ間通信の電文搬送と同じ一つの体系でサポートすれば、一つのサーバシステムで実現でき、コスト低減と運用の容易性が実現できる。

5. Pull 型電文を用いたシステムの試み

流通業界、電子工業界、及び中小企業共有 EDI を目指して行われている Pull 型電文を用いた中小企業向け EDI システムの実証結果と普及の試みを概観する。

5.1 流通業界 （財）流通システム開発センターの取り組み

流通業界では、1980 年代より流通業界標準の通信プロトコルとして「J 手順」が広く普及している。小売と卸／メーカー間における受発注業務に関しては約９割以上で採用されている。 J 手順は端末機からホストコンピュータに対し起動をかける端末起動型であり、Pull 型での電文交換を行う仕組みである。端末側では多くの管理機能は持たず、通信を行うために最小限必要な通信

先の管理を行う程度である。電文の重複送信・受信に関してはホストコンピュータ側で管理を行い、端末側の負荷を最小限にとどめるような仕組みとなっている。現在までにインターネットを使用した XML-EDI の調査研究を進める中で、通信手順についても国際標準等の調査研究を行っているが、グローバルな視点ではサーバ・サーバ間のいわゆる Push 型の通信手順しか標準仕様として公開されていない現状である。我が国の流通業界、特に中小企業が容易に新たな EDI の仕組みを導入するためには Pull 型で且つ軽量な(通信を行うための最小限の)機能のみを持った仕組みが必要となっている。これに対し、1972 年に設立され、我が国の流通情報システム化を推進するため、JAN コードをはじめとする商品コードの管理及びその普及、電子商取引に必要な標準電文や通信プロトコルの開発等を行っている（財）流通システム開発センターでは、TCP/IP 通信で一般的に使用されている「SOAP-RPC」による Pull 型の通信方式の定義を作成している。

この一環として、（財）流通システム開発センターでは、2004 年度に ebXML MS V2.0 と SOAP-RPC を使用し XML-EDI の実証実験を行った。この実証実験では、大手小売業と大手卸売業／メーカー、中小卸売業との接続で、サーバ・サーバ型、クライアント・サーバ(ASP)型（図 3 参照)、クライアント・サーバ(Web)型の３パターンを行い、信頼性、性能、導入効果などの検証を行っている。

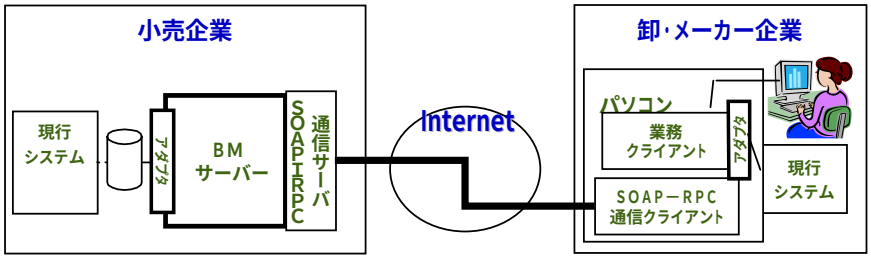

図 3 クライアント・サーバ(ASP)型の接続イメージ

性能の評価では、現行システムで使用されている J 手順と比較した。J 手順は 9600bps の回線で行っているため、インターネットによる回線スピード差が顕著に現れ、従来の 5 倍から 20 倍近くの処理性能が得られた。

5.2 電子電機業界　（社）電子情報技術産業協会　ECALGA の取り組み

製造業を取り巻く環境は「製品ライフサイクル全域を対象とした」、「企業内業務全体の同時並列化」および「企業間業務の同時並列化」に大きく変化している。「企業間業務の同時並列化」を各企業毎にバラバラに実現することは短期的には可能であるが中長期的には非効率である。ECALGA（日本の電子機器・部品業界が主導する企業間の全ビジネスプロセスを電子的にシームレスにつなぎ、相互の経営効率向上を目指す標準の総称）はこのパラダイムシフトに応える標準のソリューションとして 2003 年 12 月に誕生した。「交換の粒度を細かく」「構造化する」「標準化された記法で正確に記述する」ことがコンセプトである。

ECALGA では標準電子取引参照モデル(ISO14662)に習い、業務的視点からビジネスプロセス、

ビジネス情報を規定する BOV (Business Operational View)と、システム的視点から情報技術で実現する方法を規定する FSV(Functional Service View)と呼ばれる 2 つの視点から標準を規定している。

しかし、インターネットを利用して ECALGA を幅広い企業に導入してもらうためには次のような課題もでてきた。

① インターネットを活用した安全・確実な EDI データの送受信
② 価格・技術への対応
③ どこでも(各企業、ASP 等)接続可能な標準への対応
④ グローバルに展開できる仕様

ECALGA では FSV の通信手順として ebXML MS V2.0 を利用する。また、従来のインフラである拡張Z手順も利用できる。しかし、インターネットを利用して、幅広い企業に導入してもらうためには課題も明確になってきた。

その 1 つの解決手段として Pull 型電文を利用した共通クライアントの開発を行うこととした。この共通クライアントは、業務アプリケーションとの容易な連携を実現するために、簡単なコマンド形式による EDI データの送受信、送受信データを格納するためのディレクトリ管理をする機能や、サーバとの間で業務単位のデータ送受信が可能な BOX 機能、エラー表示／エラーログおよびリカバリー機能による容易な運用機能を提供している。また、認証／暗号化／送達確認等によって安全性・信頼性も確保している。一方、接続モデルは、中規模・中小企業まで幅広く EDI ネットワーク化できるように、ASP 経由の企業間接続や企業直結モデルを考慮した。

ソフトウエアの構造は、図 4 のように 3 レイヤに機能分担してある。通信プロトコルは本報告書で策定し、OASIS に提案している ebXML MS V3.0 の Pull 型電文仕様を用いる。

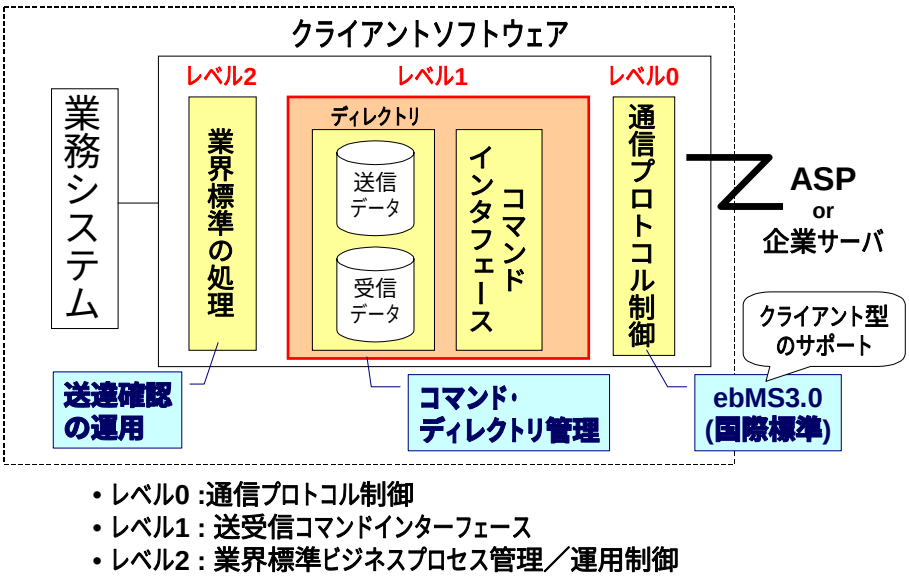

図 4 ソフトウエアの構造(3 レイヤ構造)

5.3 共通 XML/EDI 実用化推進協議会（COXEC）の活動

COXEC は、共通 XML/EDI として、中小企業までの広範囲な普及を可能にする次世代の EDI システムの構築し、ASP モデルを用いた EDI サービスの提供を目指している。現在本団体で設計が進んでいる EDI ネットワークシステムは、現状で標準化・実用化が進んでいる国際標準 ebXML を採用し、かつ実用化が進んでいるインターネット技術・HTTP 通信プロトコル・XML 技術などのソフトウエアを活用している。

共通 XML/EDI フレームワークのクライアントと EDI-ASP サービスとの通信に Pull 型電文搬送採用している。クライアントサービスと共通 EDI-ASP サービスが提供する基本機能には、①Pull 型 EDI 送受信機能、②簡易アプリケーション機能、③EDI 電文蓄積・交換機能、④EDI 標準電文変換機能、⑤ASP 間相互接続機能、がある。これ以外に、EDI-ASP サービスの安全で利便性を向上するために、セキュリティ機能・運用支援機能を用意した（図 5 参照）。

従って、本システムは、中小企業にとっては、手軽な EDI の導入、また、受注だけでなく発注も含む EDI 化ができること、さらには、市販アプリケーションに組み込んでもらうことで、EDI からはじまって業務の IT 化を図ることができる。そして、大企業や業界にとっては、国際標準を採用しているので、真のシングルインタフェースを期待でき、中小企業を含むサプライチェーンや国際取引による業務革新や、今後のｅ文書法、日本版 SOX 法対応等の新たなビジネススタイルに対応できる。

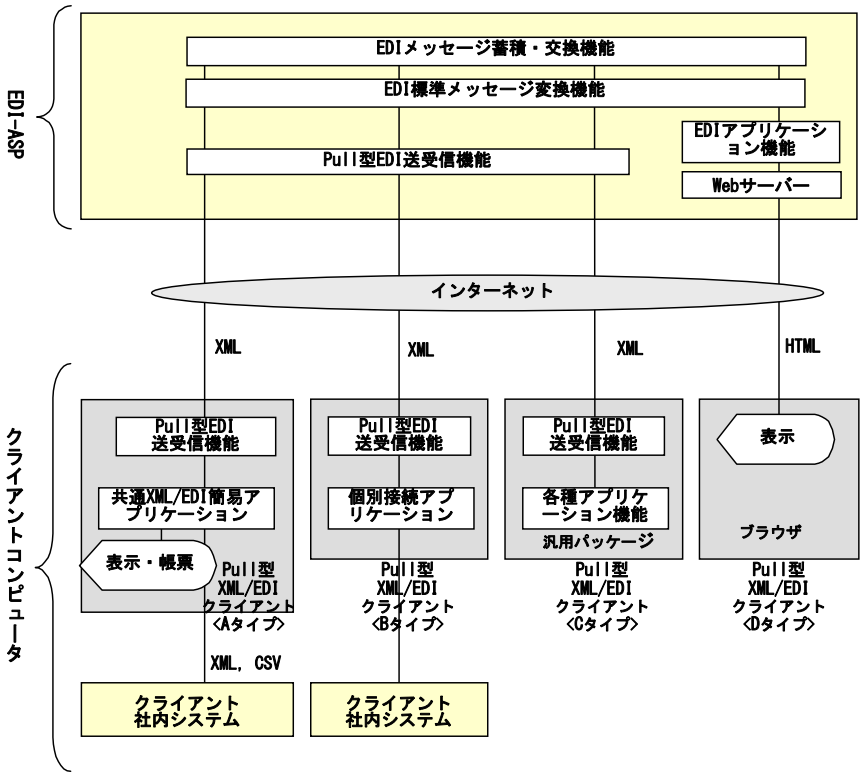

図 5 共通 XML/EDI が提供する基本機能

6. Pull 電文搬送の標準化

本章では、本 WG にて検討し ebXML MS V3.0 の一部として OASIS に提案した Pull 型電文搬送仕様について説明する。

6.1 ebXML MS V3.0 の仕様

2002 年 8 月に OASIS で標準化された ebXML 電文搬送サービス（ebMS）V2.0 は、オープンなの高信頼電文搬送として広く利用されている。現在、その後の市場のニーズ、技術の変化を反映するために、OASIS ebXML 電文搬送サービス技術委員会（Messaging Services TC）において ebXML 電文搬送サービス(MS) V3.0 を策定中で、2005 年 11 月に技術委員会（TC）ドラフトが公開されている。ebXML MS V2.0 からの主な変更点は、Web サービス仕様との整合性確保と本 WG で検討され提案した Pull 電文搬送のサポートである。Web サービス仕様との整合性確保のために追加・修正された仕様は以下のようなものがある。

① 信頼性電文搬送機能として WS-Reliability、セキュリティ技術として WS-Security といった Web サービス仕様を採用した。
② ビジネス電文を SOAP Body でも送信可能とした。

6.2 ebXML 電文搬送サービス技術委員会における議論

前章で述べた（財）流通システム開発センター、（社）電子情報技術産業協会、 共通 XML/EDI 実用化推進協議会の試みから得られた要件を盛り込み、課題解決（4 章）で述べた Pull 型電文搬送仕様を ebXML 電文搬送サービス技術委員会に提案した。この仕様案を標準化する上で、当技術委員会で議論になった点は下記の通りである。

① 電文の種類を指定して受信する機能をどのレイヤで実現するか。結果、極力簡単な機能のみプロトコルレイヤに含め、当該機能の大部分はアプリケーションで実現するとした。
② 簡易運用可能な認証技術としてどの技術を利用するか。検討の結果、デジタル署名の代わりに証明書の不要な ID とパスワードによる簡易的な認証を導入した。

6.3 ebXML MS V3.0 で実現した Pull 電文の仕様

ebXML MS V2.0 で想定されていたサーバ・サーバ間通信の環境以外での電文搬送の利用が理解され、これを実現するために、ebXML MS V3.0 では２つの機能が追加された。

- Pull 型転送モード
- 電文パイプ

これらは、ebXML ヘッダの要素として定義されている。

(1) Pull 型転送モード

Pull 型転送モードのサポートのために、以下の電文交換パターン(MEP: Message Exchange Pattern)を定義した。

Push 型転送（One-way Push）
Pull 型転送（One-way Pull）
要求応答（Request-Reply）
信頼性 Push 型転送（Reliable One-way Push）
信頼性 Pull 型転送（Reliable One-way Pull）
信頼性要求応答（Reliable Request-Reply）

これらは、HTTP・SMTP・FTP などの下位プロトコルとの結合使用を規定している。以下では Pull 型転送（One-way Pull）電文交換パターン（MEP）について述べる。

Pull 型転送電文交換パターン（One-way Pull MEP）は、受信者から送信者へ電文を取りに行く電文交換パターンである。受信側の電文搬送サービス処理機能（MSH:（Message Service Handler)から電文の転送の開始を指示することができる。電文受信側が、いちいち受信のために通信ポートを使用可能にすることなくメッセージを受信できるので、中小企業での利用環境のようにサーバが設置できない場合、常時接続でない場合、固定 IP アドレスでない場合、ファイアウォールによる制限がきつい場合に有効である。

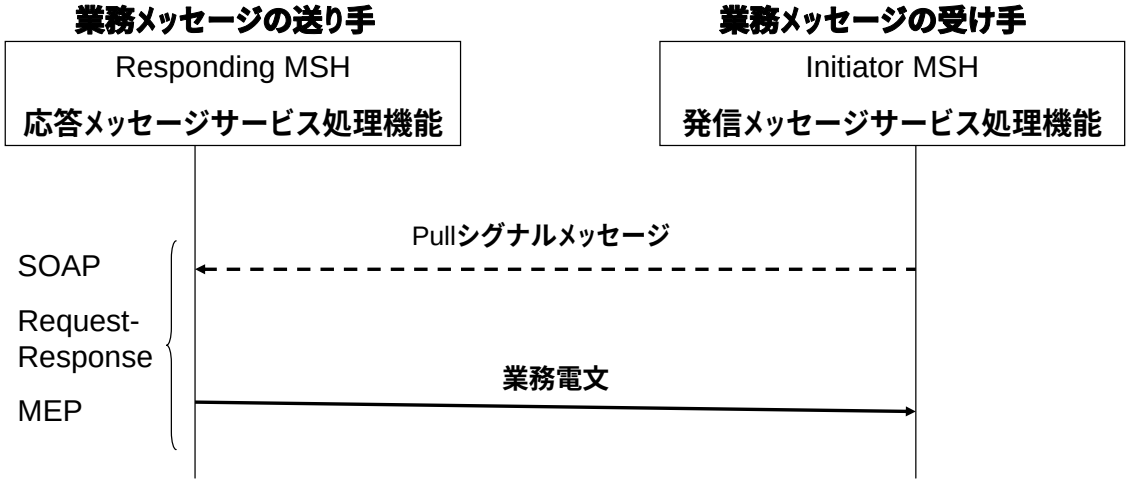

図 6 Pull 型転送（One-way Pull）電文交換パターン

さらに、このシステムでは相手側システムが、この端末への電文の送信のために度々、接続を確認するというオーバヘッドを避けることもできる。

また、信頼性 Pull 型転送（Reliable One-way Pull）と信頼性要求応答（Reliable Request-Reply）は、プロトコルレベルで重複排除や順序保証を行うことができるように、Pull 型電文搬送とリライアブル電文搬送技術(WS-Reliability1.)との結合仕様をとして策定したものである。

⑵ 電文パイプ

電文パイプは送受信する電文を、分類した電文種類毎に独立して送受信するための機能である。受信したい電文種類のパイプを指定して、Pull 電文搬送で読み出すことで、必要な種類の電文の

み受信できる(図 7 参照)。

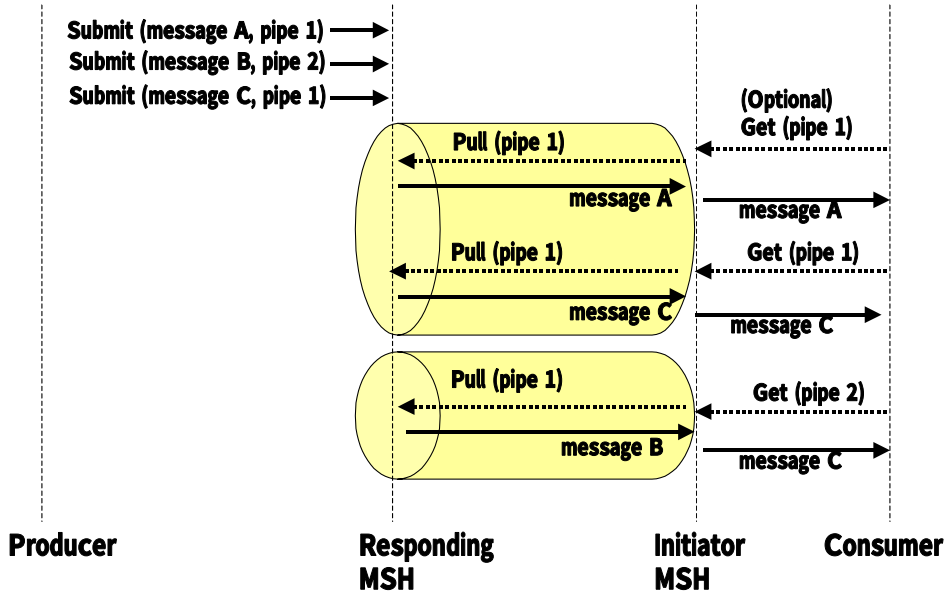

図 7 電文パイプ

どのタイプの伝票がどの電文パイプを使うかを事前に送信側と受信側が定めることで、複雑な選別機能を用意せずとも、本機能を用いて電文転送の優先度制御が実現できる。より複雑な選別は、送信側、あるいは、受信側の MSH で、実装することになる。

7．むすび

インターネットを用いた電子取引と中小企業への普及の課題を整理し、中小企業への普及の鍵となるクライアント型のシステムと Pull 型電文搬送について、実際に各業界で行われている試みを紹介するとともに、Pull 型電文搬送仕様の標準化について論じた。Pull 型電文搬送対応機能を含み、Web サービス仕様との互換性を持つ ebXML MS V3.0 により、次世代 EDI の電文搬送サービスとして一通りの機能整備は完了する。

企業間のビジネスプロセスを遂行していくためには、タイミングを合わせた連続的な情報交換を行っていかなければならない。そのためのベースとなる電文搬送サービス仕様が充実し、それが広く使われるようになることは、オープンネットワーク上での企業間情報共有を実現するための第一歩である。すでに（社）電子情報技術産業協会や（財）流通システム開発センターなど本仕様採用を決定しているが、今後、さらに本仕様を採用したシステムの開発が進むことにより、中小企業を含めた EDI システムの普及へ貢献できると期待している。

第２編

ebXML 電文搬送サービス仕様第３版（ドラフト）

1. 目的と範囲

本仕様は、電子的に取引メッセージを交換するための通信プロトコルに中立な手法について定義する。本仕様は、信頼性と安全性を確保した上で取引情報を交換するために、特定の電文搬送のための構成概念を定義する。さらに、電文があらゆるフォーマット型のペイロード（通信で搬送対象とした業務電文）を包含できるような、柔軟なエンベロープ技術（ペイロードを通信パッケージ上に搭載する仕組み）も定義する。この汎用性により、最新技術のユーザーだけではなく、従来の構文（UN/EDIFACT, ASC X12, HL7 等）を採用している従来の電子取引システムでも、ebXML 基盤の利点を活用できるようになる。

ebXML 電文搬送サービス（以降 ebMS と記述する）の主な目的は、共通のインターネット標準上において XML のフレームワークを利用し、バックエンドの適用業務システムにかかわらず、電子取引業務電文（以降メッセージと記述する）を交換できるようにすることである。

サプライチェーンの当事者で電子的にメッセージを交換するには、メッセージの容量、断続的な接続、静的な IP アドレスの欠如、またはファイアウォール制限における相違を処理することが重要になる。このような新たな性能要件は、最新の SOAP 基盤を必要とするとともに、ebXML 電文搬送サービス仕様第 3 版（以降 ebMS 3.0 と記述する）策定の動機として重要な役割を果たした。標準の業務レベルのヘッダーを格納するメッセージヘッダーの定義法は、ebXML 電文搬送サービス仕様第 2 版（以降 ebMS 2.0 と記述する）で提供されているが、電文やりとりのモニタリングにおける新しい傾向やメッセージ処理機能がサポートできなければならない電子的な取引の側面およびバックエンドシステムとの結合モデルの多様性も扱うように拡張された。

最終的に ebMS 3.0 は、SOAP または ebMS レベルのどちらかの仲介で働くか、あるいは、多様なネットワークトポロジーに要求されるさまざまな役割をサポートする。

ebXML の電文搬送フレームワークは、制限的なものではない。例えば、ebXML メッセージの「ペイロード」として識別されている取引メッセージは、XML 文に限定されない。従来の EDI フォーマットも ebMS によって送受信される。これらのペイロードは、あらゆるフォーマット（XML, ASC X12, HL7, AIAG E5, データベース表、バイナリ画像ファイルなど）を採用できる。ebXML 電文搬送プロトコルの目的は、利用可能なすべての搬送プロトコル上で利用できるようにすることである。このバージョンの仕様は、HTTP、SMTP といったコア機能を規定する。更に、SOAP が結合できる他のプロトコルを使用することもできる。XML フレームワークの選択は、増大する XML ベースの Web 基盤と開発ツールの基盤に対する信頼を反映し、それらの構成要素は開発者たちによって活用および再利用される。

ebXML 基盤は、独立しながらも関連を持つ複数の構成要素からなる。それぞれの構成要素の仕様は、独立型の仕様として構築される。各仕様は自己完結的である。つまり、ある 1 つの実装が他の ebXML の仕様を考慮しない場合がある。ebXML の仕様におけるいくつかの参照と結合は、統合のための要件としてではなく、統合の補助として解釈されなければならない。これは、特に CPPA（コラボレーション規定）の仕様を参照する ebMS にも適用される。つまり、ebMS は、実装に決断を任せる具体的な表現である「同意」（例えば、CPA や他の設定情報）という概念に依存する。

ebMS は、電文搬送の機能、プロトコル、エンベロープを SOAP（SOAP1.1 および

SOAPATTACH）上で動作させることを目的として定義する。HTTP や SMTP などの下位のトランスポート層への結合は、標準の SOAP 結合が存在する場合はそれらに依存し、ebMS は、必要に応じてこれらを補うものだけを特定する。

このバージョンの ebMS は、信頼性と安全性におけるサービスの質を扱う従来の SOAP 基盤の仕様を利用する。EbMS の仕様構成の設計は、これらの標準自体の再利用だけではなく、これらの標準の既存の実装を再利用することも考慮する。

ebMS 実装の概念は、特定の電文搬送機能を実装するものとして抽象的に定義されている ebXML の電文処理機能（MSH: Message Service Handler）にある。MSH(電文処理機能）とのインタフェースについては本仕様の対象外である。多くの場合、標準のアプリケーションインタフェース（API）を定義することは大いに役立つが、このようなインタフェースは、アプリケーションが望む MSH(電文処理機能）との他の相互作用の方法を排除してはならない。このようなインタフェースの定義は、実装ガイドラインの手引きに属する。本仕様は、完全に独立したソフトウェアの構成要素、または、大規模なシステムの組み込み構成要素として利用される。

本仕様は、2006 年 3 月現在、OASIS の ebXML 電文搬送サービス技術委員会にて審議中の委員会案（2005 年 11 月 30 日版）をベースに、技術仕様に関わる部分を翻訳したものである。

なお、本第 2 編は、ebXML 電文搬送サービス第 2 版および Web サービスの技術仕様（SOAP、WS-Reliability、WS-Security 等）を理解する IT 技術者を読者として想定している。

2　電文搬送モデル

2.1　電文搬送の概要

本節では、仕様全体で使用されている関連専門用語とともに、電文搬送モデルおよびその主な概念について説明する。

2.1.1　モデルの構成要素

ebXML 電文搬送サービス 電文搬送モデルは、以下の構成要素を持つ。

- **ebXML 電文搬送サービス MSH （電文処理機能）**：本仕様に準拠する電文を生成または処理し、ebXML 電文搬送サービス の送信と受信の 2 つの役割のうち、少なくとも 1 つとして行動できるエンティティ。SOAP 処理の観点では、MSH(電文処理機能）は単一の SOAP プロセッサーか一連の SOAP プロセッサーのいずれかである。どちらの場合も、MSH(電文処理機能）は”ebms”アクターを対象としたヘッダーを理解できなければならない。
- **生成者（または電文生成者）:** 送信 MSH（つまり、送信という役割における MSH(電文処理機能)）と相互作用してユーザー電文の送信を開始するエンティティ。例としては、アプリケーション、待ち行列システム、他の SOAP プロセッサー（MSH(電文処理機能） は同じ）が挙げられる。
- **利用者（または電文利用者）:** 受信 MSH（つまり、受信という役割における MSH(電文処理機能)）と相互作用して受信した電文からデータを取り込むエンティティ。例としては、アプリ

ケーション、待ち行列システム、他の SOAP プロセッサーが挙げられる。

図 1 は、電文交換に関与するエンティティと動作を表す。

（注）すべての図において、矢印は制御の流れ、つまり、他の構成要素に動作を呼び出す構成要素を表すわけではない。矢印が表すのは、一方の構成要素に実装されている動作が制御するデータ搬送だけである。

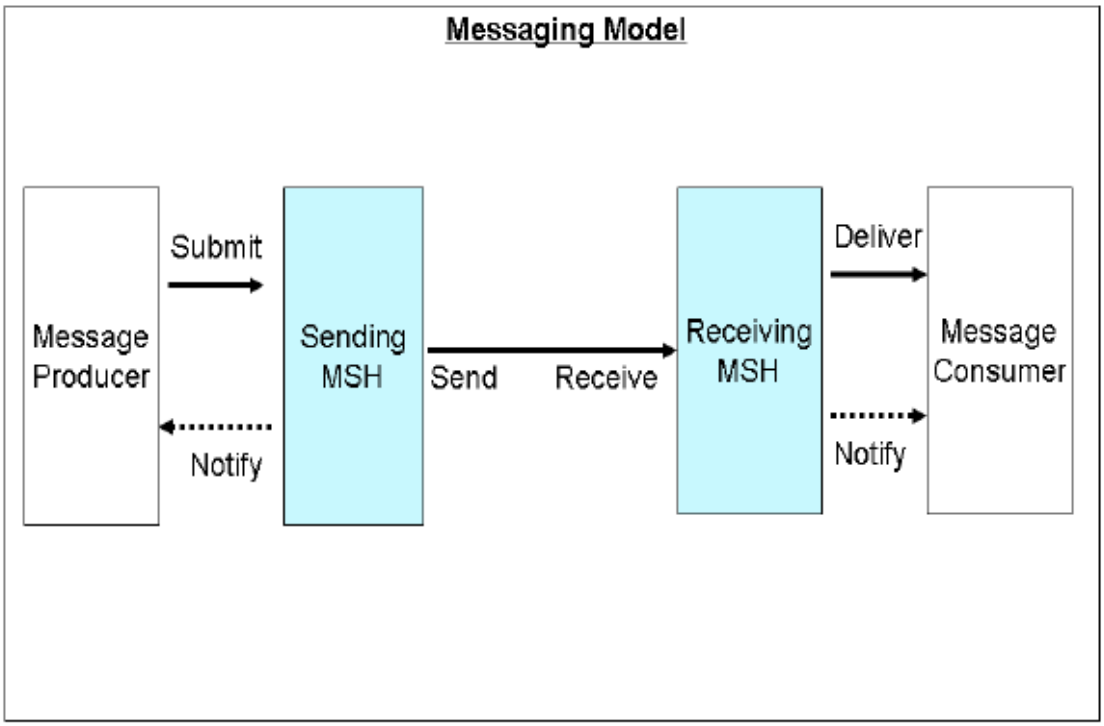

図 1：電文搬送モデルのエンティティとその相互作用

2.1.2 電文型

ebXML 電文搬送サービス 電文とは、ebXML 電文搬送サービス の名前空間に適合した SOAP ヘッダーを含み、かつ本仕様に準拠する電文である。ebXML 電文搬送サービスの MSH(電文処理機能) の構成要素は、以下の ebXML 電文搬送サービス 電文型を交換できなければならない。

- ebXML 電文搬送サービス シグナル電文：受信 MSH(電文処理機能) において特定の機能を活性状態にする役割を持つ ebXML 電文搬送サービス 電文。利用者エンティティに配信されるべきデータは送信しない。つまり、配信動作の影響は受けない。例としては、ebXML 電文搬送サービス の PullRequest シグナルが挙げられる。シグナル電文の特徴は、ebXML 電文搬送サービス ヘッダーに要素 eb:SignalMessage が存在する点である（5.3.1 項参照）。
- ebXML 電文搬送サービス ユーザー電文：（提出動作を通して）生成者エンティティによって

開始される ebXML 電文搬送サービス 電文で、(配信動作を通した) 利用者を対象とする。ユーザー電文の特徴は、ebXML 電文搬送サービス のヘッダーに要素 eb:UserMessage が存在する点である (5.2.2 項参照)。

ebXML 電文搬送サービス 電文ユニットは、ebXML 電文搬送サービス 電文のサブセットであるデータの論理ユニットである。電文ユニットには以下の 2 種類ある。

・ebXML 電文搬送サービス ユーザー電文ユニット。これは、参照されたペイロード項目とともに XML の情報集合 eb:Message/eb:UserMessage で表される。
・ ebXML 電文搬送サービス シグナル電文ユニット。これは、XML の情報集合 infoset eb:Message/eb:SignalMessage で表される。

2.1.3 電文搬送の役割

電文搬送モデルは、MSH(電文処理機能) のための以下の役割を前提とする。

・ 送信 : MSH(電文処理機能) は、ebXML 電文搬送サービス ユーザー電文の生成およびこの電文の他の MSH(電文処理機能) への送信と関連のある機能を実行する際、送信の役割の中で行動する。抽象的な動作である提出、送信、および通知は、この役割によってサポートされる (送信の役割において、MSH(電文処理機能) は、前に送信された電文と関連のあるエラー電文の受信および処理もしなければならない)。
・ 受信 : MSH(電文処理機能) は、ebXML 電文搬送サービス ユーザー電文の受信および処理と関連のある機能を実行する際、受信の役割の中で行動する。抽象的な動作である受信、配信、および通知は、この役割によってサポートされる (受信の役割において、MSH(電文処理機能) は、エラー電文や PullRequest シグナルなどの電文の受信と関連のある ebXML 電文搬送サービス シグナル電文も送信しなければならない)。

ebXML 電文搬送サービス ユーザー電文の伝送には、送信 MSH(電文処理機能) と受信 MSH(電文処理機能) のペアが必要である。これらの役割は、以下の抽象的な動作がそうであるように、ebXML 電文搬送サービス ユーザー電文のみと関連があると定義されている。

2.1.4 抽象的な電文搬送動作

ebXML 電文搬送サービス MSH(電文処理機能) は、それがどの役割の中で動作しているかによって、以下の抽象的な動作をサポートする。

・ 提出 (Submit) : この動作は、ebXML 電文搬送サービス ユーザー電文ユニットを生成する上で十分な電文データを生成者から送信 MSH(電文処理機能) に搬送する。
・ 配信 (Deliver) : この動作は、(受信動作を通して) 前に受信した、利用者が利用可能な ebXML 電文搬送サービス ユーザー電文ユニットのデータを作成する。
・ 通知 (Notify) : この動作は、前に提出または受信された ebXML 電文搬送サービス ユーザー電文ユニットの状態、または、一般的な MSH(電文処理機能) の状態を生成者または利用者に通知する。

・ 送信（Send）：この動作は、ebXML 電文搬送サービスの SOAP アクタ向けのヘッダーがすべて追加された後（必要に応じ、安全性および/または信頼性を含む）、送信 MSH(電文処理機能)から受信 MSH(電文処理機能) に ebXML 電文搬送サービス ユーザー電文の搬送を開始する。
・ 受信（Receive）： この動作は、送信 MSH(電文処理機能) から受信 MSH(電文処理機能) への ebXML 電文搬送サービス ユーザー電文の搬送を完了させる。受信が成功したということは、包含されているユーザー電文ユニットを受信 MSH(電文処理機能) がさらに処理できるようになったことを意味する。

2.2 電文交換パターン

本項では、MEP（ebXML 電文搬送サービス Message Exchange Pattern：ebXML 電文搬送サービス 電文交換パターン）の概念および MEP が SOAP の MEP とどのように関連するかを説明する。このような ebXML 電文搬送サービス の MEP は、振り付けの原子ユニット、つまり、接続制限またはアプリケーション要件によって必要とされるさまざまな交換スタイルを表す。

2.2.1 定義

MEP とは 、2 つ以上の MSH(電文処理機能) のインスタンス間で生じる典型的な一連の ebXML 電文搬送サービス電文交換を抽象的に表現したものである。MEP のインスタンスは、MEP によって表現されるパターンに準拠する電文交換である。具体的には、ebXML 電文搬送サービス 電文ユニットの交換として定義されている。

1 つ以上の ebXML 電文搬送サービス 電文ユニットが同一の MEP インスタンス内で交換される場合、ebXML 電文搬送サービス ヘッダーにおいてそれらの間に明確な参照がなければならない。つまり、同一の MEP インスタンス内で交換されるすべての電文ユニットに対して、以下のいずれかの記述が真となる。

・ この ebXML 電文搬送サービス 電文ユニットは、MEP インスタンス内で生じる最初の ebXML 電文搬送サービス 電文ユニットである。
・ この ebXML 電文搬送サービス 電文ユニットは、同一の MEP インスタンス内において、前に送信された（唯一の）電文ユニットの ID を参照している。

この参照は、要素 eb:RefToMessageId を使用して行われる。MEP インスタンスの最初の ebXML 電文搬送サービス 電文を送信している MSH(電文処理機能) は、開始 MSH(電文処理機能) と呼ばれる。他の MSH(電文処理機能) は応答 MSH(電文処理機能) と呼ばれる。

2.2.2 仮定される SOAP 電文の交換パターン

それぞれの ebXML 電文搬送サービスの MEP は、どの SOAP の MEP をどのように使用しているのかという観点からも定義される。

SOAP の MEP の概念は、SOAP 1.1 にも適用可能だが、SOAP 1.2 のみで説明している。本項は、これら SOAP の MEP の仮定の特性項目のみを考慮し、厳密な定義については考慮しない。

これらの MEP と基礎的なプロトコルとの具体的な結合については、これ以降の項で扱う。基礎的なプロトコルには一方向と双方向の 2 種類あり、双方向の場合には返信電文に暗黙の合意が

あることが仮定されている。

本仕様に記述されている ebXML 電文搬送サービス の MEP をサポートする SOAP の MEP については、以下の 2 つの項で説明する。

2.2.2.1 SOAP の一方向の MEP

本仕様を記述している現時点では、この MEP に正式な定義はない。ここでは、本仕様は、返信電文を必要としない電文を送信する際に、多くの SOAP 実装が示す MEP のプロパティの集合を識別することのみを考慮する。MSH(電文処理機能) の観点から、この MEP のサポートは以下のことを仮定する。

・ 開始 MSH(電文処理機能) は、基礎的なプロトコル上で（双方向のプロトコルの場合は最初の行程で）SOAP エンベロープ（ebXML 電文搬送サービスの電文）の送信を開始できる。

・ 双方向の基礎的なプロトコルが使用されている場合、応答電文は、SOAP エンベロープを含まないか（この場合、SOAP の一方向の MEP は「厳密」だと見なされる）、あるいは、空の SOAP ボディを伴う SOAP エンベロープ、または、ボディ SOAP フォルトがある SOAP エンベロープを含む（この場合、MEP は「頑強」だと見なされる）。

2.2.2.2 SOAP の要求－応答 MEP

この MEP の完全な定義については、 [SOAP12] のパート 1 の付録に記載されている。MSH(電文処理機能) の観点から、この MEP のサポートは以下のことを仮定する。

・ 送信 MSH(電文処理機能) は、基礎的なプロトコル上で（つまり、HTTP GET や POST などの前のプロトコルのアクションの結果としてではなく）SOAP エンベロープの送信を開始できる。

・ 受信 MSH(電文処理機能) は、何らかの形で要求と応答を関連付けた後（例えば、この関連は、HTTP などの要求－応答の基礎的なプロトコルを使用することによって結び付けられる）、SOAP エンベロープ（応答と呼ばれる）を伴う電文を返信できる。

2.2.3 シンプルな ebXML 電文搬送サービス 電文の交換パターン

本項では、ユーザー電文に関する 3 つの基本的な ebXML 電文搬送サービス の MEP について説明する。

ebXML 電文搬送サービス の MEP には以下の 2 種類ある。

・ シンプルな ebXML 電文搬送サービス の MEP : 単一の SOAP の MEP インスタンス上で実行される場合。

・ 集合的な ebXML 電文搬送サービス の MEP : 複数の SOAP の MEP インスタンス上で実行される場合。

以下の項で、3 つのシンプルな ebXML 電文搬送サービス の MEP（一方向の Push、一方向の Pull、要求－応答）について説明する。

2.2.3.1 一方向の Push 電文の交換パターン

この MEP には単一の ebXML 電文搬送サービス ユーザー電文が関与する。電文の送信は、送信 MSH(電文処理機能) によって以下のいずれかとして開始される。

・SOAP の一方向の MEP インスタンス

・SOAP の要求－応答 MEP インスタンスにある SOAP 要求
・SOAP の要求－応答 MEP インスタンスにある SOAP 応答
この MEP に準拠するためには、ユーザー電文は他のユーザー電文（要素 eb:RefToMessageId のない）と関連があったり、次のユーザー電文に参照されたりしてはならない。図 2 は、この MEP の交換パターンと動作を表す。属性 eb:soapresp を「偽」に設定するか、あるいは使用しない。

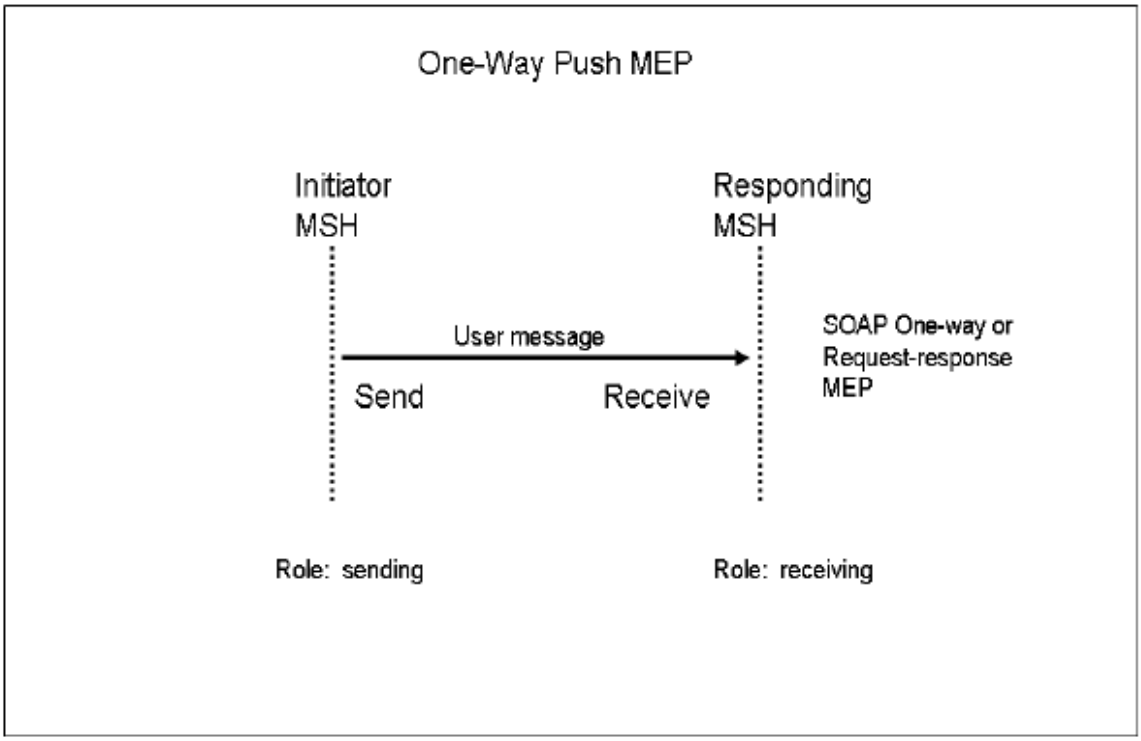

図 2：一方向の Push の MEP

2.2.3.2　一方向の Pull 電文の交換パターン

この MEP には単一の ebXML 電文搬送サービス ユーザー電文が関与する。この MEP では、電文の送信は、SOAP の要求－応答 MEP のインスタンス上で受信 MSH(電文処理機能) によって開始される。SOAP の MEP の最初の行程（要求）は、PullRequest というシグナル電文ユニットを送信する。この MEP の 2 番目の行程は、SOAP 応答として引き出されたユーザー電文ユニットを返信する。この MEP に準拠するためには、ユーザー電文は eb:RefToMessageId を持つ PullRequest 電文ユニットと関連がなければならず、次のユーザー電文によって参照されてはならない。また、PullRequest シグナルにある要素 MessageInfo は、要素 eb:RefToMessageId を包含してはならない。引き出せる電文が何もない場合、7.7.1 項に記載されているように、重大であることを表す「警告」という ebXML 電文搬送サービスのエラーシグナルと“EmptyPipe”というエラーコードを伴う SOAP 応答を返信しなければならない。図 3 は、この MEP に関与する交換パターンと動作を表す。

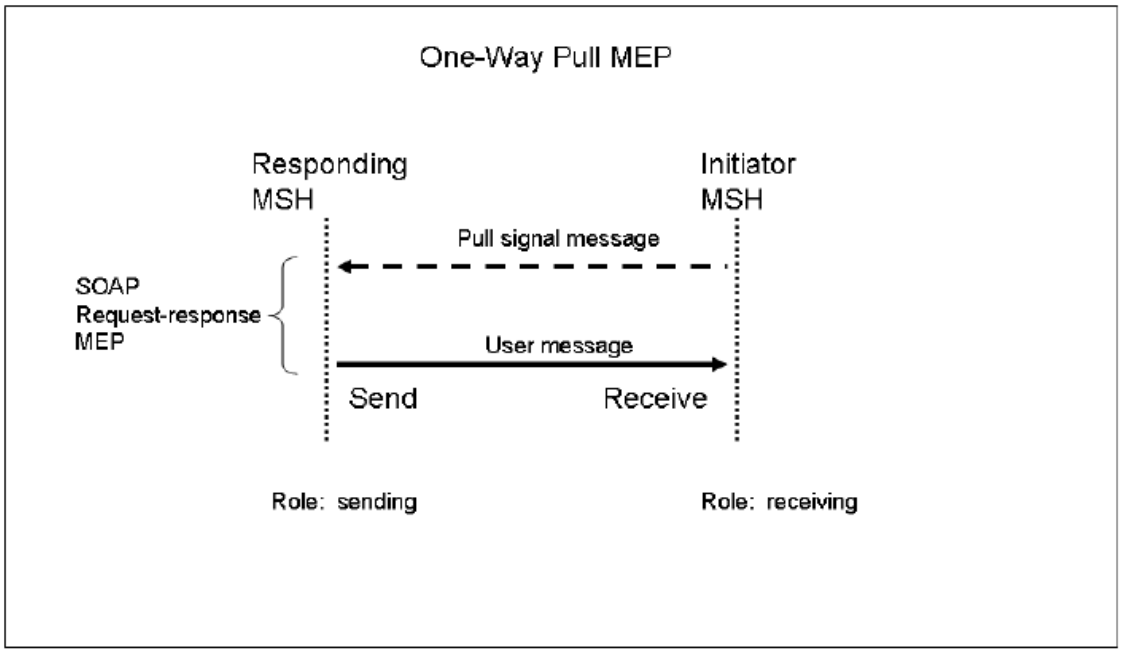

図 3：一方向の Pull の MEP

2.2.3.3　要求－応答電文の交換パターン

この MEP には、単一の SOAP 要求－応答 MEP のインスタンス上で 2 つの ebXML 電文搬送サービス ユーザー電文が関与する。これは送信 MSH(電文処理機能）によって開始される。この MEP の最初の行程では、「ebXML 電文搬送サービス要求」と呼ばれる ebXML 電文搬送サービス ユーザー電文ユニットが SOAP 要求電文上で送信される。この MEP の 2 番目の行程では、「ebXML 電文搬送サービス 応答」と呼ばれる関連のあるユーザー電文ユニットが SOAP 応答として送信される。この MEP に準拠するためには、ebXML 電文搬送サービス 要求は他のユーザー電文（要素 eb:RefToMessageId のない）と関連してはならない。また、ebXML 電文搬送サービス 応答は、5.2.3 項に記述されているように、ヘッダー要素 eb:RefToMessageId を通して ebXML 電文搬送サービス要求を参照しなければならない。図 4 は、この MEP と関連のある交換パターンと動作を表す。

属性 eb:soapresp は、ebXML 電文搬送サービス 要求で「真」に設定されていなければならない。

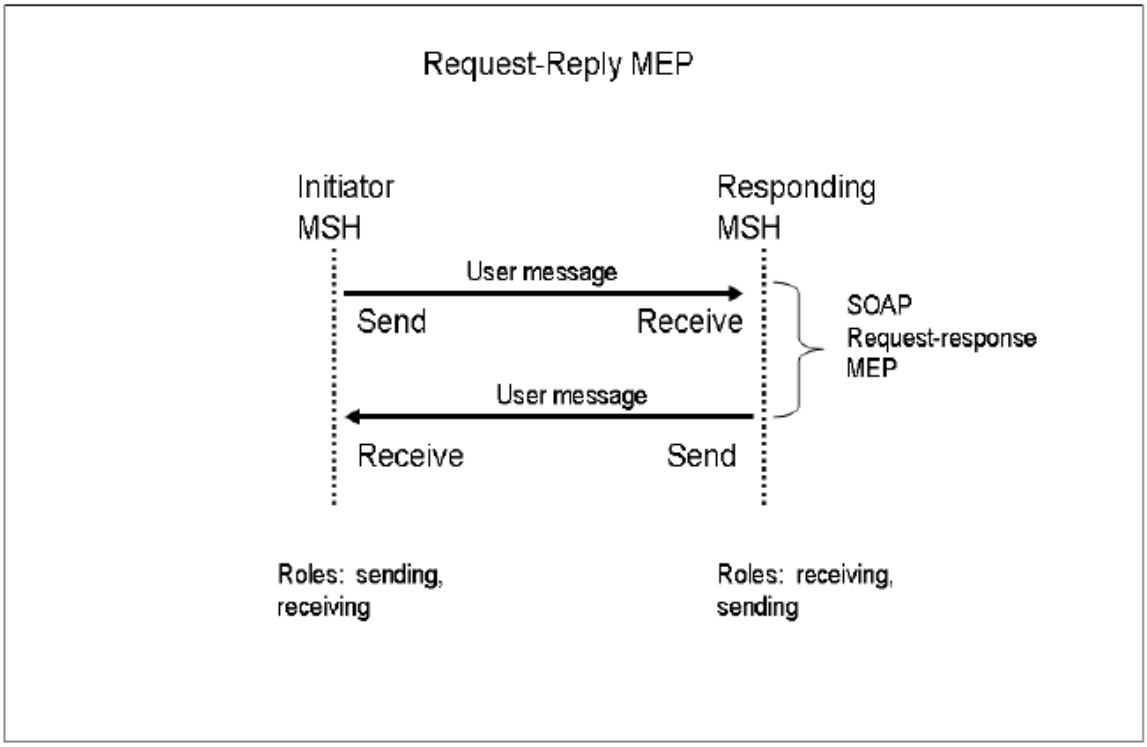

図 4：要求－応答 MEP

2.3　電文搬送パイプ

2.3.1　概念および目的

電文パイプは、送信 MSH(電文処理機能） と受信 MSH(電文処理機能）の間で電文を搬送するための論理チャンネルを決定する。中には、送信 MSH(電文処理機能） と受信 MSH(電文処理機能）のペアによってサポートされるパイプもある。

「デフォルトの」パイプと呼ばれるパイプは、常に、通信を取り合っているペアの MSH(電文処理機能）の間でサポートされていて、その 2 つの MSH(電文処理機能）間の論理チャンネルを簡潔に表している。送信 MSH(電文処理機能） と受信 MSH(電文処理機能）の間で明確に決定されているパイプが他にない場合、すべての電文はデフォルトのパイプを使用する。

パイプは、他のパイプからの電文を個別に搬送することを目的としている。つまり、SOAP 上で電文が搬送される順番と方法は、パイプの適用範囲内でのみ決定される。2 つの異なるパイプに入れられた 2 つの電文が搬送される順番は、MSH(電文処理機能） がその動作のコンテキストまたはアプリケーションの入力に基づいて制御する。これにより、2 つの MSH(電文処理機能）間でいつどのような順番で電文が搬送されるかをより適切に制御できるようになる。

パイプに電文を割り当てるサポート（例えば、提出された電文を自動的にマッピングさせる、あるいは、構成やフィルタリングに基づいてパイプから検索する）については、本仕様の対象外である。本仕様は、パイプの特性項目、および、パイプの使用法が電文プロトコルに与える影響についてのみ説明する。特定のパイプの実装または使用方法については規定しない。

2.3.2　定義および使用要件

パイプは属性の集合として定義されていて、電文をどのようにどこに搬送するかを決定する。待ち行列など、電文の流れの他の側面を制御する特定のデータ構造を利用する上で、パイプの実装は必ずしも必要ない。

パイプは、URI という名前で識別される。パイプには以下の 4 つの属性がある。

- パイプの名前
- パイプの送信 MSH(電文処理機能)　の終点の URL
- パイプの受信 MSH(電文処理機能)　の終点の URL
- パイプの搬送モード（2.4 項参照）

場合によって、例えば、関連のある MSH(電文処理機能）の 1 つが静的なアドレスを持っていない場合は、送信 MSH(電文処理機能)　の終点か受信 MSH(電文処理機能)　の終点のいずれかが（両方ではない）定義されていないことがある。デフォルトのパイプ名は、受信 MSH(電文処理機能）の URL である。パイプ名は、送信 MSH(電文処理機能）のコンテキスト内において（つまり、同一の送信 MSH(電文処理機能）を使用するすべてのパイプの中で）一意でなければならない。

送信 MSH(電文処理機能)　（提出）に提出された電文は、送信される前にパイプと関連がなければならない。この関連が提出時に結び付けられるか提出後に結び付けられるかは、実装次第である。

パイプと関連のある電文は、このパイプ特有の宛て先と搬送モードに従って送信されなければならない。

（注）

- 電文パイプは、その割り当てられている送信 MSH(電文処理機能）から受信 MSH(電文処理機能）に、一方向にのみ電文を搬送できる。
- パイプにある電文は、さまざまな理由（トランスポートの問題、安全性、媒介）により搬送が失敗する場合がある。したがって、いつでもパイプから削除できる。
  - パイプに関するサービスの質に対して特定の要件はない。QoS とパイプを関連付ける実装は自由に行えるが、安全性と信頼性は、常にパイプと直角に当事者または MSH(電文処理機能）と関連を持つ。
- パイプは、ユーザー電文とシグナル電文の両方を伝達するために使用される。パイプの属性として上述された送信の終点と受信の終点は、ユーザー電文に対して定義されている送信 MSH(電文処理機能）と受信 MSH(電文処理機能）の役割にそれぞれ適合し、シグナル電文の場合は、送信元と送信先を簡潔に識別する。
- 信頼性電文搬送を使用する際、信頼性の同意をオーダーする電文が要求される場合は、2 つの MSH(電文処理機能）間で同じ順序で送信される電文はすべて同じパイプを使用しなければならない。電文の順序をサポートするためには、パイプに割り当てられた電文は、送信 MSH(電文処理機能）に提出された時と同じ順序で送信されなければならない。

2.4　電文搬送モード

2.4.1　搬送モードの定義

ebXML 電文搬送サービス 電文には 2 つの搬送モードがある。搬送モードの適用範囲はパイプである。デフォルトのパイプ以外に定義されているパイプがない場合、搬送モードは送信 MSH(電文処理機能）から受信 MSH(電文処理機能）に送られるすべての電文に適用する。搬送モードは以下の 2 つである。

・push 型の搬送モード （Push モード)。このモードでは、送信 MSH(電文処理機能）によってパイプと関連付けられている電文は、送信 MSH(電文処理機能）自体の主導の下、受信 MSH(電文処理機能） に搬送される。このモードはデフォルトの動作モードである。

・ pull 型の搬送モード（Pull モード)。このモードでは、送信 MSH(電文処理機能）によってパイプと関連付けられている電文は、このパイプのターゲットである受信 MSH(電文処理機能）から送られる PullRequest シグナルへの応答として受信 MSH(電文処理機能）に搬送される。

どちらの搬送モードも、両方の終点がプロトコルの固定アドレスと関連していなければならない。例えば、HTTP の Pull モードは、送信 MSH(電文処理機能）のアドレスが既知でなければならない。

複数のパイプがある MSH(電文処理機能)から他の MSH(電文処理機能)に定義されている場合、すべてのパイプが同じ搬送モードで使用されなければならないわけではなく、Push モードで使用されるパイプもあれば、Pull モードで使用されるパイプもある。

2.4.2　搬送モードと MEP

Push モードのパイプは、以下の ebXML 電文搬送サービス の MEP においてユーザー電文の搬送をサポートする。

・ 一方向の Push の MEP.

・ 要求－応答 MEP の最初の行程（要求）

・ 要求－応答 MEP の 2 番目の行程（応答）

注：

ebXML 電文搬送サービスの要求－応答 MEP のインスタンスの要求と応答は、逆方向に向かう異なる 2 つのパイプを使用する。

Pull モードのパイプは、以下の ebXML 電文搬送サービスの MEP においてユーザー電文の搬送をサポートする。

・ 一方向の Pull の MEP の 2 番目の行程（応答）

（注）使用中のパイプの搬送モードは、送信 MSH(電文処理機能）と受信 MSH(電文処理機能）の両方がそれらの振る舞いを調整することを暗示する。これらの MSH(電文処理機能）は、与えられたパイプに対してどちらかのモードで動作すると言える。両方の MSH(電文処理機能） がそれらの動作モードをどのように調整するかについては本仕様の対象外である。

2.4.3　pull 型の搬送モード

pull 型の搬送モード（Pull モード）は、一方向の Pull の MEP で使用される。これにより、受信 MSH(電文処理機能）は、前に送信 MSH(電文処理機能）に提出されたがまだ送信されていない電文の搬送を開始できる。これは実装によって決まるものだが、通常、送信 MSH(電文処理機能）における待ち行列の能力によって達成される。Pull モードは、以下の状況をより適切に処理することを目的としている。

・ 受信 MSH(電文処理機能）の接続制限。重要な期間に利用できなかった場合や、ファイアウォール制限または静的な IP アドレスの欠如によってアクセスが制限された場合などを含む。

・ 受信 MSH(電文処理機能）は、電文を受信するタイミングおよび一度にいくつの電文を受信するかを制御することを望む。また、最初にどの種類の電文を受信するかを制御することも望む。

Pull モードを使用して安全に搬送するためには（安全性の項を参照）、送信 MSH(電文処理機能）と受信 MSH(電文処理機能）の間で Pull モードで使用されるパイプは、電文交換前に既知でなければならない。これは、電文パイプの動作コンテキストを共有することによって行われる（3 項の OpCtx_MessagePipes 参照)。例えば、パイプ名（URI）と属性のリストは、CPA などの合意に規定されている。PullRequest シグナルは、デフォルトのパイプの場合でも、引き出し元のパイプを特定しなければならない。また、PullRequest シグナルを Pull モードのパイプで伝達してはならない。

3　動作の概念

本節では、本仕様の実装を展開する際に考慮すべきいくつかの動作の側面について説明する。特に、配置と統合の側面に関する手引きのみならず、実装が動作して本仕様の残りの部分に論理的根拠を与えるような環境のさまざまな側面についても説明する。結果として、その目的はこれらの概念を規範的に説明することではなく、ユーザーの立場から概念を考えることである（概して、ユーザーはエンドユーザーだと考えられるが、開発者やシステムインテグレーターも含む)。

3.1　動作モード

MSH(電文処理機能）は、送信されたか受信されたかにかかわらず、電文およびそれらのヘッダーの内容を処理するように促すコンテキスト上の情報の知識の中で動作している。このような情報は MSH(電文処理機能）の動作モードを定義する。これは、交信している当事者間の同意、あるいは、そのような同意の中で電文搬送層と関連のある部分として解釈される。そのような同意の参照は、電文のヘッダーに存在する（5.2.9 節の「eb:AgreementRef 要素」を参照)。

動作モードは MSH(電文処理機能）の入力データを包含する。これは通常は電文毎には提供されないが、2 つ以上の当事者間で交換される電文の集合には共通である。この点で、これは MSH(電文処理機能） 実装の構成データとして解釈される。

動作モードのデータに関する説明および用途は本仕様の対象外だが、今後さらに具体的な説明と結び付けやすいようにこのコンテキストの主な要素を抽象的に参照する上で役立つ。動作モードは、動作コンテキストと呼ばれる 6 つの主なエンティティに分けられる。

これらは、OpCtx-<entityname>という形式の名前で識別される。

- OpCtx-Transport: トランスポートレベルの相互運用性を達成するために必要な、トランスポート上関連のあるすべての情報を包含する。このコンテキストデータは、2 つの MSH(電文処理機能）間で関与するトランスポート型（例えば、HTTP, SMTP, FTP)、および、関連する構成パラメータを決定する。アプリケーションの適用範囲は、通常 2 つの MSH(電文処理機能）の間である。
- OpCtx-Reliability: 交換される電文の信頼性を左右する、すべての信頼性の協定、または、それらの参照を包含する。これには、プロトコルおよびタイミングパラメータなども含まれる。アプリケーションの適用範囲は、通常 2 人の当事者間である。このコンテキストデータは、信頼性のヘッダーの内容を決定する。
- OpCtx-Security: 電文交換を左右する、すべての安全性の協定、または、それらの参照を包含する。これには、安全性のコンテキストおよび関連するリソース（証明書、SAML アサーションなど）が含まれる。アプリケーションの適用範囲は、MSH(電文処理機能）レベルの安全性のアドレスを取る場合もあるが（例えば、シグナル電文に対して)、通常は 2 人の当事者間である。このコンテキストデータは、wsse:Security のヘッダーの内容を決定する。
- OpCtx-BusinessCollaboration: 2 者の当事者間のコラボレーションと関連するすべてのデータ、および、それらのバックエンドシステムとの結合を包含する。これは以下のヘッダーの内容を駆動する。
- eb:UserMessage/eb:PartyInfo

・eb:UserMessage/eb:CollaborationInfo
・eb:UserMessage/eb:MessageProperties
特に、これらの当事者がどの MEP、あるいは、どの電文パイプを使用するかはここで特定される。
・OpCtx-MessagePipes: 他の MSH(電文処理機能) と電文を交換する際に、MSH(電文処理機能) が演じる役割における電文パイプおよび関連する属性すべての識別を包含する。このコンテキストデータは、属性が取り得る値を決定する。
・eb:UserMessage/eb:MessageInfo/eb:PipeId
これは、PullRequest シグナルの使用も制御する。
・ OpCtx-ErrorHandling: 各 ebXML 電文搬送サービスエラーに対して選択される報告アクション、および関連するパラメータ（例えば、ebXML 電文搬送サービス シグナル電文を使用している場合、送信先の MSH(電文処理機能）のアドレスを決定する。エラー報告にどの電文パイプを使用するかも決定する）を包含する。また、包含されている機能的モジュール（信頼性、安全性）によってもたらされたエラーから生じている ebXML 電文搬送サービスエラーに対して選択された方針も包含する。

3.2 デフォルトの動作モード

OpCtx-Transport を共有する場合を除き、2 つの MSH(電文処理機能) は、交換に先立って（帯域外の）動作コンテキストの情報を共有しなくても交信できなければならない。

送信 MSH(電文処理機能）と受信の MSH(電文処理機能）のペアが配置され、以下の条件が満たされている場合、このペアの MSH(電文処理機能）はいかなる同意を参照しなくても電文を交換できなければならない（つまり、eb:AgreementRef 要素は、ヘッダーにある eb:MessageInfo に存在しない)。eb:AgreementRef が存在しない場合、交換にはデフォルトの動作モード（または DO モード）が使用されなければならない。

(1) 安全ではない電文（つまり、wsse:Security ヘッダーが付いていない電文）の交換を防ぐ安全性の構成は存在しない。
(2) 使用される搬送プロトコルに関して事前の同意および構成の詳細が存在する。
(3) これらの MSH(電文処理機能）が、pull 型の搬送モード（電文搬送モデルの項に記載されている定義を参照）で排他的には使用されない。

2 つの MSH(電文処理機能) 間のデフォルトの動作モードは、以下の動作コンテキストを特徴とする。

・OpCtx-Reliability: empty: 信頼性が確保されない電文搬送と想定される（信頼性ヘッダーが存在しない)。
・OpCtx-Security: empty: 安全性が確保されない電文搬送と想定される（安全性ヘッダーが存在しない)。
・OpCtx-MessagePipes: empty: デフォルトでデフォルトのパイプだけが使用され、そのデフォルトの搬送モードは push モードである。つまり、ebXML 電文搬送サービス の One-Way Push

（一方向の push）MEP、または Request-Reply（要求－応答）MEP の最初の行程を伝えることができる。

(注)DO モードとは、アプリケーション間ではなく、MSH(電文処理機能）間の相互運用性のことである。送信者 MSH(電文処理機能）と受信者 MSH(電文処理機能）のそれぞれに関連している 2 つのアプリケーションが DO モードで電文を交換できるようにするためには、OpCtx-BusinessCollaboration について追加の同意を得る必要がある。そうすれば、これらの電文はアプリケーションまたは業務プロセスと適切に結合できる。

3.3　電文搬送サービス処理モデル

ebXML 電文搬送サービスは、概念上、以下の 3 つの部分に分けられる。

1. 抽象的なサービスインタフェース
2. MSH(電文処理機能）によって提供される機能
3. 基礎的な搬送サービスへのマッピング

図 5 は、ebXML 電文搬送サービスアーキテクチャとなり得る実装に存在する機能的なモジュールの論理的な配列を表す。これらのモジュールは、それらの内部関係および依存性を表す方法で配列される。

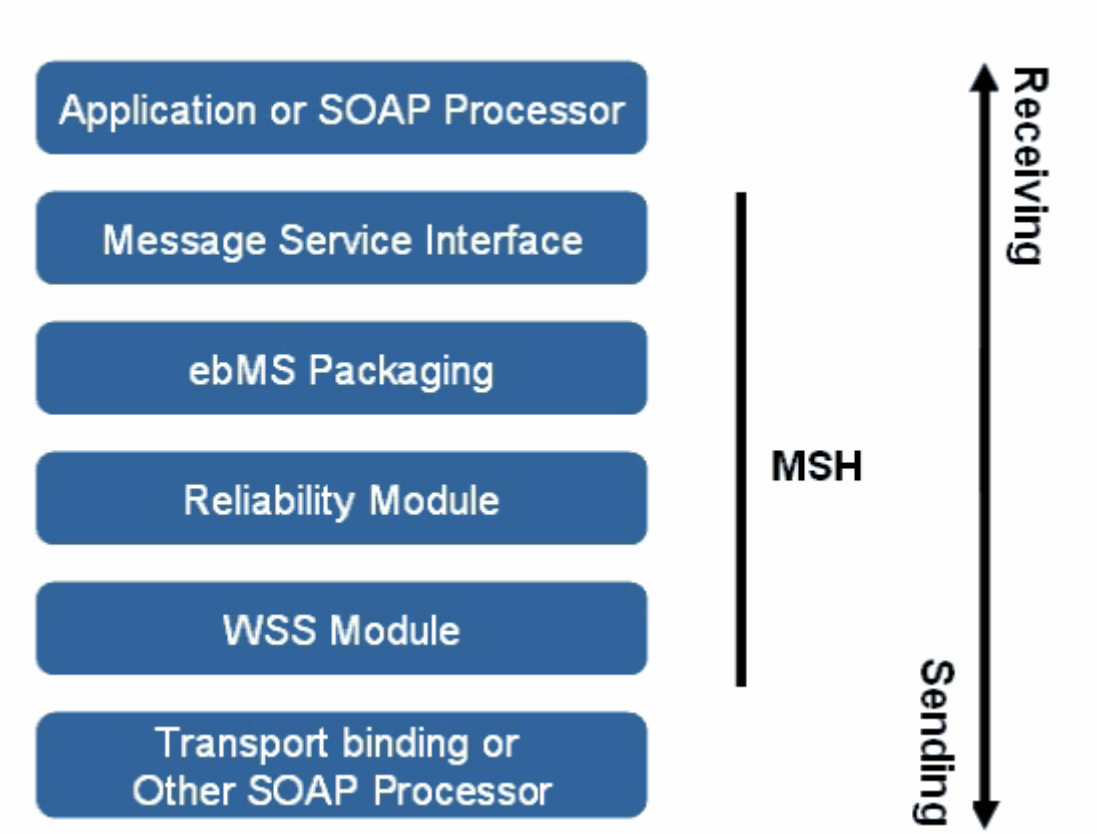

図 5: 構成要素の関係

以下は、上に図示されている各モジュールの説明である。上のスタック図は抽象的で、本仕様は、この図が提案しているアーキテクチャを実装に採用するように強要しているわけではない。

・ アプリケーションまたは SOAP プロセッサーこれは、電文交換の業務ロジックまたは業務プロセスが存在する場所である。

・ 電文搬送サービスインタフェースーこれは、MSH(電文処理機能）のコアと ebXML アプリケーションの間のインタフェースで、電文はここを通過する。

・ebXML 電文搬送サービスパッケージングー 処理、包囲、ペイロードサービスの実行は、このモジュールによって行われる。

・ 信頼性電文処理ーこのモジュールは、電文に対するサービスの質の要件を満たす。

・Web サービスの安全性の処理ーあらゆる SOAP 電文の内容の複合化、および、あらゆるデジタル署名の検証はこのモジュール内で生じる。

・ 搬送結合ーこれらは、実際の搬送結合である。本仕様は、HTTP（11.2 項）と SMTP（11.3 項）の結合を説明し、他のプロトコルの追加をサポートする。

3.4 サポートされた接続形態の例

ebXML 電文搬送サービスの MSH(電文処理機能）は、以下の電文搬送の接続形態で使用される。それぞれの接続形態において、MSH(電文処理機能）は、以下の異なるヘッダーを理解する単一の SOAP ノードまたは一連の SOAP ノードのいずれかとして実装される。(a) ebXML 電文搬送サービスヘッダーを理解するノード(b) 信頼性ヘッダーを理解するノード(c) 安全性ヘッダーを理解するノード。

・ 電文搬送の終点としての MSH(電文処理機能)。このコンテキストでは、送信 MSH(電文処理機能）は SOAP 電文の生成者である。受信 MSH(電文処理機能）は、最終的な SOAP 受信者である。このような場合、配信動作は、アプリケーションレベルの利用者（例えば、業務アプリケーション）への電文配信または待ち行列システムから構成される。提出動作は、ebXML 電文搬送サービス電文を生成するために必要とされる、MSH(電文処理機能）へのデータとペイロードの受け渡しから構成される。

・SOAP 媒介としての SOAPMSH(電文処理機能)。このコンテキストでは、送信 MSH(電文処理機能）は SOAP 媒介として行動し、（提出動作を通して）受信した SOAP 電文にヘッダーを追加しているだけである。このコンテキストで MSH(電文処理機能）を使用するためには、入力の SOAP 電文に安全性ヘッダーが存在しない方がよい。少なくとも、分離した XML 署名以外の署名がないようにすれば、ヘッダーの追加が署名を破壊することはない。SOAP 媒介として行動している受信 MSH(電文処理機能）は、ヘッダーの一部（ebXML 電文搬送サービスヘッダー、信頼性ヘッダー、および一部の安全性ヘッダーを含む）を処理および削除し、SOAP 電文（もはや ebXML 電文搬送サービス電文ではない）を次の SOAP ノードに搬送する。

・MSH(電文処理機能）マルチホップチェーンにある媒介としての MSH(電文処理機能)。このコンテキストでは、MSH(電文処理機能）媒介として行動している送信 MSH(電文処理機能）は、（受信動作を通して）受信した ebXML 電文搬送サービス電文を他の MSH(電文処理機能）に（送信動作を通して）搬送している。MSH(電文処理機能）媒介の振る舞いについてはマルチ

ホップモジュールを参照のこと。

4　Pull 型電文搬送サービス

4.1　Pull 型電文搬送サービスの目的

取引当事者ごとに、電文の流れ、断続的な接続、静的な IP アドレスの欠如、またはファイアウォール制限を処理する能力には差がある。さらに、電文が搬送され、その受領通知を受け取ることに成功すると、それを管理する責任が移動する。このことから、受信者は、(a)搬送を開始することによって基礎的なプロトコルの搬送手順を制御し続けること、および、(b)最初にどの電文をいつ受信するかを決定することを望む。2 つの機能は、以下をサポートする ebXML 電文搬送サービス V3 に導入されている。

・pull 型の搬送モード
・ 電文パイプ

pull 型の搬送モードあるいは Pull モードは、一方向の Pull ebXML 電文搬送サービス MEP によって抽象的に定義されている（「電文搬送モデル」にあるシンプルな MEP の 2.3 項参照）。この MEP により、MSH(電文処理機能) は受信者として電文の搬送を開始できる。一方向の push ebXML 電文搬送サービス MEP との組み合わせで使用される場合、MSH(電文処理機能）は、リモートの MSH(電文処理機能）とのクライアント・サーバ型の相互作用に従事しながら、入ってくる要求に対してポートを開くことなく、完全に非同期の搬送を制御し、一方から両方に開始することができる。

例：モバイルなど、静的な IP アドレスがなく（常時接続でない）、記憶容量が限られている機器は、要求や受信電文を開始し、その応答と同期することしかできない。一方向の Pull MEP により、このデバイスは、遠隔の当事者がこれらの電文送信の接続を頻繁に確認する必要を減らす一方で、受信した電文の流れを制御し、それをそれ自身のリソースに適合させることができる。

電文パイプ（2.4 項の定義を参照）は、搬送モード（Push または Pull）および搬送順序の観点から、MSH(電文処理機能) から別の MSH(電文処理機能) への電文の流れを別々の流れに分け、いずれかの MSH(電文処理機能）がそれぞれの流れを単独で制御できるようにする。

例：ペアの取引当事者（大規模なバイヤーと小規模なサプライヤー）は、バイヤーによって送信された電文を搬送するために 2 つの電文パイプを生成することにとした。1 つのパイプは、迅速な処理が求められる緊急の電文（最優先の受注オーダー、および、すでに受領した受注オーダーの更新）に割り当てられ、もう 1 つのパイプは、緊急度の高くない電文（支払い、カタログ請求、確認、受領通知など）に割り当てられている。バイヤーは、電文内では示されない緊急レベルを決定する。バイヤーとの合意毎に、サプライヤーは、最初に緊急パイプから届いたすべての電文を引き出して処理し、その後、緊急ではないパイプから届いた電文を処理する。これにより、低容量の受信 MSH(電文処理機能）（サプライヤー）は、そのリソースを最も緊急度の高い電文に投入し、緊急ではないが重要かつ短期間では処理できない電文を管理（永続性、復旧、安全性）

する経費とリスクを回避しつつ、受信した電文に優先順位を付けられるようになる。

ヘッダーデータのフィルター条件を確認する必要があるような、さらに複雑なフィルターメカニズムについては、本仕様の対象外である。これは、パイプを補うものとして送信 MSH(電文処理機能）および/または受信 MSH(電文処理機能）に実装される場合がある。パイプの概念はシンプルで頑強なソリューションで、相互運用性のリスクは低い。これによると、フィルター表現を搬送しなくても、どの電文型がどのパイプを使用するかについての生成者と利用者間の前の同意に基づいて搬送を制御できる。

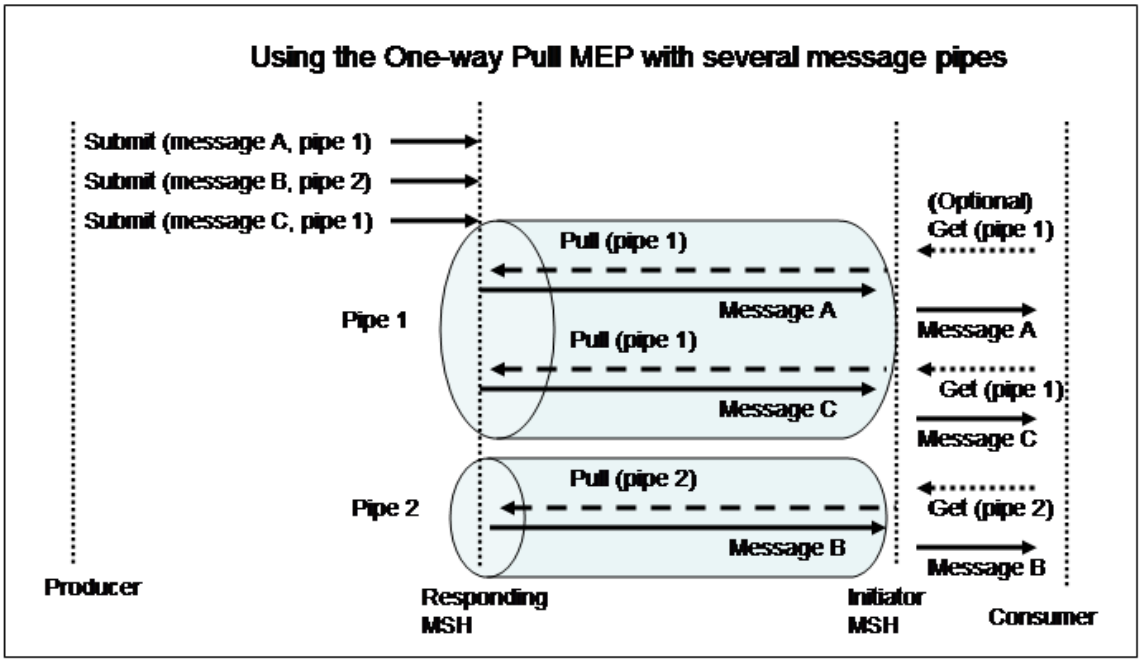

図 6：電文パイプをともなう一方向の Pull

利用者がどのようにパイプと一方向の Pull MEP を使用すれば、利用者が受信および処理を望む電文の順番を制御できるかを図 6 に示す。パイプ 1 は、利用者側によって優先して「pull される」。

この図に示されている例を変形させると、利用者側は、生成者の指導の下でクリティカルな電文を受信する一方で、接続制限を受けず、クリティカルではない電文の搬送だけを制御しようとする。パイプ 1 は、Push モード用に構成され、パイプ 2 は Pull モード内で構成されている。利用者は、これらを処理する準備が整っている場合のみ、クリティカルではない電文の搬送をパイプ 2 で開始する。

4.2 Pull モードのサポート

Pull 型の搬送モードを使用するには、MSH(電文処理機能） が一方向の Pull の MEP をサポ

ートする必要がある。この MEP を開始する PullRequest シグナルについては、4.3 項（シグナルパッケージング）で定義している。Pull モードは、 常に電文パイプの適用範囲である。

これは、送信 MSH(電文処理機能）と受信 MSH(電文処理機能）の間で常に少なくとも 1 つの電文パイプ（デフォルトのパイプ）が開かれていて、複数の電文パイプをサポートできるか否かににかかわらず Pull モードがサポートされるからである。

PullRequest シグナルを送信する際は、pull された元のパイプの名前を特定しなければならない（属性@fromPipe)。

典型的かつ適切な一方向の Pull の MEP のインスタンスに対して pull 型の電文を処理するモデルは以下である。

応答者 MSH(電文処理機能） の観点：

1. 提出：開始者側にいる利用者向けに、生産者が MSH(電文処理機能）に電文データを提出すること。電文はパイプと関連している。提出者がパイプ名を付けない場合、または、この関連自体が取るべき方法で MSH(電文処理機能）の実装が行われていない場合は、デフォルトのパイプが使用される。この電文と関連のあるパイプは、Pull モード用（OpCtx-MessagePipes の表現に特定されている）に構成されていなければならない。

開始者 MSH(電文処理機能）の観点：

2. MSH(電文処理機能） による PullRequest シグナルの送信。PullRequest シグナルは、pull された元のパイプを特定する。

応答者 MSH(電文処理機能） の観点：

3. PullRequest シグナルの受信。受信するすべての PullRequest シグナルに対して、応答者 MSH(電文処理機能）（送信の役割を演じている）は先に提出された電文を選択する。提出動作に対し、FIFO（先入れ先出し）方針に従って電文を選択するのが望ましい。送信用に特定されたパイプにユーザー電文がない場合は、代わりに"EmptyPipe"という簡単な説明を伴う警告シグナルを返信しなければならない。
4. 送信： 選択された電文は SOAP 応答を通して PullRequest に送信される。

開始者 MSH(電文処理機能）の観点：

5. 受信：pull 型の電文は MSH(電文処理機能）が処理に使用できる。ヘッダーの要素 eb:MessageInfo/eb:PipeId は、それがどのパイプから pull されたかを示し、PullRequest シグナルにある@fromPipe 値と同じである。
6. 配信: ebXML 電文搬送サービス のヘッダーを処理した後に、pull 型の電文データを MSH(電文処理機能）の利用者に配信すること。

例： PullRequest シグナルに対する eb:Message ヘッダーの例

```
<SOAP:Envelope>
<SOAP:Header>
<eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >
       <eb:SignalMessage>
              <eb:MessageInfo>
                     <eb:TimeStamp>2005-10-01T10:01:00</eb:TimeStamp>
                     <eb:MessageId>UUID-4@example.com</eb:MessageId>
              </eb:MessageInfo>
              <eb:PullRequest fromPipe="pipe://mymsh.com/pipe123"/>
       </eb:SignalMessage>
</eb:Message>
</SOAP:Header>
<SOAP:Body/>
</SOAP:Envelope>
```

例： 上記の PullReques シグナルの例に応答するための eb:Message ヘッダーの概略

```
<SOAP:Envelope>
<SOAP:Header>
<eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >
       <eb:UserMessage>
              <eb:MessageInfo>
                     <eb:TimeStamp>2005-10-01T10:02:00</eb:TimeStamp>
                     <eb:MessageId>UUID-5@example.com</eb:MessageId>
                     <eb:RefToMessageId>UUID-4@example.com</eb:RefToMessageId>
                     <eb:PipeId>pipe://mymsh.com/pipe123</eb:PipeId>
              </eb:MessageInfo>
              <eb:PartyInfo">
                     ...
              </eb:PartyInfo">
              <eb:CollaborationInfo">
                     ...
              </eb:CollaborationInfo">
              <eb:PayloadInfo">
                     ...
              </eb:PayloadInfo">
       </eb:UserMessage>
</eb:Message>
</SOAP:Header>
<SOAP:Body>
...
</SOAP:Body>

</SOAP:Envelope>
```

4.3 Pull 型電文搬送サービスの安全性と信頼性

一般的に、pull 型の電文の信頼性は、対応する PullRequest シグナルの信頼性と関連がある。これが、一方向の Pull の MEP の完全なインスタンスの信頼性が 8.3 項で扱われている理由である。

PullRequest シグナルの安全性については、5.6 項で詳しく説明している。

例： PullRequest シグナルに対する安全性、信頼性 eb:Message ヘッダーの概略。この例で使用

されている信頼性のヘッダーは、WS-Reliability の使用を前提とし、MEP の応答電文に受領確認を送信するとともに、最低 1 回の配信を特定している。

```
<SOAP:Envelope>
<SOAP:Header>
<eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >
       <eb:SignalMessage>
              <eb:MessageInfo>
                     <eb:TimeStamp>2005-10-01T10:01:00</eb:TimeStamp>
                     <eb:MessageId>UUID-4@example.com</eb:MessageId>
              </eb:MessageInfo>
              <eb:PullRequest fromPipe="pipe://mymsh.com/pipe123"/>
       </eb:SignalMessage>
</eb:Message>
<wss:Security>
       ...
</wss:Security>
<wsr:Request SOAP:mustUnderstand="1">
       ...
       <ReplyPattern>
              <Value>Response</Value>
       </ReplyPattern>
       <AckRequested/>
       ...
</wsr:Request>
</SOAP:Header>
<SOAP:Body/>
</SOAP:Envelope>
```

例：上記の PullRequest シグナルへの応答に対する、安全性、信頼性 eb:Message ヘッダーの概略

```
<SOAP:Envelope>
<SOAP:Header>
<eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >
       <eb:UserMessage>
              <eb:MessageInfo>
                     <eb:TimeStamp>2005-10-01T10:02:00</eb:TimeStamp>
                     <eb:MessageId>UUID-5@example.com</eb:MessageId>
                     <eb:RefToMessageId>UUID-
4@example.com</eb:RefToMessageId>
                     <eb:PipeId>pipe://mymsh.com/pipe123</eb:PipeId>
              </eb:MessageInfo>
              <eb:PartyInfo">
              ...
              </eb:PartyInfo">
              <eb:CollaborationInfo">
              ...
              </eb:CollaborationInfo">
              <eb:PayloadInfo">
              ...
              </eb:PayloadInfo">
       </eb:UserMessage>
</eb:Message>
<wsr:Response SOAP:mustUnderstand="1">
       ...
</wsr:Response>
<wss:Security>
```

```
      ...
</wss:Security>
</SOAP:Header>
<SOAP:Body>
...
</SOAP:Body>
</SOAP:Envelope>
```

（注）上記の例で、WS-Reliability の使用を仮定している安全性のヘッダーは、応答という要素である。これは、PullRequest シグナルに対する安全性の受領確認を包含する。この例には、wsr:Request という安全性のヘッダーは存在しない。つまり、この応答には受領確認は予期されない。これは、この ebXML 電文搬送サービスのユーザー電文に対して、最低 1 回の信頼性の同意がサポートされることを妨げない。つまり、ユーザー電文が開始 MSH(電文処理機能）によって受信されなかった場合、受信確認の欠如は PullRequest シグナルの再送を生じさせる。これは、順に同じユーザー電文の再送を引き起こす。配信に失敗した場合、開始 MSH(電文処理機能）側にエラーが通知される。

5　電文パッケージング

5.1　電文エンベロープ

5.1.1　MIME 構造および SOAP プロファイル

ebXML 電文搬送サービスの SOAP ヘッダーeb:Message において、接頭辞 “eb” は、ebXML 電文搬送サービス 3.0 の名前空間に対応する接頭辞の例である。ebXML 電文搬送サービス電文は、簡潔な [SOAP11] 電文としてパッケージ化されるか、あるいは、ペイロードまたは添付ファイルを包含できるように MIME のマルチパート内でパッケージ化される。どちらのパッケージングを使用してもよいので、実装はマルチパートではない電文をサポートしなければならない。

ebXML 電文搬送サービス電文は、ヘッダーブロック eb:Message 以外の SOAP 拡張要素を含む場合がある。例えば、電文の信頼性と安全性をサポートするヘッダーブロックは、これらの機能に対する配置要件を満たすために MSH(電文処理機能）によって生成および処理される場合がある。

ebXML 電文搬送サービス電文は、通信プロトコルから独立した [SOAP11]（最新版では SOAP1.1/SOAP1.2）電文としてパッケージ化される。MIME マルチパートの電文エンベロープとして表現される場合、このエンベロープは、電文パッケージとして参照される、添付ファイルつき SOAP 電文 [SOAPATTACH] W3C 注釈に従って構造化されなければならない。

電文パッケージ内には以下の 2 つの論理セクションがある。

- 1 番目のセクションは、それ自体が SOAP ヘッダーに含まれている ebXML 電文搬送サービスヘッダー（つまり、SOAP のヘッダーブロック eb:Message）である。
- 2 番目のセクションは、それ自体が 2 つのセクション（(a) SOAP エンベロープ内にある SOAP のボディ要素、および、MIME パッケージングの場合、(b)アプリケーションレベルの追加のペイロードを包含している、ゼロ以上の追加の MIME 部分）から構成されている ebXML 電文搬送サービスのペイロードである。SOAP ボディと MIME の部分も、ebXML 電文搬送サービスのペイロードコンテナとして参照される。SOAP ボディは、XML の内容を必要とする唯一のペイロードコンテナである。

ebXML 電文搬送サービスユーザー電文の一般的な構造と構成については図 7、シグナル電文については図 8 に示す。

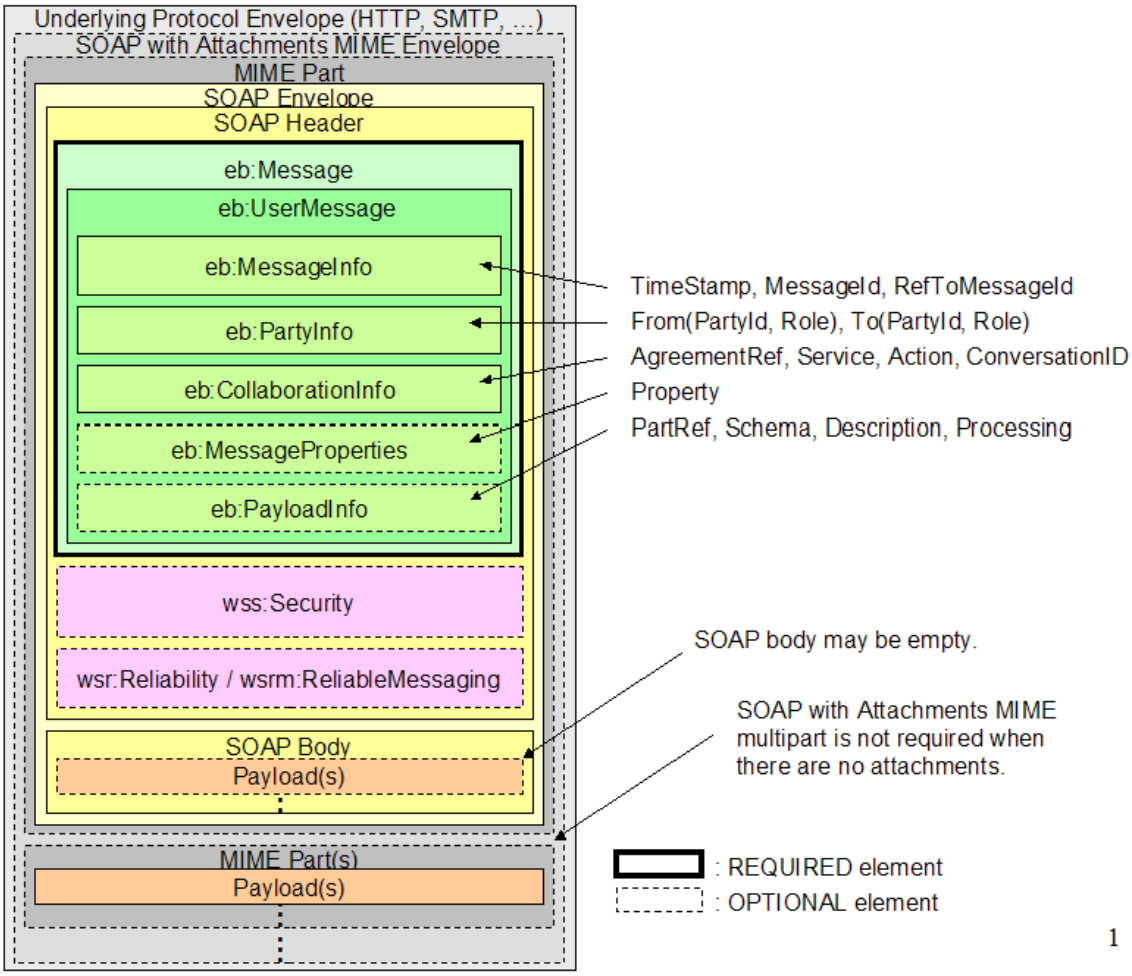

図 7：ユーザー電文の構造

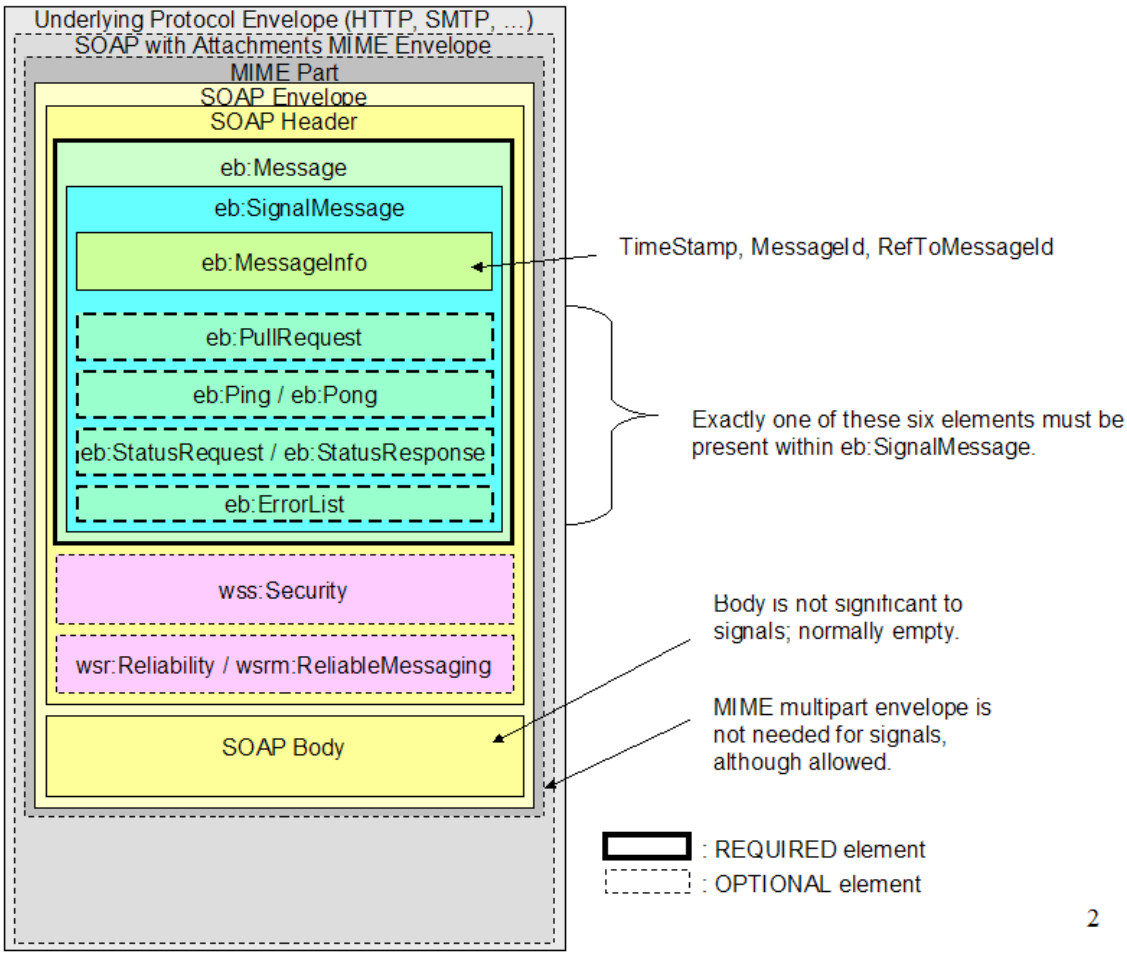

図 8：シグナル電文の構造

SOAP のヘッダーブロック eb:Message の処理は、SOAP の処理意味に従って行われる。 つまり、MSH(電文処理機能）は SOAP プロセッサまたは SOAP ノードとして振る舞い、このヘッダーブロックを理解しなければならない。他のヘッダーブロック（ebXML 電文搬送サービス電文の信頼性および安全性と関連のあるもの以外）は、ebXML 処理の影響を受けない。その結果、送信 MSH(電文処理機能）の実装の中には、ヘッダーeb:Message を追加するだけで、入力として適切に形成された SOAP 電文から ebXML 電文搬送サービス電文を生成するものもある。また、受信 MSH(電文処理機能）の実装の中には、ヘッダーeb:Message を削除するだけで、出力として適切に形成された SOAP 電文を配信できるものもある。

電文パッケージのすべての MIME ヘッダー要素は、添付ファイルつき SOAP 電文 [SOAPATTACH] W3C 注釈と一致しなければならない。さらに、 電文パッケージにある Content-Type MIME ヘッダーは、SOAP 電文の内容を包含している MIME のボディ部分の MIME メディア型と一致する型属性を含む。[SOAP11] の仕様に従い、SOAP 電文の MIME メディア型は、"text/xml"という値を取る。最初のヘッダーに、MIME [RFC2045]に従って構造化されたコンテンツ ID の MIME ヘッダーを含めること、および、Multipart/Related 型に必要と

されるパラメータに加えて、開始パラメータ（MIME の Multipart/Related [RFC2387]においては任意）が常に存在することが望まれる。これにより、より頑強にエラーを検出できるようになる。以下の事例は、Multipart/Related 電文パッケージに対する MIME ヘッダーの例である。

例　Multipart/Related 電文パッケージに対する MIME ヘッダーフラグメント

```
Content-Type: multipart/related; type="text/xml";
boundary="boundaryValue";start="<messagepackage-123@example.com>"
```

```
--boundaryValue
Content-ID: messagepackage-123@example.com
```

実装はマルチパートではない電文をサポートしなければならないので、ペイロードのない ebXML 電文搬送サービス電文は、簡潔な SOAP 電文か、あるいは、ボディ部分を 1 つしか持たないマルチパート電文[SOAPATTACH]として送信される場合がある。

5.1.2　MIME の条件

5.1.2.1　追加の MIME パラメータ

本仕様に記載されているすべての MIME 部分は、MIME [RFC2045]の仕様に適合する追加の MIME ヘッダーを含む場合がある。実装は、本仕様に定義されていない MIME ヘッダーについては考慮しない場合がある。実装は、認識できない MIME ヘッダーを無視しなければならない。例えば、実装は電文に内容の長さを含むことができる。しかし、内容の長さを含む電文の受信者は、それを無視することができる。

5.1.2.2　MIME エラーの報告

電文パッケージで MIME エラーが検出された場合、添付書類を伴う SOAP[SOAPATTACH]に規定されているように、そのエラーを報告しなければならない。

5.1.2.3　XML プロローグ

SOAP 電文の XML プロローグが存在する場合、それは XML 宣言を包含する場合がある（MAY）。本仕様は、XML プロローグに登場している追加のコメントや処理指示については定義していない。例えば、以下のとおりである。

```
Content-Type: text/xml; charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
```

5.1.2.4　XML 宣言

XML 宣言は、SOAP 電文に存在する場合がある。存在する場合、XML 宣言は XML 推奨[XML10]によって必要とされているバージョンの使用を包含しなければならない。またエンコーディング宣言を包含する場合がある。以下に説明されている意味は、適切な ebXML 電文搬送サ

ービスによって実装されなければならない。

5.1.2.5　エンコーディング宣言

エンコーディング宣言と MIME ルーツ部分の文字集合の両方が存在する場合、SOAP 電文に対する XML プロローグは、ルート部分の MIME 内容型の文字集合の属性と等しいエンコーディング宣言を包含する（5.1.4 項参照）。その場合、エンコーディング宣言は、SOAP 電文を生成する際に使用されるエンコーディングと矛盾する値を包含してはならない。SOAP 電文をエンコーディングする際には、UTF-8 を使用することが望ましい。XML [XML10]の 4.3.3 項に規定されている規則を使用して XML プロセッサによって文字のエンコーディングが実行されない場合は、XML 宣言およびそれに包含されているエンコーディング宣言が ebXML の SOAP ヘッダー文に提供される。

（注）　XML v1.0 の仕様[XML10]に従い、XML 文にエンコーディング宣言は必要ない。

5.1.3　ebXML の SOAP エンベロープの拡張

[XML10] の仕様に従い、すべての拡張要素の内容は名前空間で修飾される。ebXML の SOAP 拡張に対する名前空間宣言（擬似属性 xmlns）は、SOAP エンベロープまたはヘッダー要素に包含されたり、あるいは、直接 ebXML の SOAP 拡張要素に包含されたりする場合がある。

5.1.3.1　名前空間の擬似属性

ebXML の SOAP エンベロープ拡張に対する名前空間宣言（擬似属性 xmlns）は（[XMLNS] 参照）、以下の必須の（REQIRED）値を取る。

```
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-3_0.xsd
```

5.1.3.2　属性 xsi:schemaLocation

以下の SOAP の名前空間は W3C XML スキーマの仕様の名前を解決する。

```
http://schemas.xmlsoap.org/soap/envelope/
```

検証を行うパーサーに文を検証するために使用されるスキーマ文の場所を示すために、ebXML の MSH(電文処理機能）の実装に、SOAP エンベロープの要素にある属性 schemaLocation に適合する XMLSchema インスタンスの名前空間を含めることが強く推奨される。

属性 schemaLocation を含めることができなかった場合、受信した電文の XML スキーマ検証を行うことができないかもしれない。

例：

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
http://schemas.xmlsoap.org/soap/envelope/">
```

さらに、ebXML の SOAP ヘッダーの拡張要素の内容は、SOAP の拡張要素定義で修飾された

ebXML の名前空間を包含するスキーマ文を見つける場所の検証を行うパーサーが識別できるように、同じように修飾される場合がある。ebXML の SOAP 拡張要素スキーマ（9 項で説明）は、XML スキーマ仕様[XMLSCHEMA]の W3C 推奨バージョンを使用して定義されている。属性 schemaLocation で修飾された XMLSchema インスタンスの名前空間は、ebXML の SOAP エンベロープ拡張名前空間を宣言する同じ要素にあるそのスキーマ文に対して、ebXML の SOAP エンベロープの拡張名前空間のマッピングを包含すべきである。
先の 5.1.3.1 項で説明されている名前空間の schemaLocation は以下のとおりである。

```
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-3_0.xsd
```

属性 schemaLocation は別々にすることが望ましい。 以下は例である。

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
http://schemas.xmlsoap.org/soap/envelope/">
      <SOAP:Header>
            <Message xmlns:eb="http://www.oasis-
open.org/committees/ebxml-msg/schema/msg-header-3_0.xsd"
                  xsi:schemaLocation="http://www.oasis-
open.org/committees/ebxml-msg/schema/msg-header-3_0.xsd
                  http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-3_0.xsd" eb:version="3.0">
                  <eb:UserMessage>
                        <eb:MessageInfo >...</eb:MessageInfo>
                        ...
                        <eb:PayloadInfo >...</eb:PayloadInfo>
                        ...
                  </eb:UserMessage>
            </eb:Message>
      </SOAP:Header>
      <SOAP:Body>
            ...
      </SOAP:Body>
</SOAP:Envelope>
```

5.1.3.3　SOAP のヘッダー要素

SOAP ヘッダー要素は、SOAP エンベロープ要素の最初の子要素である。これは、名前空間 "http://schemas.xmlsoap.org/soap/envelope/"に対して、SOAP エンベロープの名前空間宣言と一致する名前空間修飾子を持たなければならない。

5.1.3.4　SOAP のボディ要素

SOAP ボディ要素は、SOAP エンベロープ要素の 2 番目の子要素である。これは、名前空間 "http://schemas.xmlsoap.org/soap/envelope/"に対して、SOAP エンベロープの名前空間宣言と一致する名前空間修飾子を持たなければならない。

（注）ebXML 電文搬送サービス V2.0 とは異なり、ebXML 電文搬送サービス第 3 版は、SOAP ボディ内の要素を定義したり利用したりしない。それは、完全にユーザー指定のペイロードデータのために確保されている。

5.1.3.5　ebXML の SOAP 拡張

ebXML 電文搬送サービス電文は、“eb”が ebXML 電文搬送サービス V3.0 の名前空間の場合、拡張要素 eb:Message によって SOAP 電文を拡張する。

安全性ヘッダー（wsse:Security）や信頼性ヘッダーなどのように、ebXML 電文搬送サービス電文搬送のある側面をサポートする他のヘッダーが存在する場合もある。それらは ebXML 電文搬送サービス名前空間の下にはない。

5.1.3.6　属性 id

本仕様に定義されている ebXML の SOAP 拡張要素は、XML ID である属性 id を持つ。 これは、SOAP 電文内にある要素を一意に識別する能力を規定するために追加される場合がある。参照要素にある"#<idvalue>"の URI を指定することによって個別の ebXML の SOAP 拡張要素が包含または排除の対象となるように、属性 id は、デジタル署名を ebXML の SOAP に適用する際に使用される。

5.1.3.7　属性 version

必須のバージョン属性は、ebXML の SOAP ヘッダーの拡張が適合する ebXML 電文搬送サービスヘッダー仕様のバージョンを示す。その目的は、今後のバージョニングを可能性にすることである。本仕様に適合させるために、本仕様に定義されている SOAP 拡張要素のバージョン属性は、"3.0"の値を取らなければならない。ebXML 電文搬送サービス電文は、"3.0"以外の値を持つ SOAP ヘッダー拡張要素を包含する場合がある。本仕様に適合し、"3.0"以外のバージョンの ebXML の SOAP 拡張を伴う電文を受信する実装は、識別されたそのバージョンを認識し、それを処理できるならば、その電文も処理できる。この実装は、識別されたバージョンを認識しなかった場合、エラー（詳細は未定）で応答しなければならない。バージョン属性は、先に定義された ebXML 電文搬送サービスの名前空間によって修飾された名前空間でなければならない。

5.1.4　ebXML 電文搬送サービスヘッダー

MIME パッケージングの場合、電文パッケージのルートボディ部分は、添付ファイルつき SOAP 電文 [SOAPATTACH] W3C 注釈に定義されているように、SOAP 電文である。このルートパーツは、常に ebXML 電文搬送サービスヘッダーを包含する。

ルート部分に対する MIME の Content-Type ヘッダーは、[SOAP11]電文を包含している MIME ボディパートの MIME メディア型と一致させるために、"text/xml"という値を取らなければならない。Content-Type ヘッダーは、属性"charset" を包含する場合がある。例は以下のとおりである。

```
Content-Type: text/xml; charset="UTF-8"
```

MIME の属性 charset は、SOAP 電文を生成するために使用される文字集合を識別する。この属性の意味は、[RFC3023]に規定されているように、text/xml の"charset parameter / encoding

considerations" に説明されている。有効な値のリストは[IANAMEDIA]にある。

両方が存在する場合、MIME の属性 charset は、SOAP 電文のエンコーディング宣言と等しくなる（SHALL)。その場合、MIME の属性 charset は、SOAP 電文を生成する際に使用されるエンコーディングと矛盾する値を包含してはならない。

相互運用性を最大限にするには、この文をエンコーディングする際に UTF-8 [UTF8]を使用するのが望ましい。text/xml [RFC3023]のメディア型に定義されている処理ルールにより、この MIME 属性はデフォルトを持たない。

MIME のルートパーツの例：

以下の事例は、ebXML 電文搬送サービスの MIME パッケージングのルートパーツの例を表す。

```
Content-ID: <messagepackage-123@example.com>
Content-Type: text/xml; charset="UTF-8"

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP:Header>
                <eb:Message>
                …
                </eb:Message>
        </SOAP:Header>
        <SOAP:Body>
        …
        </SOAP:Body>
</SOAP:Envelope>
```

5.1.5 ペイロードコンテナ

添付ファイルを伴う SOAP 電文[SOAPATTACH]の仕様と適合している電文パッケージ内に、SOAP ボディ以外の他のペイロードコンテナが存在する場合もある。

電文パッケージ内にアプリケーションペイロードが存在しない場合、SOAP ボディは空でなければならず、かつ、追加のペイロードコンテナが存在してはならない。

それぞれのペイロードコンテナの内容は（SOAP ボディを含む）、要素 /eb:Message/eb:UserMessage/eb:PayloadInfo 内で識別されなければならない。

XML 文でなければならない SOAP ボディを除き、ebXML 電文搬送サービスの仕様には、構造やアプリケーションペイロードの内容に規定も制限もまったくない。ペイロードは、単純な平文オブジェクトの場合もあれば、ネストされた複雑なマルチパートオブジェクトの場合もある。ペイロードオブジェクトの構造および構成の仕様は、ebXML 電文搬送サービスを使用した業務プロセスまたは情報交換を定義する組織の特権である。

ebXML 電文搬送サービスヘッダーを包含する SOAP 電文の例

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
http://schemas.xmlsoap.org/soap/envelope/">
       <SOAP:Header
xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
3_0.xsd"
xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-
header-3_0.xsd
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-3_0.xsd">
              <eb:Message eb:version="3.0" SOAP:mustUnderstand="1">
                     <eb:UserMessage>
                            ...
                            <eb:PayloadInfo>
                                   ...
                            </eb:PayloadInfo>
                            ...
                     </eb:UserMessage>
              </eb:MessageHeader>
       </SOAP:Header>
       <SOAP:Body
xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
3_0.xsd"
xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-
header-3_0.xsd
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-3_0.xsd">
              ...
       </SOAP:Body>
</SOAP:Envelope>
```

5.2　電文ヘッダーの拡張

5.2.1　コンテナ要素 eb:Message

必須の要素 eb:Message は、SOAP ヘッダーの子要素である。これは、ユーザー電文またはシグナル電文のいずれかのコンテナである。これは、要素 eb:timestamp も包含する。

ユーザー電文の場合、ebXML ヘッダーブロックは以下のような形式になる。

```
<eb:Message>
       <eb:UserMessage>
              ...(header elements of the ebMS user message)
       </eb:UserMessage>
</eb:Message>
```

シグナル電文の場合、ebXML ヘッダーブロックは以下のような形式になる。

```
<eb:Message>
       <eb:SignalMessage>
              <eb:[signalname]>
                     ...(header elements of this ebMS signal message)
              </eb:[signalname]>
       </eb:SignalMessage>
</eb:Message>
```

例えば、signalname は “PullRequest”になり得る。

要素 eb:Message は、以下の属性を持つ。

- eb:Message/@eb:version: この値は“3.0”に設定されていなければならない。この属性は必須である。

・eb:Message/@SOAP:mustUnderstand: これは、要素の内容が MSH(電文処理機能）によって理解されなければならないかどうかを示す。SOAP の名前空間に修飾される名前空間を持つこの属性は必須である（http://schemas.xmlsoap.org/soap/envelope/）。この属性は、その要素が理解されなければならない（MUST）か、あるいは拒否されなければならないかを示す'1'（真）の値を持たなければならない。

要素 eb:Message は、以下の子要素を持つ。

・eb:Message/eb:UserMessage: 任意の要素 UserMessage は、ユーザー電文に対するすべてのヘッダー情報を包含する。この要素が存在しない場合、シグナル電文を記述している要素が存在しなければならない。

・eb:Message/eb: SignalMessage/eb:[*signalname*]: 任意のこの要素は、シグナル電文にちなんだ名前を付けられる。これは、シグナル電文に対するすべてのヘッダー情報を包含する。

ebXML 電文搬送サービス電文ヘッダーの例

```
<SOAP:Header …>

  <eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >

    <eb:UserMessage eb:syncresp=”false”>

      <eb:MessageInfo>
        <eb:timestamp>2000-07-25T12:19:05</eb:timestamp>
        <eb:MessageId>UUID-2@example.com</eb:MessageId>
        <eb:RefToMessageId>UUID-1@example.com</eb:RefToMessageId>
      </eb:MessageInfo>

      <eb:PartyInfo>
        <eb:From> (no change beside renaming)
          <eb:PartyId>uri:example.com</eb:PartyId>
            <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
        </eb:From>

        <eb:To> (no change beside renaming)
          <eb:PartyId eb:type="someType">QRS543</eb:PartyId>
          <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
        </eb:To>
      </eb:PartyInfo>

      <eb:CollaborationInfo>
        <eb:AgreementRef>http://www.oasis-open.org/cpa/123456
        </eb:AgreementRef>
        <eb:Service>QuoteToCollect</eb:Service>
        <eb:Action>NewPurchaseOrder</eb:Action>
        <eb:ConversationID>4321</eb:ConversationID>
      </eb:CollaborationInfo >

      <eb:MessageProperties>
        <eb:property name=”ProcessInst”>PurchaseOrder:123456
        </eb:property>
        <eb:property name=”ContextID”> 987654321
        </eb:property>
      </eb:MessageProperties >

      <eb:PayloadInfo>
        <eb:Partref eb:id="…" eb:idref="cid:foo | #idref">
          <eb:Schema eb:location=http://foo/bar.xsd eb:version="2.0"/>
          <eb:Description xml:lang="en-US">Purchase Order for 100,000
foo widgets</eb:Description>
        </eb:Partref>
      </eb:PayloadInfo>

    </eb:UserMessage>
  </eb:Message>

</SOAP:Header>
```

5.2.2 要素 eb:Message/eb:UserMessage

この要素は以下の属性を持つ。

- eb:Message/eb:UserMessage/@syncresp: この任意のブール属性は、使用中の MEP の型およびこの MEP にある電文の役割によって決定された値を持つ。この電文が SOAP 要求応答 MEP の要求として送信され、かつ、同期 ebXML 電文搬送サービスユーザー電文応答が期待される

場合（同じ MEP インスタンスの応答行程として）、この属性の値は「真」でなければならない。「真」ではない場合、その値を「偽」（デフォルトの値）とするか、あるいは、その属性が存在しないようにしなければならない。

この要素は、以下の子要素を持つ。

- eb:Message/eb:UserMessage/eb:MessageInfo: この必須の要素は、一度だけ現れ、電文を識別するデータを包含し、他の電文の識別子と関連を持つ。
- eb:Message/eb:UserMessage/eb:PartyInfo: この必須の要素は、一度だけ現れ、送信元の当事者と送信先の当事者に関するデータを包含する。
- eb:Message/eb:UserMessage/eb:CollaborationInfo: この必須の要素は、一度だけ現れ、当事者間のコラボレーションを促進する要素を包含する。
- eb:Message/eb:UserMessage/eb:MessageProperties: この任意の要素は、最大で 1 回現れ、ユーザー特有の電文プロパティを包含する。プロパティなどがヘッダーの一部として、さらに効率の良いモニタリング、相互関連付け、振り分け、検証機能を規定する（ただし、これらは ebXML 電文搬送サービス仕様の対象外である）。そうでなければ、ペイロードアクセスが必要とされる。
- eb:Message/eb:UserMessage/eb:PayloadInfo: この任意の要素は、最大で 1 回現れ、ペイロード文がペイロードコンテナに包含されているように電文の一部として包含されているか、あるいは、URL 経由でアクセス可能なリモートリソースとして包含されている電文と関連のあるペイロードデータを識別する。PayloadInfo の目的は、(a)このユーザー電文と関連のある特定のペイロードを直接展開しやすくすること、および、(b)ペイロードを構文解析せずに処理できるかどうかをアプリケーションが決定できるようにすることである。

5.2.3 要素 eb:Message/eb:UserMessage/eb:MessageInfo

この要素は以下の子要素を持つ。

- eb:Message/eb:UserMessage/eb:MessageInfo/eb:timestamp: 必須の要素 Timestamp は、電文ヘッダーが生成された日付を表す値を持ち、dateTime（[XMLSCHEMA]参照）に適合している。これは UTC として表現されなければならない。要素 Timestamp で、識別子'Z'を包含することによって UTC を示すのは任意である。
- eb:Message/eb:UserMessage/eb:MessageInfo/eb:MessageId: この必須の要素は、それぞれの電文に対し、MessageId [RFC2822]に適合しているグローバルに一意の識別子を表す値を持つ。（注）Message-Id および Content-Id の MIME ヘッダーでは、値は常に山括弧で囲まれている。しかし、mid:または cid:スキーマ URI の参照および要素 MessageId と RefToMessageId は、これらの区切り文字を含んではならない。
- eb:Message/eb:UserMessage/eb:MessageInfo/eb:RefToMessageId: この任意の要素は最大 1 回現れる。この要素が存在する場合、使用中の MEP に適合する方法で（MEP の項参照）この電文が関連を持つ、ebXML 電文搬送サービス電文の MessageId の値を包含しなければならない（MUST）。
- eb:Message/eb:UserMessage/eb:MessageInfo/eb:PipeId: この任意の要素は最大 1 回現れる。これは、電文が割り当てられている電文パイプを識別する URI を包含する。MSH(電文処理機

能）がプル搬送モードで、これが PullRequest シグナル電文への応答として送信されたユーザー電文の場合、この要素は存在し、PullRequest にある属性 eb:fromPipe と同じ値を取らなければならない。MSH(電文処理機能）がプッシュ搬送モードの場合、この要素がないということは、デフォルトのパイプが使用されることを示す。

5.2.4 要素 eb:Message/eb:UserMessage/eb:PartyInfo

この要素は以下の子要素を持つ。

- eb:Message/eb:UserMessage/eb:PartyInfo/eb:From: この必須の要素は、一度だけ現れ、送信元の当事者に関する情報を包含する。
- eb:Message/eb:UserMessage/eb:PartyInfo/eb:To: この必須の要素は、一度だけ現れ、送信先の当事者に関する情報を包含する。

5.2.5 要素 eb:Message/eb:UserMessage/eb:PartyInfo/eb:From

この要素は以下の子要素を持つ。

- eb:Message/eb:UserMessage/eb:PartyInfo/eb:From/eb:PartyId: 必須の要素 PartyId は 1 回以上現れる。
- eb:Message/eb:UserMessage/eb:PartyInfo/eb:From/eb:Role: 任意の要素 eb:Role は、まったく現れないか一度だけ現れる。役割要素は、電文を送信（要素 From の子要素として存在する場合）または受信（要素 To の子要素として存在する場合）する当事者に付与された役割（fromAuthorizedRole または toAuthorizedRole）を識別する。役割要素の値は、空ではない文字列である。取り得る値は CPA（このような文が使用されるなら）に規定されている。

例：以下のフラグメントは、要素 From の用途を表す。

```
<eb:From>
<eb:PartyId eb:type="urn:duns">123456789</eb:PartyId>
<eb:PartyId eb:type="SCAC">RDWY</PartyId>
<eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
</eb:From>
```

5.2.6 要素 eb:Message/eb:UserMessage/eb:PartyInfo/eb:To

この要素は、eb:Message/eb:UserMessage/eb:PartyInfo/eb:From と同じ子要素を持つ。

例：以下の事例は、要素 To の用途を表す。

```
<eb:To>
<eb:PartyId>mailto:joe@example.com</eb:PartyId>
<eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
</eb:To>
```

5.2.7 要素 eb:Message/eb:UserMessage/eb:PartyInfo/eb:From/eb:PartyId

この要素は、当事者を識別する、あるいは、この当事者の識別子の 1 つである文字列の値の内容を持つ。

これは、単一の属性@eb:type を持つ。型属性は、要素 PartyId の内容にある文字列が属する名

前のドメインを表す。型属性の値が URI になるのが望ましい。これらの値が EDIRA [ISO6523]、EDIFACT [ISO9735]、または ANSI ASC X12 [ASCI05]のレジストリから取られるとさらに望ましい。
要素 PartyId の例は以下である。

```
<eb:PartyId eb:type="urn:duns">123456789</eb:PartyId>
```

属性 eb:PartyId /@eb:type が存在しない場合、要素 PartyId の内容は URI [RFC2396]でなければならない(MUST)。そうでなければ、受信 MSH(電文処理機能)は、errorCode に Inconsistent を、severity に Error をセットして、エラーを返さなければならない。要素 eb:PartyId の内容は、URI とするのが強く推奨される。

5.2.8 要素 eb:Message/ eb:UserMessage/eb:CollaborationInfo

この要素は以下の子要素を持つ。

- eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:AgreementRef: この任意の要素は、まったく現れないか一度だけ現れる。要素 AgreementRef は、当事者間の電文交換を支配するエンティティまたは成果物を識別する文字列である。
- eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:Service: この必須の要素は一度だけ現れる。これは、電文上で行動するサービスを識別している文字列で、サービスの設計者によって規定されている。
- eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:Action: この必須の要素は一度だけ現れる。この要素は、これらの複数をサポートできるサービス内の動作またはアクティビティを識別する文字列である。
- eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:ConversationID: この必須の要素は一度だけ現れる。この要素は、当事者間の会話を構成する、関連のある電文の集合を識別する文字列である。

5.2.9 要素 eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:AgreementRef

AgreementRef は、交換を支配する動作コンテキストを識別する文字列の値である。電文の受信者は、この動作コンテキスト、または、送信および受信当事者間の関連のある同意に対する AgreementRef を解決できなければならない。

要素 AgreementRef の値は、2 人の当事者が相互に同意に達した名前空間内において一意でなければならない (MUST)。これは、From および To PartyId の値の連結、一方の当事者のインターネットドメイン名を接頭につけた URI、または 他のネーミングサービスやレジストリサービスによって提供および管理されている名前空間となる場合がある。望ましいのは (RECOMMENDED)、AgreementRef が URI となることである。AgreementRef は、[ebCPPA]に定義されているように、CPA のインスタンスを参照する場合がある。
要素 CPAId の例は以下のとおりである。

```
<eb:AgreementRef>http://example.com/cpas/ourcpawithyou.xml</eb:AgreementRef>
```

CPA が参照され、参照された CPA と電文が矛盾すると受信者が判断した場合、 この矛盾の適切な処理については本仕様に定義されていない。したがって送信者は、受信者がその矛盾に対処できるか事前にわからない場合、そのような電文を生成すべきではない。受信 MSH(電文処理機能）が不一致を発見した場合は、errorCode を Inconsistent、重大度を Error として、その不一致を報告しなければならない。AgreementRef が認識されない場合、errorCode を NotRecognized、重大度を Error としてそのことを報告しなければならない。
要素 AgreementRef は、電文を送信および受信する当事者が参照の値（例えば、CPA を基にした同意表現を使用している当事者にとって、その値は“ebcppa2.1”である）をどのように解釈するかを示す単一の属性@type を持つ。型属性の値に制約はない。つまり、本仕様の対象外であるプロファイリングの実行に委ねられる。

5.2.10 要素 eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:Service

この要素は、電文上で動作するサービスを識別する。その実際の意味については本仕様の対象外である。サービスの設計者は、標準機関の場合もあれば、個人または企業の場合もある。
サービス要素の例は以下のとおりである。

```
<eb:Service>urn:services:SupplierOrderProcessing</eb:Service>
```

サービス要素は、電文を送信および受信する当事者がその要素の値をどのように解釈するかを示す単一の属性@type を持つ。型属性の値に制約はない。型属性が存在しない場合、サービス要素の内容は URI（[RFC2396]参照）でなければならない。URI ではない場合、MSH(電文処理機能) は、errorCode を Inconsistent、重大度を Error としてエラーを報告しなければならない（「エラー処理モジュール」の項参照)。

5.2.11 要素 eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:Action

この要素は、サービス内の動作またはアクティビティを識別する文字列である。その実際の意味については本仕様の対象外である。アクションは、それを定義するサービスにおいて一意でなければならない。要素 Action の値は、サービスの設計者によって規定される。
アクション要素の例は以下のとおりである。

```
<eb:Action>NewOrder</eb:Action>
```

サービスまたはアクション要素の値のどちらかが受信 MSH(電文処理機能）によって認識されない場合は、errorCode を NotRecognized、重大度を Error としてエラーを報告しなければならない。

5.2.12 要素 eb:Message/eb:UserMessage/eb:CollaborationInfo/eb:ConversationId

この要素は、当事者間の会話を構成する、関連のある電文の集合を識別する文字列である。

（EdNote: この要素および他の CPA の参照を付録に移す）
CPA が eb:AgreementRef によって参照される場合、この CPA と関連のある会話の数は、CPA の要件と適合しなければならない。eb:ConversationID の値は、この CPA のコンテキストにある会話を一意に識別しなければならない。
要素 ConversationId の例は以下のとおりである。

```
<eb:ConversationId>20001209-133003-28572</eb:ConversationId>
```

会話を開始する当事者は、会話に関連するすべての電文に反映される要素 ConversationId の値を決定する。この値の実際の意味については本仕様の対象外である。実装は、識別スキーマと他の実装によって生成された ConversationId の間でマッピングを行うための機能を提供する。

5.2.13 要素 eb:Message/eb:UserMessage/eb:MessageProperties

この要素は、ゼロまたはそれ以上の子要素 eb:Property を持つ。
要素 eb:Property は、xs:anySimpleType（例えば、文字列、URI）で、属性@name を持つ。この値は当事者間で同意に達していなければならない。
この要素の実際の意味については、本仕様の対象外である。この要素は、ebXML 電文搬送サービス特有の機能以外で処理される。これは、ペイロードデータを修飾、またはペイロードを抽象化する情報、または、業務プロセスに電文を結合させる情報を含む場合がある。このようなプロパティのヘッダーにある表現は、ペイロードアクセスを要求しないより効率的なモニタリング、相互関連付け、振り分け、検証機能を規定する（ただし、これらは ebXML 電文搬送サービス仕様の対象外である）。
例：

```
<eb:MessageProperties>
<eb:property name="ContextId">C1234</eb:property>
<eb:property name="processinstanceID">3A4-1234</eb:property>
<eb:property name="transactionID">45764321</eb:property>
</eb:MessageProperties>
```

5.2.14 要素 eb:Message/ eb:UserMessage/eb:PayloadInfo

それぞれの要素 PayloadInfo は、電文と関連のあるペイロードデータを識別する。PayloadInfo の目的は以下である。
・ この ebXML 電文搬送サービス電文と関連のある特定のペイロード部分を簡単に直接展開できるようにする。
・ これらのペイロード部分を構文解析せずに処理できるかどうかをアプリケーションが決定できるようにする。
要素 PayloadInfo は、以下の属性を持つ。
・ 属性 id（詳細については「属性 id」の項参照）
・ 属性 version （詳細については「属性 version」の項参照）
要素 PayloadInfo は以下の子要素を持つ。

・eb:Message/eb:UserMessage/eb:PayloadInfo/eb:Partref

この要素は、まったく現れないか、1 回以上現れる。要素 Partref 自体は、シンプルなリンクである [XLINK]。このコンテキストにおいて XLINK を使用するのは、もっぱら関連を説明している OASIS ebXML 電文搬送サービスに簡潔なボキャブラリを提供するためである。XLINK プロセッサまたはエンジンの使用は必須ではないが、ある実装においては役立つことがわかる。

EdNote:事例の識別子の定義については、 http://www.w3.org/TR/xptr-framework/を参照する。さしあたり、属性 xlink:*を URI と置換する。

5.2.15　要素 eb:Message/ eb:UserMessage/eb:PayloadInfo/eb:Partref

この要素は以下の属性を持つ。

· eb:Message/eb:UserMessage/eb:PayloadInfo/eb:Partref/eb:cid：この任意の属性は、参照されるペイロードオブジェクトの CID URI である値を持つ。例えば、cid:foo または"#idref"である。要素 eb:Partref に属性 eb:cid が存在しないということは、参照されるペイロード部分が SOAP ボディ要素自体であることを示す。例えば、以下に簡潔に述べている宣言は、SOAP ボディはこの ebXML 電文搬送サービス電文にある一意のペイロード部分である、ということである。

```
<eb:PayloadInfo> <eb:Partref/> </eb:PayloadInfo>
```

他の名前空間で修飾された属性が存在することもある。受信 MSH(電文処理機能）は、先に定義された属性以外の無関係な名前空間を無視することを選択できる。

ebXML 電文搬送を使用した業務プロセスまたは情報交換の設計者は、マニフェストが参照するペイロードデータ、そして xlink:role に使用される値を決定する。

この要素は以下の子要素を持つ。

· eb:Message/eb:UserMessage/eb:PayloadInfo/eb:Partref/eb:Schema

この要素は、まったく現れないか、1 回以上現れる。これは、親要素 Partref に識別されているインスタンス文を定義するスキーマを参照する。参照される項目がスキーマ（XML スキーマ、DTD、またはデータベーススキーマ）について何か説明しているスキーマを持つ場合、スキーマ要素は要素 Partref の子要素として存在しなければならない。これは、親要素 Partref によって識別されるペイロードオブジェクトを定義するスキーマおよびそのバージョンの識別方法を提供する。スキーマ要素は以下の属性を含む。(a) 名前空間－必須のスキーマのターゲット名前空間 (b) 場所－必須のスキーマの URI (c) バージョン– スキーマのバージョン

· eb:Message/eb:UserMessage/eb:PayloadInfo/eb:Partref/eb:Description

この要素は、まったく現れないか、1 回以上現れる。この要素の目的は、ペイロード部分の目的または意図の説明を可読にすることである。説明言語は、必須の属性 xml:lang によって定義される。属性 xml:lang は、XML [XML10]に規定されている識別言語のルールに適合しなければならない。この要素の発生時の xml:lang の値は常に異なる。

例：

```
<eb:PayloadInfo>
      <eb:Partref eb:id="…" eb:idref="cid:foo | #idref">
<eb:Schema eb:location="http://foo/bar.xsd" eb:version="2.0"/>
<eb:Description xml:lang="en-US">Purchase Order for 100,000 foo
widgets</eb:Description>
</eb:Partref>
      <eb:Partref eb:id="…" eb:idref="cid:goo | #idref">
<eb:Schema eb:location="http://goo/bar.xsd" eb:version="2.0"/>
```

```
<eb:Description xml:lang="en-US">Purchase Order for 50,000 goo
widgets</eb:Description>
</eb:Partref>
</eb:PayloadInfo>
```

5.3 電文搬送シグナル

5.3.1 要素 eb:Message/eb:SignalMessage

この要素は、要素 eb:UserMessage の代替要素である。これは以下の 2 つの子要素を持つ。

・eb:Message/eb:SignalMessage/eb:MessageInfo: この必須の要素は、ユーザー電文に定義されているように eb:MessageInfo と類似している。

・eb:Message/eb:SignalMessage/eb:[*SignalName*]

この必須の要素は、ebXML 電文搬送サービスシグナル電文を包含する。

さらに、シグナルの認証が必要とされる場合、要素 wsse:SecurityTokenReference を子要素として追加できる。

ebXML 電文搬送サービスシグナルは、SOAP ボディを必要としない。つまり、SOAP ボディが空ではない場合、シグナルの解釈が関与する限り、ebXML 電文搬送サービスシグナルは MSH（電文処理機能）によって無視されなければならない。

5.3.2 要素 eb:Message/eb:SignalMessage/eb:PullRequest

この要素は以下の属性を持つ。

```
eb:Message/eb:SignalMessage/eb:PullRequest/@eb:fromPipe
```

この必須の属性は、その電文がどのパイプからプルされなければならないかを示すパイプ Id（URI）を包含する。言い換えると、これは搬送を開始する受信 MSH(電文処理機能）を表す。

5.3.3 エラー電文のパッケージング

EdNote: この項が必要かどうか判断する。

6 安全性モジュール

ebXML 電文搬送サービスは、まさにその性質上安全性のリスクを有する。電文搬送サービスは、以下を用いることによってリスクにさらされる場合がある。

・不正アクセス

・データの保全性および/または機密性の攻撃（例えば、介入者（man-in-the-middle）攻撃を通して）

・ サービス拒絶、および、なりすまし

本仕様は、一般に利用可能な技術に基づいて、重要なリスクを扱うために、選ばれたプロファイルの集合、または、選択された解決策の組み合わせについて説明する。それぞれの特定のプロファイルには、位置づけられていないリスクの説明も含む。

解決策となるアプリケーションは、本来のリスクの評価、および、リスクにさらされる可能性がある資産の価値とバランスを保たなければならない。

6.1 要素セキュリティ

ebXML 電文搬送サービス電文は、安全性の解決策を提供するために、電子的に署名されたり暗号化されたりする。電文の署名および暗号のサポートは、Web サービスセキュリティおよび Web サービスセキュリティの添付ファイルを伴う SOAP 電文プロファイルに規定されている。

名前空間を定義した Web サービスセキュリティに属している、ターゲット毎のゼロまたは 1 つの要素セキュリティは、SOAP ヘッダーの子要素として表される場合がある。安全性の要素は、Web サービスセキュリティに従って適合とされた名前空間でなければならない。要素セキュリティの構造および内容は、Web サービスセキュリティの仕様および Web サービスセキュリティの添付ファイルを伴う SOAP 電文プロファイルに準拠しなければならない。

相互運用性を高めるためには、安全性の要素は WS 相互運用性基本安全性プロファイルのバージョン 1.0 および WS 相互運用性添付プロファイルのバージョン 1.0 に準拠しなければならない。

6.2 署名電文

ebXML 電文搬送サービス電文は、以下のステップに従って署名される。

1. Web サービスセキュリティに規定されているように、安全性の要素を生成する。
2. 以下を規定している要素 BinarySecurityToken を生成する。

```
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3"
```

3. 安全性の要素の子要素として BinarySecurityToken を追加する。
4. Web サービスセキュリティ規定されている SecurityTokenReference を包含する要素 KeyInfo を生成する。SecurityTokenReference は、先に生成された要素 BinarySecurityToken を参照する。
5. Web サービスセキュリティに規定されているとおり、SOAP エンベロープのための、SignatureMethod、CanonicalizationMethod 、および SOAP エンベロープ参照要素を伴う要素を生成する。
6. Web サービスセキュリティに規定されているとおり、正規化ののち、SignedInfo で特定されているアルゴリズムに基づいて SignedInfo に SignatureValue を計算する。
7. Web サービスセキュリティに規定されているとおり、要素 SignedInfo、KeyInfo および SignatureValue を包含する要素 Signature を構築する。
8. 先に生成した要素 Security に要素 Signature を含める。

x509-token-profile の ValueType を持つ BinarySecurityToken の使用で望ましいのは、EncodingType –Base64Binary である。

```
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/10/xml-exc-c14n#"/>
```

要素 SignatureMethod が存在し、この要素は属性 Algorithm を持つ。属性 Algorithm の値として望ましいのは以下である。

```
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
```

この望ましい値は、適合する ebXML 電文搬送サービスソフトウェアの実装によってサポートされる。

適合する MSH(電文処理機能） 実装は、XML 署名仕様で定義されているように、分離（Detached）署名をサポートする必要がある。

コンテナ要素 eb:Message と SOAP ボディを署名に含めることが望ましい。

MSH(電文処理機能）実装は、XML 署名仕様で定義されているように、エンベロープされた（Enveloped）署名をサポートする場合がある。エンベロープされた署名は、XML 要素が SOAP ヘッダーに追加されるのを防ぐための安全性のレベルを追加する。エンベロープされた署名は、電文を処理する仲介能力を制限する場合がある。

電子的に署名された ebXML の SOAP 電文の例：

```
Mime-Version: 1.0
Content-Type: text/xml
Content-Transfer-Encoding: binary
SOAPAction: ""
Content-Length: 4871

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-3_0.xsd">
        <eb:Message eb:version="3.0" id="ebMessage" soap:mustUnderstand="1">
            <eb:UserMessage eb:syncresp="false">
                <eb:MessageInfo>
                    <eb:Timestamp>2005-10-28T17:29:54.606Z</eb:Timestamp>
                    <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>
                    <eb:RefToMessageId>UUID-1@msh-
server.example.com</eb:RefToMessageId>
                </eb:MessageInfo>
                <eb:PartyInfo>
                    <eb:From>
                        <eb:PartyId>uri:msh-server.example.com</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
                    </eb:From>
                    <eb:To>
                        <eb:PartyId eb:type="someType">QRS543</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
                    </eb:To>
                </eb:PartyInfo>
                <eb:CollaborationInfo>
                    <eb:AgreementRef>http://msh-
server.example.com/cpa/123456</eb:AgreementRef>
                    <eb:Service eb:type="someType">QuoteToCollect</eb:Service>
                    <eb:Action>NewPurchaseOrder</eb:Action>
                    <eb:ConversationId>22fedc9d-d450-4482-934b-
ff35ba41953f</eb:ConversationId>
                </eb:CollaborationInfo>
                <eb:MessageProperties>
                    <eb:Property
name="ProcessInst">PurchaseOrder:123456</eb:Property>
                    <eb:Property name="ContextID">987654321</eb:Property>
                </eb:MessageProperties>
                <eb:PayloadInfo>
                    <eb:Partref eb:id="ID6039001113052059465 5"
```

```
                        eb:idref="cid:PO_Image">
                        <eb:Description xml:lang="en-US">PO Image</eb:Description>
                    </eb:Partref>
                </eb:PayloadInfo>
            </eb:UserMessage>
        </eb:Message>
        <wsse:Security soap:mustUnderstand="1"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="signingCert">...</wsse:BinarySecurityToken>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#ebMessage">
                        <ds:Transforms>
                            <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>hONgCCP9O1KwOZhZXburzsUOMk0=</ds:DigestVal
ue>
                    </ds:Reference>
                    <ds:Reference URI="#myBody">
                        <ds:Transforms>
                            <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>8Yqxg2jXETXZBgbJ+Ua8Ka2edDw=</ds:DigestVal
ue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>
                    jUKqmjw1hqsbeK+16LAmhtSmZ0Op65jNdyZdh0Fbl+WPB7W1GQN4uUNJpvZc21
Vc461O14WI
                    OIm8O3s1aj49qHUvwh6aTQ5g2Lt37C3eHDrxFNxrHnUM4qsNQFbBB9QiwS50KI
2gVKtqrax8
                    6YHbVB48ZWE5I36VuOazHKz3EjE= </ds:SignatureValue>
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference
                        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">
                        <wsse:Reference URI="#signingCert"/>
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
            </ds:Signature>
        </wsse:Security>
    </soap:Header>
    <soap:Body id="myBody">
        <tru:StockSymbol
xmlns:tru="http://payloads.example.com/">QQQ</tru:StockSymbol>
    </soap:Body>
</soap:Envelope>
```

6.3　添付ファイルを伴う SOAP 電文の署名

添付ファイルを包含する ebXML 電文搬送サービス電文は、以下のステップに従い署名される。

1. Web サービスセキュリティに規定されているように、要素 Security を生成する。
2. 以下のように特定している要素 BinarySecurityToken を生成する。

```
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3"
```

3.要素 Security の子要素として BinarySecurityToken を追加する。
4. Web サービスセキュリティに規定されているように、SecurityTokenReference を包含する要素 KeyInfo を生成する。SecurityTokenReference は、先に生成された要素 BinarySecurityToken を参照する。
5. Web サービスセキュリティで規定されているように、SignatureMethod、CanonicalizationMethod および SOAP エンベロープに対する参照要素を伴う要素 SignedInfo を生成する。Web サービスセキュリティの添付ファイルを伴う SOAP 電文プロファイルで規定されているように、各添付ファイルに対する参照要素を生成する。
6. 正規化した後、Web サービスセキュリティで規定されているように、SignedInfo に規定されているアルゴリズムに基づいて SignedInfo に SignatureValue を正規化して計算する。
7. Web サービスセキュリティに規定されているように、SignedInfo、KeyInfo および SignatureValue の要素を包含する要素 Signature を構築する。
8. 先に生成された要素 Security に要素 Signature を含める。

適合する MSH(電文処理機能) 実装は、Attachment-Content-Only（添付ファイルの内容のみ）の変換をサポートする必要がある。適合する MSH(電文処理機能) 実装は、Attachment-Complete（添付ファイル全体）の変換をサポートすることが望ましい。

コンテナ要素 eb:Message、および、包含されているペイロードの MIME ボディ部分すべてを署名に含めることが望ましい。

電子的に署名された添付ファイルを伴う ebXML の SOAP 電文の例：

```
Mime-Version: 1.0
Content-Type: multipart/related; type="text/xml";  boundary="----
=_Part_0_10369852.1130520597081"
SOAPAction: ""
Content-Length: 5163

------=_Part_0_10369852.1130520597081
Content-Type: text/xml
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-3_0.xsd">
        <eb:Message eb:version="3.0" id="ebMessage" soap:mustUnderstand="1">
            <eb:UserMessage eb:syncresp="false">
                <eb:MessageInfo>
                    <eb:Timestamp>2005-10-28T17:29:56.963Z</eb:Timestamp>
                    <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>
                    <eb:RefToMessageId>UUID-1@msh-
server.example.com</eb:RefToMessageId>
                </eb:MessageInfo>
                <eb:PartyInfo>
                    <eb:From>
                        <eb:PartyId>uri:msh-server.example.com</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
                    </eb:From>
                    <eb:To>
                        <eb:PartyId eb:type="someType">QRS543</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
                    </eb:To>
                </eb:PartyInfo>
                <eb:CollaborationInfo>
                    <eb:AgreementRef>http://msh-
server.example.com/cpa/123456</eb:AgreementRef>
                    <eb:Service eb:type="someType">QuoteToCollect</eb:Service>
                    <eb:Action>NewPurchaseOrder</eb:Action>
                    <eb:ConversationId>213dc65a-a86a-4a14-b36b-
90706ff18003</eb:ConversationId>
                </eb:CollaborationInfo>
```

```
                <eb:MessageProperties>
                    <eb:Property
name="ProcessInst">PurchaseOrder:123456</eb:Property>
                    <eb:Property name="ContextID">987654321</eb:Property>
                </eb:MessageProperties>
                <eb:PayloadInfo>
                    <eb:Partref eb:id="ID:13965142113052059696 7"
                        eb:idref="cid:PO_Image">
                        <eb:Description xml:lang="en-US">PO Image</eb:Description>
                    </eb:Partref>
                </eb:PayloadInfo>
            </eb:UserMessage>
        </eb:Message>
        <wsse:Security soap:mustUnderstand="1"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="signingCert">...</wsse:BinarySecurityToken>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#ebMessage">
                        <ds:Transforms>
                            <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>3bp59GvvgDHynHeONH1YqDdxmP4=</ds:DigestVal
ue>
                    </ds:Reference>
                    <ds:Reference URI="cid:PO_Image">
                        <ds:Transforms>
                            <ds:Transform
                                Algorithm="http://docs.oasis-
open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-Only-
Transform"
                                />
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>R4hCV4K4I5QZdSsrP4KrLu46hFo=</ds:DigestVal
ue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>
                    qzFvmFphuRcrKYh7nPRA7TBWFwCe+Xm8CZe65CidEMLlGls6OMHaWCgJ/R6NCi
AWIQ3onu9m
                    nJurP2sQsxl8kRoPq9oUAsXyOyuymHTC9L9PBwC+KN5six+RDjybVZ/fW8jioB
d2tKyjfgz9
                    CfyXg1F6JG26IipGWj61E5+eG9w= </ds:SignatureValue>
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference
                        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">
                        <wsse:Reference URI="#signingCert"/>
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
            </ds:Signature>
        </wsse:Security>
    </soap:Header>
    <soap:Body/>
</soap:Envelope>

------=_Part_0_10369852.1130520597081
Content-Type: image/jpeg
```

```
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=empty.jpg
Content-Id: <PO_Image>

/9j/4AAQSkZJRg==

------=_Part_0_10369852.1130520597081--
```

6.4　電文の暗号化

ebXML 電文搬送サービス電文は、以下のステップに従い暗号化される。

1. Web サービスセキュリティに規定されているように、要素 Security を生成する。
2. 以下のように特定している要素 BinarySecurityToken を生成する。

```
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3"
```

3. 要素 Security の子要素として BinarySecurityToken を追加する。
4. 要素 Security の子要素として EncryptedKey を生成する。要素 EncryptedKey は、そのキーを使用して暗号化された各要素 EncryptedData に対する DataReference を包含しなければならない。要素 EncryptedKey は、添付された BinarySecurityToken を参照する SecurityToken 参照サブ要素を包含する要素 KeyInfo を包含しなければならない。
5. 暗号化されるデータ項目の場所を見つける。
6. 次のようにデータ項目を暗号化する。XML の暗号化仕様の処理ルールに従って、ターゲットの SOAP エンベロープ内の各 XML 要素または要素内容を暗号化する。選択されたそれぞれの元の要素または要素内容は、結果として生じる要素 EncryptedData によって削除および置換されなければならない。
7. 生成された要素 EncryptedData を参照する要素 DataReference を生成する。生成された要素 DataReference を ReferenceList に追加する。

MSH(電文処理機能）実装は、コンテナ要素 eb:Message を暗号化する場合がある。セクション eb:PartyInfo は、復号化の前に電文経路指定に役立つように使用される。セクション eb:PartyInfo は暗号化しないのが望ましい。

要素 EncryptionMethod が存在し、属性 Algorithm を持つ。属性 Algorithm に望ましい値は以下である。

```
<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
```

この望ましい値は、適合するすべての ebXML 電文搬送サービスソフトウェアの実装によってサポートされる。

暗号化された ebXML の SOAP 電文の例

```
Mime-Version: 1.0
Content-Type: text/xml
Content-Transfer-Encoding: binary
SOAPAction: ""
Content-Length: 4591

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-3_0.xsd">
        <eb:Message eb:version="3.0" id="ebMessage" soap:mustUnderstand="1">
            <eb:UserMessage eb:syncresp="false">
                <eb:MessageInfo>
                    <eb:Timestamp>2005-10-28T17:29:57.720Z</eb:Timestamp>
                    <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>
                    <eb:RefToMessageId>UUID-1@msh-
server.example.com</eb:RefToMessageId>
                </eb:MessageInfo>
                <eb:PartyInfo>
                    <eb:From>
```

```
                        <eb:PartyId>uri:msh-server.example.com</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
                    </eb:From>
                    <eb:To>
                        <eb:PartyId eb:type="someType">QRS543</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
                    </eb:To>
                </eb:PartyInfo>
                <eb:CollaborationInfo>
                    <eb:AgreementRef>http://msh-
server.example.com/cpa/123456</eb:AgreementRef>
                    <eb:Service eb:type="someType">QuoteToCollect</eb:Service>
                    <eb:Action>NewPurchaseOrder</eb:Action>
                    <eb:ConversationId>54e98d3b-ed1a-4a70-997a-
d6245d098912</eb:ConversationId>
                </eb:CollaborationInfo>
                <eb:MessageProperties>
                    <eb:Property
name="ProcessInst">PurchaseOrder:123456</eb:Property>
                    <eb:Property name="ContextID">987654321</eb:Property>
                </eb:MessageProperties>
                <eb:PayloadInfo>
                    <eb:Partref eb:id="ID19367511130520597724"
                        eb:idref="cid:PO_Image">
                        <eb:Description xml:lang="en-US">PO Image</eb:Description>
                    </eb:Partref>
                </eb:PayloadInfo>
            </eb:UserMessage>
        </eb:Message>
        <wsse:Security soap:mustUnderstand="1"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="encryptionCert">...</wsse:BinarySecurityToken>
            <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
                <enc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
                    xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd"/>
                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                    <wsse:SecurityTokenReference>
                        <wsse:Reference URI="#encryptionCert"/>
                    </wsse:SecurityTokenReference>
                </KeyInfo>
                <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <CipherValue>bJnt6rUFzNOUAXVd5Xk0g9OciYPT8DmyALCW7CynrtomgeB9x
7MbmheRwkIhdi7tR65FGyfR+uIplbvNwT8O1YCQeCu/oNw1qJXQIXkhGAUNnPrCexuf20b3ppUbpgsBaT9
6O5kKDHh0l3DSwAS29n0xFnXXJDgg+HCSgW8qpxY=</CipherValue>
                </CipherData>
                <ReferenceList xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <DataReference URI="#enc"/>
                </ReferenceList>
            </enc:EncryptedKey>
        </wsse:Security>
    </soap:Header>
    <soap:Body wsu:Id="body"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
        <EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
            xmlns="http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
            <CipherData>
                <CipherValue>r+WpG9C1HOzNpodaY/E07puEdDw2CshGpMWKwfTJUJcegYP2tCaF9
TQAzK55dpE5aTQnMtuIiY93ggDwuvgXNVJAzHuUvBsuS6wFwoLJORcv490+HVp55Q==</CipherValue>
            </CipherData>
        </EncryptedData>
    </soap:Body>
</soap:Envelope>
```

6.5　添付ファイルを伴う SOAP 電文の暗号化

ebXML 電文搬送サービス電文は、以下のステップに従い暗号化される。

1. Web サービスセキュリティに規定されているように、要素 Security を生成する。
2. 以下のように特定している要素 BinarySecurityToken を生成する。

```
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3"
```

3. 要素 Security の子要素として BinarySecurityToken を追加する。
4. 要素 Security の子要素として EncryptedKey を生成する。要素 EncryptedKey は、そのキーを使用して暗号化されたそれぞれの要素 EncryptedData に対する DataReference を包含する。要素 EncryptedKey は、添付された BinarySecurityToken を参照する参照サブ要素 SecurityToken を包含する要素 KeyInfo を包含する。
5. 暗号化されるデータ項目の場所を見つける。
6. 添付ファイルと主な SOAP エンベロープの内容の両方を暗号化する場合は、添付ファイルの処理を最初に行う。
7. XML 暗号化のルールに従い、XML 暗号化を使用してそれぞれの添付ファイル部分を暗号化する。
8. 暗号化されたもの（MIME の内容またはヘッダーを含むすべての MIME の部分）を特定する URI に型属性 EncryptedData を設定する。
9. 暗号化の前に、添付ファイル MIME パートの Content-Type ヘッダーに適合するように属性 EncryptedData MimeType を設定する。
10. 添付ファイルの内容のエンコーディングを反映するように、つまり、暗号化の際に安全性の層に可視であるように、エンコーディング属性 EncryptedData を設定する。
11. 暗号化の前に使用された添付ファイルの同じ参照 URL に EncryptedData CipherReference を設定する。これは、添付ファイル部分の内容 ID を参照している CID スキーマ URL でなければならない。暗号化の後、暗号データを伝えている部分にこの MIME ヘッダーが存在するようにする。
12. MIME パートの参照変換を CipherReference 変換リストに含める。
13. 要素 EncryptedData を要素 Security の先頭に追加する。
14. 元の内容が XML の暗号化ステップによって生成された暗号テキストに置換される添付ファイルの MIME パートを更新する。
15. 添付ファイル MIME パートのヘッダーの MIME Content-Type および暗号データに適した内容の長さ（Content Length）を更新する。
16. それぞれの追加添付ファイルに対してステップを繰り返す。
17. 次のように SOAP ヘッダーのデータ項目を暗号化する。XML 暗号化仕様の処理ルールに従って、ターゲットの SOAP エンベロープ内にある各 XML 要素または要素内容を暗号化する。選択されたそれぞれの元の要素または要素内容は、結果として生じる要素 EncryptedData によって削除および置換されなければならない。
18. 生成された要素 EncryptedData を参照する要素 DataReference を生成する。生成された要素 DataReference を ReferenceList に追加する。

添付ファイルを伴う暗号化された ebXML の SOAP 電文の例

```
Mime-Version: 1.0
Content-Type: multipart/related; type="text/xml";  boundary="----
=_Part_1_1804119.1130520598736"
SOAPAction: ""
Content-Length: 5172

------=_Part_1_1804119.1130520598736
Content-Type: text/xml
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="UTF-8"?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-3_0.xsd">
        <eb:Message eb:version="3.0" id="ebMessage" soap:mustUnderstand="1">
            <eb:UserMessage eb:syncresp="false">
                <eb:MessageInfo>
                    <eb:Timestamp>2005-10-28T17:29:58.665Z</eb:Timestamp>
                    <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>
                    <eb:RefToMessageId>UUID-1@msh-
server.example.com</eb:RefToMessageId>
                </eb:MessageInfo>
                <eb:PartyInfo>
                    <eb:From>
                        <eb:PartyId>uri:msh-server.example.com</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
                    </eb:From>
                    <eb:To>
                        <eb:PartyId eb:type="someType">QRS543</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
                    </eb:To>
                </eb:PartyInfo>
                <eb:CollaborationInfo>
                    <eb:AgreementRef>http://msh-
server.example.com/cpa/123456</eb:AgreementRef>
                    <eb:Service eb:type="someType">QuoteToCollect</eb:Service>
                    <eb:Action>NewPurchaseOrder</eb:Action>
                    <eb:ConversationId>37074bd0-2363-4a7e-ac53-
cf5ce39cecd2</eb:ConversationId>
                </eb:CollaborationInfo>
                <eb:MessageProperties>
                    <eb:Property
name="ProcessInst">PurchaseOrder:123456</eb:Property>
                    <eb:Property name="ContextID">987654321</eb:Property>
                </eb:MessageProperties>
                <eb:PayloadInfo>
                    <eb:Partref eb:id="ID:134857231130520598670"
                        eb:idref="cid:PO_Image">
                        <eb:Description xml:lang="en-US">PO Image</eb:Description>
                    </eb:Partref>
                </eb:PayloadInfo>
            </eb:UserMessage>
        </eb:Message>
        <wsse:Security soap:mustUnderstand="1"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="encryptionCert">...</wsse:BinarySecurityToken>
            <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
                <enc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
                    xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd"/>
                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                    <wsse:SecurityTokenReference>
                        <wsse:Reference URI="#encryptionCert"/>
                    </wsse:SecurityTokenReference>
                </KeyInfo>
                <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <CipherValue>JP63XjqJsLfqkHxkcKGI52gWu11ueF+uumRjb4Va8nchowRDd
t62flJNkN56G31SuVkF8RsKMVzmvjgKewIqxVd/pGniHUkCf42f7MAbfh2b9ALX27sifnJtXh8QzQJJ3Gz
ddO71zNQQK9De/D40w9/QqGWgxfSVplgMjFoxJT8=</CipherValue>
                </CipherData>
                <ReferenceList xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <DataReference URI="#encrypted-attachment"/>
                </ReferenceList>
            </enc:EncryptedKey>
            <EncryptedData Id="encrypted-attachment" MimeType="image/jpeg"
```

```
                Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
profile-1.0#Attachment-Content-Only"
                xmlns="http://www.w3.org/2001/04/xmlenc#">
                <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
                <CipherData>
                    <CipherReference URI="cid:PO_Image">
                        <Transforms>
                            <Transform
                                Algorithm="http://docs.oasis-
open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-Only-
Transform"
                                xmlns="http://www.w3.org/2000/09/xmldsig#"/>
                        </Transforms>
                    </CipherReference>
                </CipherData>
            </EncryptedData>
        </wsse:Security>
    </soap:Header>
    <soap:Body/>
</soap:Envelope>

------=_Part_1_1804119.1130520598736
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64
Content-Id: <PO_Image>
Content-Description: WSS XML Encryption message; type="image/jpeg"

jsXi6IxcXpEVuerYtv0oR3qEubT4I0ii

------=_Part_1_1804119.1130520598736--
```

6.6　電文の署名と暗号化

ebXML 電文搬送サービス電文は、 Web サービスセキュリティ を使用して署名および暗号化される。署名と暗号化の両方が MSH(電文処理機能）に必要とされる場合は、最初に署名し、それから暗号化する。ebXML 電文搬送サービス電文の署名および暗号化は、以下のステップに従う。

1. 電文の署名は、先の 6.2 項で説明されているステップに従う。
2. 電文の暗号化は、先の 6.4 項で説明されているステップに従う。サブ要素 EncryptedKey は、電文に署名をした際に生成された、既存の要素 Security の先頭に追加される。

電子的に署名および暗号化された ebXML 電文の例

```
Mime-Version: 1.0
Content-Type: text/xml
Content-Transfer-Encoding: binary
SOAPAction: ""
Content-Length: 7205

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-3_0.xsd">
        <eb:Message eb:version="3.0" id="ebMessage" soap:mustUnderstand="1">
            <eb:UserMessage eb:syncresp="false">
                <eb:MessageInfo>
                    <eb:Timestamp>2005-10-31T17:36:20.656Z</eb:Timestamp>
                    <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>
                    <eb:RefToMessageId>UUID-1@msh-
server.example.com</eb:RefToMessageId>
                </eb:MessageInfo>
                <eb:PartyInfo>
                    <eb:From>
                        <eb:PartyId>uri:msh-server.example.com</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
                    </eb:From>
                    <eb:To>
                        <eb:PartyId eb:type="someType">QRS543</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
```

```
                    </eb:To>
                </eb:PartyInfo>
                <eb:CollaborationInfo>
                    <eb:AgreementRef>http://msh-
server.example.com/cpa/123456</eb:AgreementRef>
                    <eb:Service eb:type="someType">QuoteToCollect</eb:Service>
                    <eb:Action>NewPurchaseOrder</eb:Action>
                    <eb:ConversationId>2a81ffbd-0d3d-4cbd-8601-
d916e0ed2fe2</eb:ConversationId>
                </eb:CollaborationInfo>
                <eb:MessageProperties>
                    <eb:Property
name="ProcessInst">PurchaseOrder:123456</eb:Property>
                    <eb:Property name="ContextID">987654321</eb:Property>
                </eb:MessageProperties>
                <eb:PayloadInfo>
                    <eb:Partref eb:id="ID7016111307801806 59"
                        eb:idref="cid:PO_Image">
                        <eb:Description xml:lang="en-US">PO Image</eb:Description>
                    </eb:Partref>
                </eb:PayloadInfo>
            </eb:UserMessage>
        </eb:Message>
        <wsse:Security soap:mustUnderstand="1"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="signingCert">...</wsse:BinarySecurityToken>
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="encryptionCert">...</wsse:BinarySecurityToken>
            <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
                <enc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
                    xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd"/>
                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                    <wsse:SecurityTokenReference>
                        <wsse:Reference URI="#encryptionCert"/>
                    </wsse:SecurityTokenReference>
                </KeyInfo>
                <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <CipherValue>F3HmZ2Ldyn0umLCx/8Q9B9e8OoslJx9i9hOWQjh6JJwYqDLbd
g0QVFiVT1LVjazlThS9m9rkRtpkhCUIY1xjFKtDsuIIAW8cLZv7IHkVoDtQ7ihJc8hYIlEESX9qZN65Jgy
Aa3BYgW9ipjGHtNgZ9RzUdzKdeY74DFm27R6m8b0=</CipherValue>
                </CipherData>
                <ReferenceList xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <DataReference URI="#enc"/>
                </ReferenceList>
            </enc:EncryptedKey>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#ebMessage">
                        <ds:Transforms>
                            <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>Ae0PLUKJUnUyAMXkLQD/WwKiFiI=</ds:DigestVal
ue>
                    </ds:Reference>
```

```
                    <ds:Reference URI="#body">
                        <ds:Transforms>
                            <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>kNY6X7LnRTwxXXBzSw07tcA0KSU=</ds:DigestVal
ue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>
                    T24okA0MUh5iBNMG6tk8QAKZ+lFMmY1rcPnkOr9j3fHRGM2qqUnoBydOTnClcE
MzPZbnlhdN
                    YZYmab1lqa4N5ynLjwlM4kp0uMip9hapijwL67aBnUeHiFmUau0x9DBOdKZTVa
1QQ92106ge
                    j2YPDt3VKIlLLT2c8O4TfayGvuY= </ds:SignatureValue>
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference
                        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">
                        <wsse:Reference URI="#signingCert"/>
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
            </ds:Signature>
        </wsse:Security>
    </soap:Header>
    <soap:Body wsu:Id="body"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
        <EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
            xmlns="http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
            <CipherData>
                <CipherValue>tjOgUPMmQwd6hXiHuvl42swqv4dTYiBfmg8u1SuFVRC3yfNlokshv
oxs1/qQoqN1prDiSOxsxsFvg1la7dehjMWb0owuvU2de1eKr5KPcSApnG+kTvNrtg==</CipherValue>
            </CipherData>
        </EncryptedData>
    </soap:Body>
</soap:Envelope>
```

6.7　添付ファイル付き SOAP 電文の署名と暗号化

ebXML 電文搬送サービス電文は、以下のステップに従い、Web サービスセキュリティを使用して署名および暗号化される。

署名と暗号化の両方が MSH(電文処理機能) に必要とされる場合は、最初に署名し、それから暗号化する。

1. 電文の署名は、先の 6.3 項で説明されているステップに従う。
2. 電文の暗号化は、先の 6.5 項で説明されているステップに従う。サブ要素 EncryptedKey は、電文に署名した際に生成された、既存の要素 Security の先頭に追加される。

電子的に署名および暗号化された、ebXML の添付ファイルを伴う SOAP 電文の例

```
Mime-Version: 1.0
Content-Type: multipart/related; type="text/xml";  boundary="----
=_Part_2_6825397.1130520599536"
SOAPAction: ""
Content-Length: 7860

------=_Part_2_6825397.1130520599536
Content-Type: text/xml
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-3_0.xsd">
        <eb:Message eb:version="3.0" id="ebMessage" soap:mustUnderstand="1">
            <eb:UserMessage eb:syncresp="false">
```

```
                <eb:MessageInfo>
                    <eb:Timestamp>2005-10-28T17:29:59.119Z</eb:Timestamp>
                    <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>
                    <eb:RefToMessageId>UUID-1@msh-
server.example.com</eb:RefToMessageId>
                </eb:MessageInfo>
                <eb:PartyInfo>
                    <eb:From>
                        <eb:PartyId>uri:msh-server.example.com</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
                    </eb:From>
                    <eb:To>
                        <eb:PartyId eb:type="someType">QRS543</eb:PartyId>
                        <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
                    </eb:To>
                </eb:PartyInfo>
                <eb:CollaborationInfo>
                    <eb:AgreementRef>http://msh-
server.example.com/cpa/123456</eb:AgreementRef>
                    <eb:Service eb:type="someType">QuoteToCollect</eb:Service>
                    <eb:Action>NewPurchaseOrder</eb:Action>
                    <eb:ConversationId>782a5c5a-9dad-4cd9-9bbe-
94c0d737f22b</eb:ConversationId>
                </eb:CollaborationInfo>
                <eb:MessageProperties>
                    <eb:Property
name="ProcessInst">PurchaseOrder:123456</eb:Property>
                    <eb:Property name="ContextID">987654321</eb:Property>
                </eb:MessageProperties>
                <eb:PayloadInfo>
                    <eb:Partref eb:id="ID:8587580113052059916 4"
                        eb:idref="cid:PO_Image">
                        <eb:Description xml:lang="en-US">PO Image</eb:Description>
                    </eb:Partref>
                </eb:PayloadInfo>
            </eb:UserMessage>
        </eb:Message>
        <wsse:Security soap:mustUnderstand="1"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="signingCert">...</wsse:BinarySecurityToken>
            <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
                wsu:Id="encryptionCert">...</wsse:BinarySecurityToken>
            <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
                <enc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
                    xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd"/>
                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                    <wsse:SecurityTokenReference>
                        <wsse:Reference URI="#encryptionCert"/>
                    </wsse:SecurityTokenReference>
                </KeyInfo>
                <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <CipherValue>jJRbQBjzYpfdCkPk5F7jUoFjw6Ls6DQ8D9sdI62fwjW9Um/g9
QfivLeVzvSndgnthfEBC1Z6loKiuEF5/Ztw/tFrRgkboR7EBG5XaJUnt0rt8iCChy4PfxCEhH1KjFgTJhU
bXxNW3FxSLkouCn2qIBDrJqwZXAIistt29JrANCc=</CipherValue>
                </CipherData>
                <ReferenceList xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <DataReference URI="#encrypted-attachment"/>
                </ReferenceList>
            </enc:EncryptedKey>
            <EncryptedData Id="encrypted-attachment" MimeType="image/jpeg"
```

```
                Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
profile-1.0#Attachment-Content-Only"
                xmlns="http://www.w3.org/2001/04/xmlenc#">
                <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
                <CipherData>
                    <CipherReference URI="cid:PO_Image">
                        <Transforms>
                            <Transform
                                Algorithm="http://docs.oasis-
open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-Only-
Transform"
                                xmlns="http://www.w3.org/2000/09/xmldsig#"/>
                        </Transforms>
                    </CipherReference>
                </CipherData>
            </EncryptedData>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#ebMessage">
                        <ds:Transforms>
                            <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>xUISuIg5eVxy3FL/4yCrZoEZrTM=</ds:DigestVal
ue>
                    </ds:Reference>
                    <ds:Reference URI="cid:PO_Image">
                        <ds:Transforms>
                            <ds:Transform
                                Algorithm="http://docs.oasis-
open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-Only-
Transform"
                            />
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>R4hCV4K4I5QZdSsrP4KrLu46hFo=</ds:DigestVal
ue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>
                    BGnJV/b7EUbAEsn7GmNhZ8yYN6ZoO6uz29E5r9GHxDW+MUH4wksgA654w+sB0r
Wl8xNranag
                    3dhKoHbaRERzYHDGq1VfIRqgEwOrHwhz4h7uoLX4yxOU6G9T/gily67Q3pENGp
mVowzoppHm
                    /yd/A2T0+v4vso20aJiSieEIzSQ= </ds:SignatureValue>
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference
                        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">
                        <wsse:Reference URI="#signingCert"/>
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
            </ds:Signature>
        </wsse:Security>
    </soap:Header>
    <soap:Body/>
</soap:Envelope>

------=_Part_2_6825397.1130520599536
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64
Content-Id: <PO_Image>
Content-Description: WSS XML Encryption message; type="image/jpeg"

VEhmwb4FHFhqQH8m5PKqVu8H0/bq2yUF

------=_Part_2_6825397.1130520599536--
```

6.8　Pull 要求シグナルの安全性確保

6.8.1 認証

送信 MSH(電文処理機能）は、Pull 要求を送信する受信 MSH(電文処理機能）を認証できなければならない。この機能の使用は任意だが、この機能のサポートは MSH(電文処理機能）の要件である。認証は、この MSH(電文処理機能）を使用している可能性がある当事者に依存しない方法で達成するのが望ましい。言い換えると、この MSH(電文処理機能）を使用している当事者特有の安全性の要件に関係なく、受信 MSH(電文処理機能） を認証できなければならない。例えば、これらの当事者が電文交換に安全性を何も使用しないと決めても（これは、CPA またはその他の同意形式によって示される)、受信 MSH(電文処理機能）は、Pull による電文搬送を行う際、または、他の MSH(電文処理機能） レベルのシグナルを送信する際に認証される必要がある。

特定の受信 MSH(電文処理機能）に認証が必要な場合は、送信 MSH(電文処理機能）が SOAP のプロトコルレベルで安全性を使用するのが望ましい。受信 MSH(電文処理機能）が SOAP レベルの安全性を使用できない場合は、HTTP の基本アクセス認証スキーマ など、パスワードに基づいた認証を使用できる。この場合、安全な通信プロトコルを使用しなければならない（例えば、TLS、SSL)。

6.8.1.1 ユーザー名/ パスワードのオプション

Web サービスセキュリティおよびXML のデジタル署名の管理が完全にサポートされていないような制約された環境では、最も簡潔で可能なメカニズム、つまり、ユーザー名とパスワードの組み合わせに基づく認証の代替物がサポートされなければならない。Pull 要求シグナルのユーザー名およびパスワードは、Web サービスセキュリティのユーザー名/パスワードのトークンを使用しなければならない。

上記のように認証される Pull 要求シグナルの例

```
<wsse:Security xmlns:wsse="…" xmlns:wsu="…">
    <wsse:UsernameToken wsu:Id="#TokenID">
        <wsse:Username>hamid</wsse:Username>
        <wsse:Password>SomePassword</wsse:Password>
    </wsse:UsernameToken>
</wsse:Security>
<eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >
    <eb:SignalMessage>
        <wsse:SecurityTokenReference>
            <wsse:Reference URI="#TokenID" />
        </wsse:SecurityTokenReference>
        <eb:MessageInfo> …. </eb:MessageInfo>
        <eb:PullRequest eb:pipeId="…" />
    </eb: SignalMessage>
</eb:Message>
```

注：

- 属性/wsse:Security/wsse:UsernameToken/wsse:Password/@Type は、型テキストのパスワードのデフォルトになる場合は存在しない。ユーザーは、ダイジェスト型のパスワードの処理を要求されることはない。
- 要素 wsse:UserName の値は、実装の問題である。「ユーザー」は、MSH(電文処理機能）自体を表す場合もあれば、MSH(電文処理機能）を使用している当事者を表す場合もある。後者の場合、このユーザー名は eb:From/PartyId の値と同一でなくてもよい。

6.8.1.2 BinaryToken のオプション

このオプションには、より完全な WEB サービスセキュリティ（WSS）の実装が必要である。PullRequest シグナルの構文も同じである。つまり、要素<wsse:SecurityTokenReference>は、要素<eb:SignalMessage> の中に取り入れられる。以下は、X509 証明を使用したサンプルリストである。

```
<wsse:Security soap:mustUnderstand="1"
               xmlns:wsse="…"
               xmlns:wsu="…">
    <wsse:BinarySecurityToken wsu:Id="signingCert"

               EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
               ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">
               ...
    </wsse:BinarySecurityToken>
        <ds:Signature xmlns:ds="…"> … </ds:Signature>
    </wsse:Security>
<eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >
    <eb:SignalMessage>
        <wsse:SecurityTokenReference>
            <wsse:Reference URI="#signingCert" />
        </wsse:SecurityTokenReference>
        <eb:MessageInfo> …. </eb:MessageInfo>
        <eb:PullRequest eb:pipeId="…" />
    </eb: SignalMessage>
</eb:Message>
```

6.8.2 権限付与

送信 MSH(電文処理機能）によって受信される pullRequest シグナルの処理は、これらが電文搬送の開始を許可されるパイプとともに、送信 MSH(電文処理機能）が管理し、事前許可された受信 MSH(電文処理機能）に関する終点情報のリストと関連のある内部情報に基づいて権限を付与される。

6.8.3 リピート攻撃（Repeat Attacks）の予防

悪質な複製を行ったり、PullRequest シグナルを再利用したりすると、シグナル電文の有効な要求に対してではなく、権限のない宛て先にユーザー電文が搬送される可能性がある。この攻撃を防ぐには、(1) 複製の排除が PullRequest の複製を排除できるように最大 1 回 (At-Most-Once) の信頼性を使用するか、(2) WEB サービスセキュリティに従って信頼性のヘッダーの統合を適切に実行するのが望ましい。

6.9 対抗策の技術

6.9.1 永続的なデジタル署名

ebXML 電文搬送サービス電文（ebXML の SOAP ヘッダーとボディ、および、関連するペイロードオブジェクト）に電子的に署名を行う目的に適用できる唯一の技術は、Web サービスセキュリティおよび Web サービスセキュリティの添付ファイルを伴う SOAP 電文プロファイルに適合する技術によって提供される。これらの仕様に適合する XML 署名は、XML 文を部分選択して署名することができるので、署名の妥当性を維持しつつ、文の内容を増やすことができる（新し

い要素内容を追加できる)。

ebXML 電文搬送サービス電文に電子的に署名を行うのに署名が使用される場合、Web サービスセキュリティおよび Web サービスセキュリティの添付ファイルを伴う SOAP 電文プロファイルは、ebXML の SOAP ヘッダーとボディを ebXML のペイロードコンテナまたは電文と関連のある Web 上のどこかにあるデータに結びつけるために使用されなければならない。

デジタル署名を必要とする ebXML 電文搬送サービス電文は、本項に定義されているプロセスに従って署名され、Web サービスセキュリティ および Web サービスセキュリティの添付ファイルを伴う SOAP 電文プロファイル と完全に適合する。

6.9.2 永続的な署名済みの受信

電子的に署名された ebXML 電文搬送サービス電文は、それ自体が電子的に署名される（前項で説明されている方法で）確認電文によって承認される場合がある。確認電文は、元の電文の Web サービスセキュリティ の署名要素に包含されているリストと適合する Web サービスセキュリティ の参照要素リストを包含しなければならない。

6.9.3 非永続的な認証

非永続的な認証は、ebXML 電文搬送サービス電文のトランスポートに使用される通信チャンネルによって行われる。この認証は、一方向か双方向のいずれかである。具体的な方法は、使用された通信プロトコルによって決定される。例えば、TLS [RFC2246]や IPSec [RFC2402]などの安全なネットワークプロトコルを使用すると、ebXML 電文搬送サービス電文の送信者は TCP/IP 環境の方向を認証できるようになる。

6.9.4 非永続的な整合性

TLS [RFC2246] や IPSec [RFC2402] などの安全なネットワークプロトコルは、ネットワーク接続を介して送信されるパケットのダイジェストおよび比較を規定するために構成される場合がある。

6.9.5 永続的な機密性

永続的な機密性は、Web サービスセキュリティおよび Web サービスセキュリティの添付ファイルを伴う SOAP 電文プロファイル に適合する技術によって提供される。これらの仕様に適合する暗号化は、SOAP ヘッダーを含む ebXML 電文搬送サービス電文内にある要素の永続的で選択的な機密性を提供できる。

6.9.6 非永続的な機密性

TLS [RFC2246] や IPSEC [RFC2402] などの安全なネットワークプロトコルは、2 つ ebXML に隣接した MSH(電文処理機能）ノード間で電文が搬送される際、その電文の一時的な機密性を提供する。

6.9.7 永続的な権限付与

永続的な権限付与は、Web サービスセキュリティの SAML トークンファイルを使用して提供される。

6.9.8 非永続的な権限付与

TLS [RFC2246] や IPSEC [RFC2402] などの安全なネットワークプロトコルは、セッションを確立する前に、証明書の双方向の認証を規定するために構成される場合がある。これは、ebXML

の MSH（電文処理機能）が接続のソースを認証し、そのソースを権限のある ebXML 電文搬送サービス電文のソースとして認識する能力を規定する。

6.10 安全性の考慮

実装者は、ebXML 電文搬送サービス 電文の整合性および元々の形を守るために Web サービセキュリティが使用される場合でも、脆弱性が存在することに注意しなければならない。脆弱性の重要性は、配置された環境および ebXML 電文搬送サービス電文の交換に使用されたトランスポートに必然的に依存する。

脆弱性が存在するのは、ebXML 電文搬送が XML と MIME の技術両方を統合したものだからである。2 つ以上の技術を結合する場合は、常に追加の（場合によっては一意の）安全性の問題を処理する必要がある。この場合、MIME は、SOAP エンベロープおよびすべてのペイロードコンテナを包含している電文パッケージのフレームワークとして使用される。SOAP エンベロープのさまざまな要素は、MIME のメカニズムを介して識別されたペイロードを参照する。さらに、さまざまなラベルは、SOAP エンベロープと MIME フレームワークの両方で複製される。例えば、ペイロードの内容の型などである。問題は、これらのすべての情報がどのようにしていつ使用されるかである。

具体的には、MIME Content-ID: ヘッダーは、それぞれのペイロードに対する一意の識別ラベルを特定するために使用される。ラベルは、ペイロードが必要とされる時に必ずそのペイロードを識別するために、SOAP エンベロープで使用される。MIME Content-Type: ヘッダーは、ペイロードに伝えられた内容の型を識別するために使用され、内容型の中には、実際の型にさらに適合させる役割を果たす追加のパラメータを含むものもある。この情報は SOAP エンベロープ内で利用できる。

デジタル署名に基づく Web サービスセキュリティが適用されても、あるいは、暗号化に基づく Web サービスセキュリティが適用されても、MIME ヘッダーは保護されない。したがって、SOAP エンベロープの情報と照らして MIME ヘッダーの情報がどのように処理されるかによって、ebXML 電文搬送サービス電文がリスクにさらされる場合がある。

Content-ID: MIME ヘッダーは重要である。攻撃者は、ペイロードと Content-ID: ヘッダーを混合および適合させることにより、容易にサービス拒絶攻撃（denial-of-service attack）を開始できる。多くのサービス拒絶攻撃同様、この脆弱性に特定の予防はない。しかし、デジタル署名が有効な場合、実際のペイロードに対して計算されたダイジェストが SOAP エンベロープに含まれているダイジェストと一致しないので、この攻撃は検出される。

内容型が MIME ヘッダーと SOAP エンベロープの両方に存在するのは問題である。通常、安全性の習慣として、2 つの場所に情報を複製することは推奨されない。情報が複製された場合、通常の安全性を実行するには、両方の場所にある情報を比較してそれらが等しいことを確認する必要がある。両方の情報集合が一致しなかった場合は、安全性の侵害だと見なされる。

電文が送信元から送信先へ移動している間、攻撃者は MIME ヘッダーを変更できる。そして、安全性サービスが有効の場合、この変更は検出されない。この脅威は、マルチホップのトランスポート環境と比較するとピアツーピアのトランスポート環境ではあまり問題にならない。ebXML

電文搬送サービス電文が長期のストレージ領域に記憶されている場合、その領域の潜在的脅威が電文を修正のリスクにさらすので、すべての実装がリスクにさらされる。

実際のリスクは、複製された情報の集合それぞれを実装がどのように使用するかに拠る。ボディ部分を識別および区別する MIME 構文解析を越えた処理が MIME ヘッダーの情報に左右される場合、実装は、意図されない、または、望ましくないアクションを取る方向に向かうリスクにさらされる。これがどのように不正利用されるかは、バッファのオーバーフローを許す通常のプログラミングミスと比較するのがよい。つまりこれは、攻撃者の創造性と永続性に左右される。

したがって、実装は、MIME ヘッダーにある保護されていない情報が、ボディ部分の識別と区別のみを目的とした MIME パーサー以外によって決して使用されないことを確実にすることにより、リスクを減らすことができる。このバージョンの仕様は、実装が複製の情報集合を対照すべきかどうかについても、その対照の結果に基づいてどのアクションを取るべきかについても推奨はしない。

7　エラー処理モジュール

エラー処理は、処理の信頼性や安全性などの比較的独立した SOAP モジュールを含む MSH(電文処理機能）の構造の性質を考慮しなければならない。エラー報告は、通常の電文の交換と同じ接続性の制約にも左右される。これは、より包括的なエラーモデルを要求する。エラーを報告する他の方法に関して、このモデルは、同意と関連するものと、不変の相互運用性の要件と関連するものとの明確な区別を考慮しなければならない。

エラー生成とエラー報告は、ここでは直交する概念として扱われる。エラー処理のある側面（特にエラー報告）は、例えば、同意から生じるエラー処理の動作コンテキスト（OpCtx-ErrorHandling）に最もよく規定されている。

7.1　用語

・ フォルト：フォルトは常に SOAP Fault を意味する。これは、SOAP の仕様に従って生成および処理されなければならない。

・ エラー：SOAP Fault ではないエラー。これは、規定のモジュール（ebXML 電文搬送サービスモジュール、信頼性モジュール、安全性モジュール）の 1 つに現れる。

・ ebXML 電文搬送サービスエラー：これはエラーの特定のケースで、この仕様に準拠している ebXML 電文搬送サービスモジュールによって生成される。

・ 信頼性エラー：これはエラーの特定のケースで、信頼性モジュールによって生成される。

・ 安全性エラー：これはエラーの特定のケースで、安全性モジュールによって生成される。

・ エスカレートされた ebXML 電文搬送サービスエラー：これは、ebXML 電文搬送サービスモジュール以外のモジュール（つまり、安全性モジュールまたは信頼性モジュール）で発生する ebXML 電文搬送サービスエラーである。

・ebXML 電文搬送サービスエラー生成:ある失敗または警告状態に基づいて ebXML 電文搬送サービスエラーオブジェクトを生成する動作。

・ebXML 電文搬送サービスエラー報告 : ebXML 電文搬送サービスエラーオブジェクトを他のエンティティに伝える動作。

・ 誤った電文：ある種のエラーを生じさせている欠陥のある電文。

7.2　電文搬送サービスエラー

7.2.1　要素 eb:Error

ebXML 電文搬送サービスエラーは、報告される方法に関わらず、eb:Error XML インフォセットによって表される。MSH(電文処理機能）によって生じたそれぞれのエラーは、以下の属性を持つ。

・origin（必須の属性）

・category（任意の属性）

・errorCode（任意の属性）

・severity（任意の属性）

・shortDescription（任意の属性）
・Description（任意の属性）
・errorDetail（任意の属性）
例：

```
<eb:Error eb:origin="ebMS" eb:category="Unpackaging"
          eb:errorCode="eb_InvalidHeader" eb:severity="fatal">
    <eb:Description> … </eb:Description>

</eb:Error>
```

7.2.2 属性: eb:Error/@origin

この必須の属性は、エラーが生じた機能的なモジュールを識別する。このモジュールとなり得るのは、信頼性モジュール、安全性モジュール、ebXML 電文搬送サービスモジュール、または、アドレス指定モジュールである。この属性が取り得る値は、ebXML MS、security、 reliability、addressing である。

7.2.3 属性: eb:Error/@category

この任意の属性は、特定の発生元と関連のあるエラーの型を識別する。例えば、Content、Packaging、UnPackaging、Communication、InternalProcess などである。

7.2.4 属性: eb:Error/@errorCode

この必須の属性は、エラーの型に対する一意の識別子である。

7.2.5 属性: eb:Error/@severity

この任意の属性は、エラーの重大度を示す。有効な値は、warning および failure である。warning の値は、潜在的な障害状況は検出されたが、電文の処理も交換も今のところ失敗していないことを表す。特に、電文が利用者に配信されることになっていた場合、たとえ warning が発せられても電文は配信される。この問題があっても、会話または MEP にある他の関連電文を生成したり交換したりできる。

failure の値は、電文処理が予期されたとおりに行われず、成功とは見なされないことを表す。それにもかかわらず電文のペイロードが配信される状態にある場合、同意の状態でなければデフォルトの振る舞いはそれを配信しない（OpCtx-ErrorHandling 参照）。電文が関与する会話または MEP のインスタンスが無効になるリスクにさらされていても、このエラーは MSH(電文処理機能）が他の電文を処理する能力を推測しない。

7.2.6 属性: eb:Error/shortDescription

この任意の要素は、可読性を促進するように、ログに報告されるエラーについて簡単に説明する。

7.2.7 要素: eb:Error/Description

この任意の要素は、属性 xml:lang によって定義された言語におけるエラーを叙述的に説明する。この要素の内容は、実装特有の決定に委ねられる。

7.2.8 要素: eb:Error/ErrorDetail

この任意の要素は、エラーが生じた内容について詳細に説明する。例としては、例外のトレー

スが挙げられる。

7.3　ebXML 電文搬送サービスエラー電文

電文として報告される場合、ebXML 電文搬送サービスエラーは、ebXML 電文搬送サービスシグナル電文としてパッケージ化される。eb:SignalMessage の下に複数の要素 eb:Error が存在する場合がある。このような場合、eb:RefToMessageId が eb:SignalMessage/eb:MessageInfo の子要素として存在するなら、すべての要素 eb:Error は、eb:RefToMessageId によって識別される ebXML 電文搬送サービス電文（誤った電文）と関連がなければならない。

要素 eb:SignalMessage/eb:MessageInfo が eb:RefToMessageId を包含しない場合、要素 eb:Error は、特定の ebXML 電文搬送サービス電文と関連があってはならない。

ebXML エラー電文の例：

```
<SOAP:Header …>
```

```
  <eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >

     <eb:SignalMessage>

        <eb:MessageInfo>
           <eb:timestamp>2000-07-25T12:19:05</eb:timestamp>
           <eb:MessageId>UUID-2@example.com</eb:MessageId>
           <eb:RefToMessageId>UUID-1@example.com</eb:RefToMessageId>
        </eb:MessageInfo>

        <eb:Error eb:origin="Security" category="Processing"
                  eb:errorCode="FailedAuthentication"
                  eb:severity="failure">
                 <eb:Description>Validation of signature
failed</eb:Description>
        </eb:Error>
        <eb:Error eb:origin="ebMS" category="Communication"
                  eb:errorCode="EmptyPipe" eb:severity="warning">
        <eb:Description>PullRequest done on an empty
pipe</eb:Description>

     </eb:SignalMessage>

  </eb:Message>

</SOAP:Header>
```

7.4　エラーモジュールの拡張性

7.4.1　新しい ebXML 電文搬送サービスエラーの追加

属性 errorCode（eb:Message/eb:SignalMessage/eb:Error/@errorCode）は、MSH(電文処理機能）の適用範囲内で一意の識別子でなければならない。ここで規定されているもの以外の新しい ebXML 電文搬送サービスエラーは、新しい errorCode の値を生成することによって追加される。属性 errorCode の値は、5 文字の “EBMS:”から始まらなければならない。

7.5　ebXML 電文搬送サービスエラーの生成

この仕様は、ebXML 電文搬送サービスエラーが生成される状況だけではなく、重要な ebXML 電文搬送サービスエラーも識別する。エラーを生じさせるこれらの状況の中には、信頼性および安全性モジュールによって生成された失敗かエラーのいずれかの ebXML 電文搬送サービスエラーとしてのエスカレーションを含むものもある。これらのモジュールは、エラーを生じさせる MSH(電文処理機能）に含まれる場合もあれば、エラーを生じさせている MSH(電文処理機能）と通信を取っているリモートの MSH(電文処理機能）に含まれる場合もある。本仕様に定義されているいくつかの場合を除き、エラーのエスカレーションの方針は、MSH(電文処理機能）の動作コンテキスト（OpCtx-ErrorHandling）に表現されているユーザー間の同意に委ねられる。

7.6 エラーの報告

エラー報告には主に以下の 3 つの方法がある。

・ フォルト送信による報告：ある型の ebXML 電文搬送サービスエラーが生じる場合、MSH(電文処理機能）は SOAP フォルトを生成できる。他の報告アクションが成功している場合にフォルト報告のアクションを使用するのは望ましくない。

・ 通知による報告：MSH(電文処理機能）からあるエンティティ（電文生成者、利用者、または他のエンティティ。ローカルの場合もリモートの場合もある。）に送られるエラー情報の帯域外（out-of-band）の搬送。電文生成者または利用者に電文を通知する場合、このような報告アクションは、電文搬送モデルにある「通知」動作によって抽象化される。

・ エラー電文：MSH(電文処理機能）から他の MSH(電文処理機能）に送信される ebXML 電文搬送サービスシグナル電文で、少なくとも 1 つの要素 eb:Error を包含する。このような報告アクションは、そのような電文上にある送信および受信の抽象動作によってモデル化される。

送信 MSH(電文処理機能）に生じる報告エラーにおけるさまざまなオプションの例：

提出された電文がパッケージ化される前に検出されたエラーは、通常、ローカルで電文生成者に通知され、宛て先の MSH(電文処理機能）には報告されない。しかし、この電文が、受信側で完了を待っている状態を保持している大規模な交換の一部だった場合、望ましい方針は、誤った電文も（エラー電文を使用して）受信 MSH(電文処理機能）に報告されることである。Pull モードの搬送が確立されて、受信 MSH(電文処理機能）がその電文を PullRequest への応答として捉える場合、このような ebXML 電文搬送サービスエラーは同じ方法で プルされる。Push モードが送信者から受信者の間で活性状態の場合、送信者側で生成されるエラーは通常の電文同様にプッシュされる。また、それらが Pull モードで別の電文パイプに提出されることも決定されるので、受信 MSH(電文処理機能）には要求に応じてのみ、そのような送信側のエラーを入手するオプションがある。

受信 MSH(電文処理機能）に生じる報告エラーにおけるさまざまなオプションの例：

受信 MSH(電文処理機能）が受信した電文にエラーを検出した場合、そのようなエラーを処理する当事者のコンテキストや能力によって報告の方針は変わる。例えば、エラーを生じさせる受信 MSH(電文処理機能）は、それ自身の利用者当事者だけに通知する場合もあれば、エラー電文を送信 MSH(電文処理機能）に返信する場合もあれば、その両方の場合もある。これらすべての場合に通常共通な要件は、何らかの方法でエラーが報告され、それが要素 eb:Error の構造に適合

することである。

7.7 標準的な ebXML 電文搬送サービスエラー

7.7.1 ebXML 電文搬送サービス処理エラー

以下の表は、ebXML 電文搬送サービスモジュール自体（エスカレートされたエラーではない ebXML 電文搬送サービスエラー）に発生する可能性があるエラー、つまり、@origin=”ebms”を伴うエラーについて説明している。これらのエラーは、MSH(電文処理機能）に報告される際に MSH(電文処理機能）によって適切に生成または理解されるように、MSH(電文処理機能）によってサポートされなければならない。

エラーコード	簡単な説明	推奨される重大度	カテゴリ値	説明または意味
EBMS:0001	ValueNotRecognized	failure	Content	電文は適切に作成され、スキーマは有効だが、いくつかの要素/属性は、認識されず、したがって MSH に使用されない値を包含する。
EBMS:0002	FeatureNotSupported	warning	Content	電文は適切に作成され、スキーマは有効だが、いくつかの要素/属性値は、予期されたとおりに処理されない。これは、関連のある失敗が MSH によってサポートされていないからである。
EBMS:0003	ValueInconsistent	failure	Content	電文は適切に作成され、スキーマは有効だが、いくつかの要素/属性値は、他の要素/属性の内容、または、MSH の動作内容、または、ebMS 仕様の規範的要件のい

				ずれかと一致しない。
EBMS:0004	Other	failure	Content	
EBMS:0005	ConnectionFailure	failure	Communication	リモート MSH でトランスポート接続を開こうとする際、MSH は一時的または永久的な失敗を経験している。
EBMS:0006	EmptyPipe	warning	Communication	この段階では、この電文パイプにプルするための電文はない。
EBMS:0007	MimeInconsistency	failure	Unpackaging	MIME の使用は、本仕様で必要とされる用途と一致しない。
EBMS:0008	FeatureNotSupported	failure	Unpackaging	電文は適切に作成され、スキーマは有効だが、ある要素/属性が存在するか否かは、MSH の能力と一致しない。
EBMS:0009	InvalidHeader	failure	Unpackaging	ebXML 電文搬送サービスヘッダーが、XML 文として適切に形成されていないか、あるいは、ebXML 電文搬送サービスのパッキングルールに適合しない。

7.7.2 安全性の処理エラー

以下の表は、安全性モジュール内で発生するエラー、つまり、@origin="security"を伴うエラーについて説明している。これらのエラーは、MSH(電文処理機能) に報告される際に適切に生成または理解されるように、MSH(電文処理機能) によってエスカレートされなければならない。

エラーコード	簡単な説明	推奨され	カテゴリ値	説明または意味

		る重大度		
EBMS:0011	FailedAuthentication	failure	Processing	ebms SOAP アクタ用の安全性ヘッダーにある署名は、安全性モジュールによって検証されない。

7.7.3 信頼性エラー

以下の表は、信頼性モジュール内で発生するエラー、つまり、@origin="reliability"を伴うについて説明している。これらのエラーは、MSH(電文処理機能）に報告される際に適切に生成または理解されるように、MSH(電文処理機能）によってエスカレートされなければならない。

エラーコード	簡単な説明	推奨される重大度	カテゴリ値	説明または意味
EBMS:0010	DeliveryFailure	failure	communication	電文は、生成された配信要件の下で送信されたが、信頼性モジュールは、何度も再送したにもかかわらず、電文が適切に配信されたかどうか確信がない。

8　信頼性電文搬送モジュール

8.1　信頼性電文搬送モデル

本項では、ebXML 電文搬送サービス電文に対する信頼性電文搬送モデルについて説明する。信頼性のモデルは、信頼性の合意の形で表現される。これらの合意自体は、ebXML 電文搬送サービス電文処理に関与する構成要素間における電文データの搬送を表す抽象的な動作を使用して表現される。本質的に、このモデルは、相互運用可能な MSH（電文処理機能）を開発する必要のある実装者に十分な手引きを提供することはできない。このモデルは、信頼性が確保された既存の電文搬送標準に基づくインスタンスによって補足されなければならない。このようなインスタンスをここでは信頼性モジュールと呼ぶ。

ebXML 電文搬送サービス V3.0 の基本設計原則は、主な電文搬送 QoS の機能をモジュール化する。つまり、電文処理の他の側面への介入がないので、関心分野にある既存の標準を MSH（電文処理機能）が信頼できるだけではなく、そのような標準の実装を修正せずに、あるいは、わずかに修正するだけで再利用できる。

信頼性の機能は、ebms ヘッダーとは別に処理される。この処理は、別の SOAP ノードとして行動しているモジュールによって実行されると抽象的に定義され、信頼性電文搬送プロセッサ（RMP）と呼ばれる。ebXML 電文搬送サービス電文の信頼性は、ebms ヘッダーとは区別される SOAP ヘッダーの拡張（ここでは「信頼性ヘッダー」と呼ばれる）によってサポートされる。ebXML 電文搬送サービス電文の処理は以下を含む。

・ebms ヘッダーを処理する機能

・信頼性電文搬送プロセッサ（RMP）によって実装される際の信頼性の機能。これは信頼性ヘッダーを処理する機能

・ 本仕様でも必要とされる、SOAP ヘッダーを処理する他の機能（安全性など）

図 9 と図 10 に示されているように、信頼性 MEP を実行する際、信頼性のモデルには、異なる役割を演じている RMP のインスタンスが 2 つ必要である。これらのインスタンスは、開始者 RMP（開始 MSH と関連がある）と応答者 RMP（応答 MSH と関連がある）である。注意しなければならないのは、ユーザー電文が交換されるたびに変化する送信および受信の役割とは対照的に、これらの役割はシンプルな ebXML 電文搬送サービスの MEP インスタンスを実行しても変わらない点である。これは、例えば、確実に要求電文を送信するために必要な機能を開始者が予測し、かつ、応答がある場合（2.1.1 項の「電文搬送モデル」に定義されているように、引き続き送信そして次に受信の役割を担う場合）、その応答を受信することを意味する。

5 つの抽象的な動作（RM-Submit、RM-Deliver、RM-SubmitResponse、RM-DeliverResponse、RMNotify）は、RMP の抽象的なインターフェースを表す。これらは、RMP と MSH（電文処理機能）の他の構成要素（RM-Producer か the RMConsumer）間の電文データか通知データのいずれかを搬送する。本項では、「確実に送信された」という表現は、送信が信頼性の合意に左右される場合があることを意味する（8.2.1 項参照）。

抽象的な RM 動作は、以下のように定義されている。

・RM-Submit

RM-Producer から開始 RMP に SOAP 電文を搬送する抽象的な動作なので、この電文は確実に送信される。

・RM-Deliver

応答 RMP からその RMConsumer に SOAP 電文を搬送する抽象的な動作なので、この電文からのペイロードは後に MSH(電文処理機能）によって配信される。

・RM-SubmitResponse

前に受信した信頼性電文への応答として RM-Producer から応答 RMP に SOAP 電文を搬送する抽象的な動作。この応答は、前に信頼性電文を伝えたのと同じ SOAP 要求応答 MEP インスタンスの応答行程に確実に返信される。

・RM-DeliverResponse

受信した SOAP 応答電文を開始 RMP からその RM-Consumer に搬送する抽象的な動作。

・RM-Notify

前に送信された電文の失敗状態を RM-Producer または RM-Consumer に提供する抽象的な動作（例えば、信頼性電文が配信されなかったことを伝える通知)。

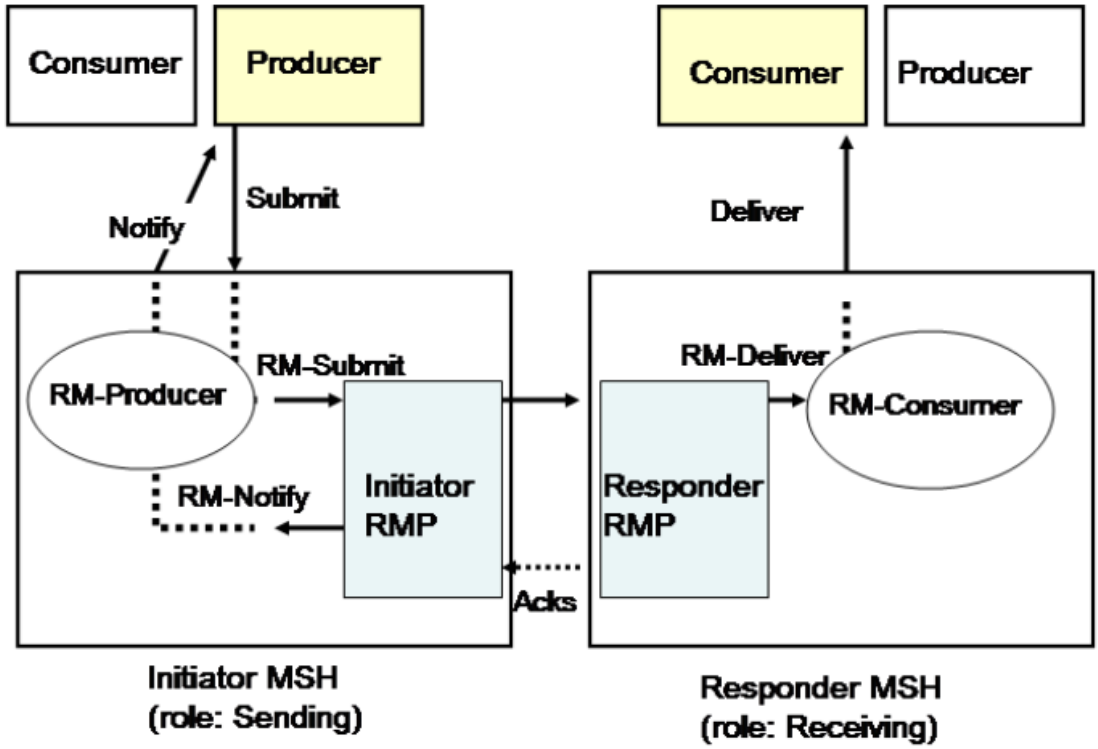

図 9：信頼性要求

図 9 は、信頼性要求（ 一方向の Push MEP、一方向の Pull MEP の最初の行程、または要求応答 MEP の最初の行程のいずれかにあるユーザー電文）を送信する際に関与する動作を表す。

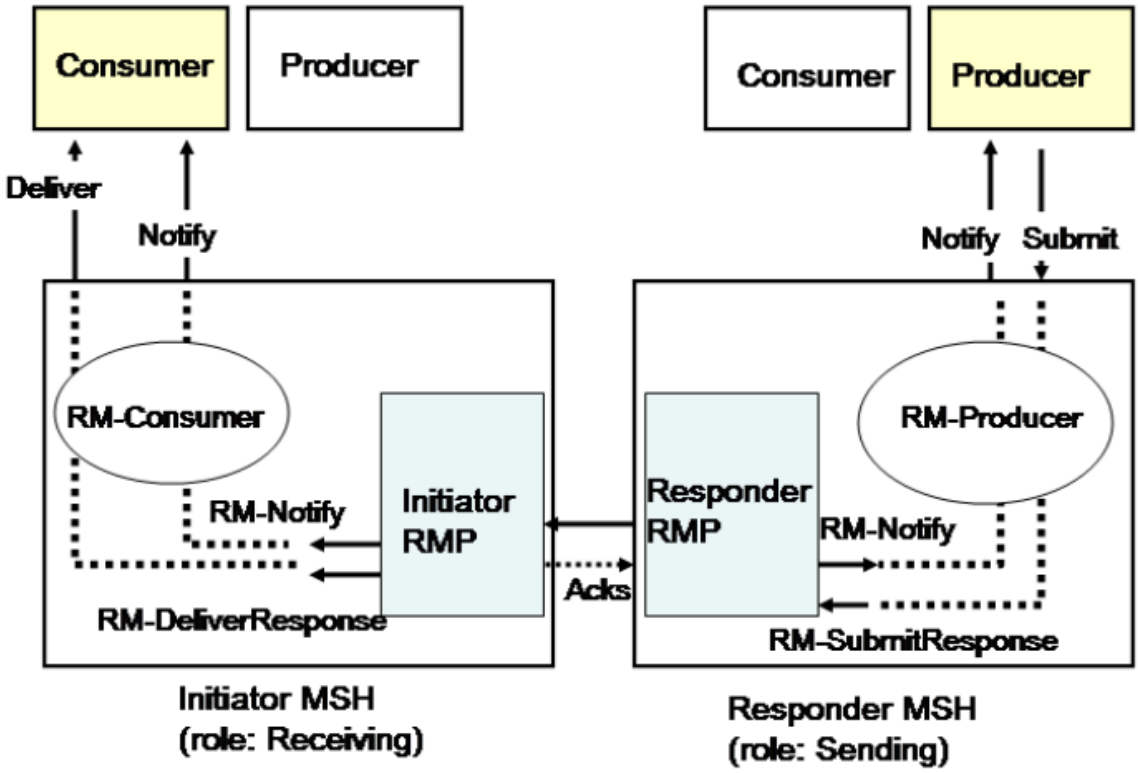

図 10：信頼性応答

図 10 は、一方向の Pull MEP にある pull されたユーザー電文または要求応答 MEP にある応答ユーザー電文（シンプルな ebXML 電文搬送サービス MEP の 2 番目の行程）のいずれかである応答を確実に送信する際に関与する抽象的な動作および構成要素を表す。信頼性の動作コンテキスト（OpCtx-Reliability）により、どちらにも配信失敗の通知が検出される。

8.2 ebXML 電文搬送サービス電文の信頼性

8.2.1 信頼性の合意

以下の信頼性の合意（またはサービスプロファイルの質）は、RMP によってサポートされなければならない。

・ 最小 1 回の配信

この信頼性の要件（RM-Submit の呼び出し）を伴う電文を送信する際、次の 2 つの結果のうちの 1 つが生じる。 (1) 応答 RMP は、RM-Consumer への電文配信（動作 RMDeliver）に成功する。(2) 開始 RMP または応答 RMP のいずれかが、RM 生成者と RM 利用者のそれぞれに配信失敗を通知する（動作 RM-Notify）。

・ 最大 1 回の配信

この信頼性の要件の下では、RM 生成者（動作 RM-Submit）によって開始 RMP に提出され

る電文は、応答 RMP からその RM-Consumer に 1 回以上は配信されない。電文複製の概念は、使用されている信頼性の仕様によってサポートされる電文 ID の概念に基づく。

・ 順番の配信

この信頼性の要件の下では、開始 RMP に提出される一連の電文（一連の RM-Submit の呼び出し）は、応答 RMP によって同じ順番でその RM-Consumer に配信される。

これらの合意は、ここで MEP 応答と呼ばれている応答電文にも適用できる。そのような場合、応答電文は、動作 RM-SubmitResponse と RM-DeliverResponse（それぞれ RM-Submit と RM-Deliver の代わりに）を伴う上記の合意に表現され、応答 RMP と開始 RMP の役割が入れ替わる。

これらの合意は組み合わせにすることもできる。特に、確実に 1 回の合意（最小 1 回と最大 1 回の合意の組み合わせから生じる）はサポートされなければならない。

これらの信頼性の合意をサポートするためには、開始 RMP と応答 RMP の両方が、トランスポートプロトコルから独立していて、エンドツーエンドの確認と電文再送能力を持つ信頼性のプロトコルを使用しなければならない。これらのプロトコル機能と関連のある詳細およびパラメータについては、信頼性モデルをインスタンス化する信頼性プロファイルに委ねられる。

8.2.2 ebXML 電文搬送サービス における信頼性の合意のサポート

信頼性のサービスの質（QoS）は、RMP の構成要素と相互に作用する MSH(電文処理機能)の内部の構成要素（RM-Producer および RM-Consumer と呼ばれる）に対してのみならず、MSH(電文処理機能）のユーザー層（生成者、利用者）にとっても意味のあるものでなければならないので、上記の合意を拡張し、それらを以下の抽象的な MSH(電文処理機能）の動作の観点から表現する必要がある。

・ 最小 1 回の ebXML 電文搬送サービス配信

この信頼性の要件（提出の呼び出し）を伴う電文を送信する場合、次の 2 つの結果のうちのいずれかが生じる。(1) 応答 MSH(電文処理機能）は、利用者への電文配信（配信動作）に成功する (2) 開始 MSH(電文処理機能）が生成者に配信失敗を通知（通知動作）するか、あるいは、応答 MSH(電文処理機能）が利用者に配信失敗を通知する。

・ 最大 1 回の ebXML 電文搬送サービス配信

この信頼性の要件の下では、開始 MSH(電文処理機能）に呼び出された提出の結果として送信された電文は、応答 MSH(電文処理機能）からその利用者に 1 回以上配信されることはない。

・ 順番の ebXML 電文搬送サービス配信

この信頼性の要件の下では、電文生成者によって開始 MSH(電文処理機能）に提出された一連の電文は、応答 MSH(電文処理機能）によって同じ順番で利用者に配信される。

上記のサービスの質の要件を満たすためには、MSH(電文処理機能）は、RMP によって提供される信頼性の機能と相互作用する以外に、以下のことを実行しなければならない。

・ MSH(電文処理機能）の抽象的な動作と RMP の抽象的な動作の間で適切なマッピングが行われるようにする。使用中の ebXML 電文搬送サービス MEP に依存するこのマッピングについては、次の項で説明している。

・ RMP 処理およびトランスポート層の外で生じる可能性がある追加の失敗も処理する。例えば

最小 1 回の配信において、提出された電文が RM-Submit が呼び出される前に失敗した場合、RM-Submit が呼び出された後に電文処理が失敗した時と同様に、MSH(電文処理機能）は、ユーザー層（生成者）が失敗の通知（通知の呼び出し）を受け取るようにしなければならない。同様に、RM-Deliver の呼び出しが成功したにもかかわらず受信側（配信）に電文が配信されなかった場合、受信者側（通知）に利用者への通知が発生しなければならない。

同様の合意は、上記の定義における開始 MSH（電文処理機能）と応答 MSH(電文処理機能）を入れ替えることによって、応答電文（例えば、ebXML 電文搬送サービスの Request-Reply MEP の 2 番目の行程）に適用される。

8.2.3 電文ヘッダーの処理ルール

推測の中には、ebXML 電文搬送サービス電文のヘッダーを処理する順番で行われるものもある。

MSH(電文処理機能）による ebXML 電文搬送サービス電文の処理は、Web サービスセキュリティヘッダーなどの ebms に適格なヘッダー以外のヘッダーの処理を含む場合がある。

RMP 実装を構成および再利用するためには、信頼性をサポートする SOAP ヘッダーの処理と ebms の処理が区別されていて、厳密に順番になっているのが望ましい。

信頼性のヘッダーと ebms に適格なヘッダーの間には、以下のシリアル化が必須である。

送信側：

1. ebXML 電文搬送サービス ヘッダーの処理（ebms に適格なヘッダーが電文に追加される）
2. 信頼性ヘッダーの処理（ヘッダーが電文に追加される）

受信側：

1. 信頼性ヘッダーの処理（電文からヘッダーが削除される）
2. ebXML 電文搬送サービス ヘッダーの処理 （ebms に適格なヘッダーが電文から削除される）

8.2.4 シグナル電文の信頼性

信頼性電文の中には、生成者が MSH（電文処理機能）にペイロードを提出することから生じないものもあり、代わりに MSH(電文処理機能）の機能（ebXML 電文搬送サービスシグナル電文として定義されている）によって開始される。これらも信頼性が確保される。このような電文に対しては、RMP の抽象的な動作の観点からのみ、信頼性の合意が表現される。送信側では、合意は RM-Submit の呼び出しから始まる（例えば、RM-Producer が RMP に電文データを提出する)。RM-Deliver の呼び出しが成功した場合、つまり、RM-Deliver が受信 MSH（電文処理機能）（RM-Consumer）の構成要素に配信された場合、電文は確実に送信されている。

8.3 ebXML 電文搬送サービス MEP の信頼性

8.3.1 一方向の Push MEP の信頼性

SOAP の一方向または SOAP 要求応答 MEP の最初の行程のいずれかとして送信されるプッシュされた電文は、動作“RM-Submit” を通して RMP モジュールに提出される。この MEP の信頼性インスタンスに対する一連の抽象的な動作の呼び出しは以下のとおりである。

開始 MSH(電文処理機能）側：

・ ステップ (1): Submit: 生成者当事者が MSH(電文処理機能）に電文データを提出する。
・ ステップ (2): RM-Submit: ebXML ヘッダーを処理した後、RMP に提出する
応答 MSH(電文処理機能）側：
・ ステップ (3): RM-Deliver: 信頼性ヘッダーを処理した後、MSH(電文処理機能）の他の機能に配信する。
・ ステップ (4): Deliver: ebXML ヘッダーを処理した後、MSH(電文処理機能）の利用者に電文データを配信する。
注：
・ 配信に失敗した場合、ステップ (4)（配信）に失敗して応答側に通知が呼び出されるか、あるいは、ステップ(3) と (4)の両方に失敗して、どちらかに RM-Notify が呼び出される。ステップが失敗するのは、それがワークフローの中で呼び出されない場合、あるいは、呼び出されたものの正しく完了しない場合のいずれかである。
・RM-Deliver は、MSH(電文処理機能）からその利用者への配信を含むものとして解釈される場合がある。言い換えると、受信した電文に対する配信の呼び出しが失敗した場合、応答 MSH(電文処理機能）あるいは開始 MSH(電文処理機能）のいずれかの側の失敗通知が誘因となって、同じ電文に対する RM-Deliver の呼び出しも失敗する（信頼性プロトコルに基づく)。

図 11 は、この信頼性 MEP の電文フローを表す。

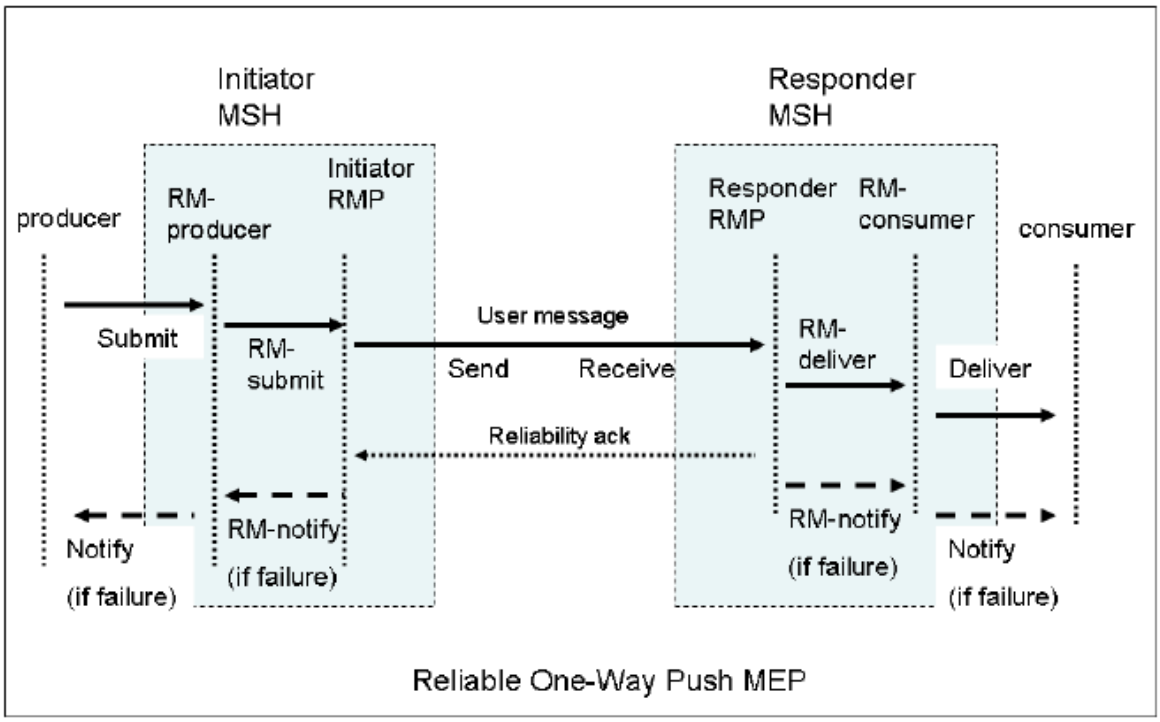

図 11：一方向の信頼性 Push MEP

8.3.2　一方向の Pull MEP の信頼性

この MEP の典型的かつ信頼性インスタンスに対する処理モデルは以下のとおりである。

応答 MSH(電文処理機能）側：

・ ステップ (1): Submit: 開始側の利用者向けに、生成者当事者が MSH(電文処理機能）に電文データを提出する。

開始 MSH(電文処理機能）側：

・ ステップ (2): 信頼性 PullRequest シグナルを MSH(電文処理機能）が作成する。このシグナルに対し、開始 RMP に RM-Submit が呼び出される。

応答 MSH(電文処理機能）側：

・ ステップ (3): MSH(電文処理機能）の機能が PullRequest シグナルを受信する。このシグナルに対し、応答 RMP に RM-Deliver が呼び出される。

・ ステップ (4): pull された電文が RMP に提出される。これにより RMSubmitResponse が呼び出される。

開始 MSH(電文処理機能）側：

・ ステップ (5): RM-DeliverResponse: プルされた電文の信頼性ヘッダーを処理した後、RM-Consumer に配信する。

・ ステップ (6): Deliver: ebXML 電文搬送サービスヘッダーを処理した後、プルされた電文データを MSH(電文処理機能）の利用者に配信する。

図 12 は、この信頼性 MEP の電文フローを表す。

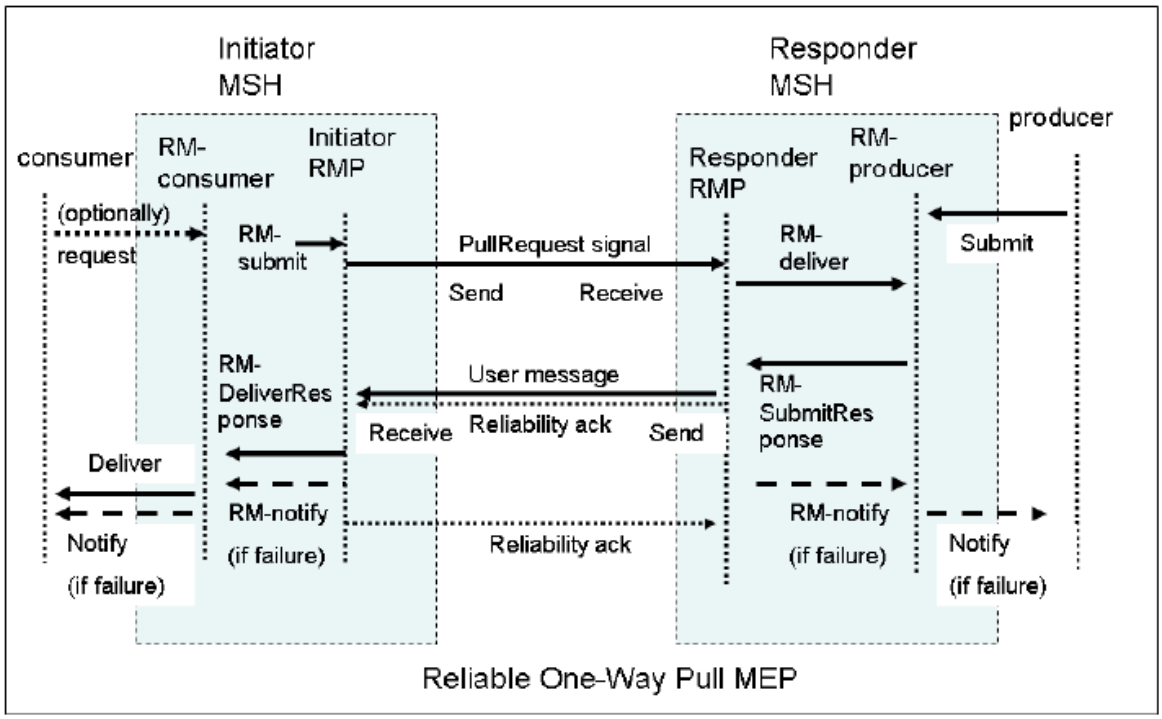

図 12：一方向の信頼性 Pull MEP

以下のシンプルな要求応答 MEP だけではなく、この MEP でも、MEP 要求（ここでは

PullRequest シグナル）に適用する同じ信頼性の合意は、動作 RMSubmitResponse および RM-DeliverResponse によって処理される MEP 応答にも適用できる。

このような場合、つまり、MEP 応答が信頼性の合意を結んでいる場合、以下の要件が適用される。

・MEP 応答が最小 1 回の信頼性合意を結んでいる場合、MEP 要求も最小 1 回の信頼性合意を結んでいなければならない。さらに、MEP 要求が最大 1 回の信頼性合意も結んでいて、応答 RMP によって配信および応答されている場合、 MEP 応答の複製が後に受信されるなら、最初の要求に返された同じ応答の複製は、複製の要求に返されなければならない。

・MEP 応答が最大 1 回の配信と合意している場合、MEP 要求答も最大 1 回の配信と合意していなければならない。

8.3.3 要求応答 MEP の信頼性

要求応答 MEP の信頼性は、RMP が関与する限り、一方向の Pull MEP の信頼性と同様に処理される。この MEP の典型的かつ適格なインスタンスに対する処理モデルは以下のとおりである。

開始 MSH(電文処理機能）側：

・ ステップ (1): Submit: 生成者当事者が MSH(電文処理機能）に要求電文データを提出する。

・ ステップ (2): RM-Submit: 開始 RMP に要求電文を提出する。

応答 MSH(電文処理機能）側：

・ ステップ (3): RM-Deliver: 信頼性ヘッダーを処理した後、要求電文を RM-Consumer に配信する。

・ ステップ (4): Deliver: MSH(電文処理機能）の利用者に要求電文データを配信する。

・ ステップ (5): Submit: 開始側の生成者向けに、要求電文の利用者が応答電文データを提出する。

・ ステップ (6): RM-SubmitResponse: 応答電文の RM-Producer が応答 RMP に提出する。

開始 MSH(電文処理機能）側：

・ ステップ (7): RM- DeliverResponse: 応答電文を RM-Consumer に配信する。

・ ステップ (8): Deliver: 開始 MSH(電文処理機能）の利用者に応答電文データを配信する。

図 13 は、この信頼性 MEP の電文フローを表す。

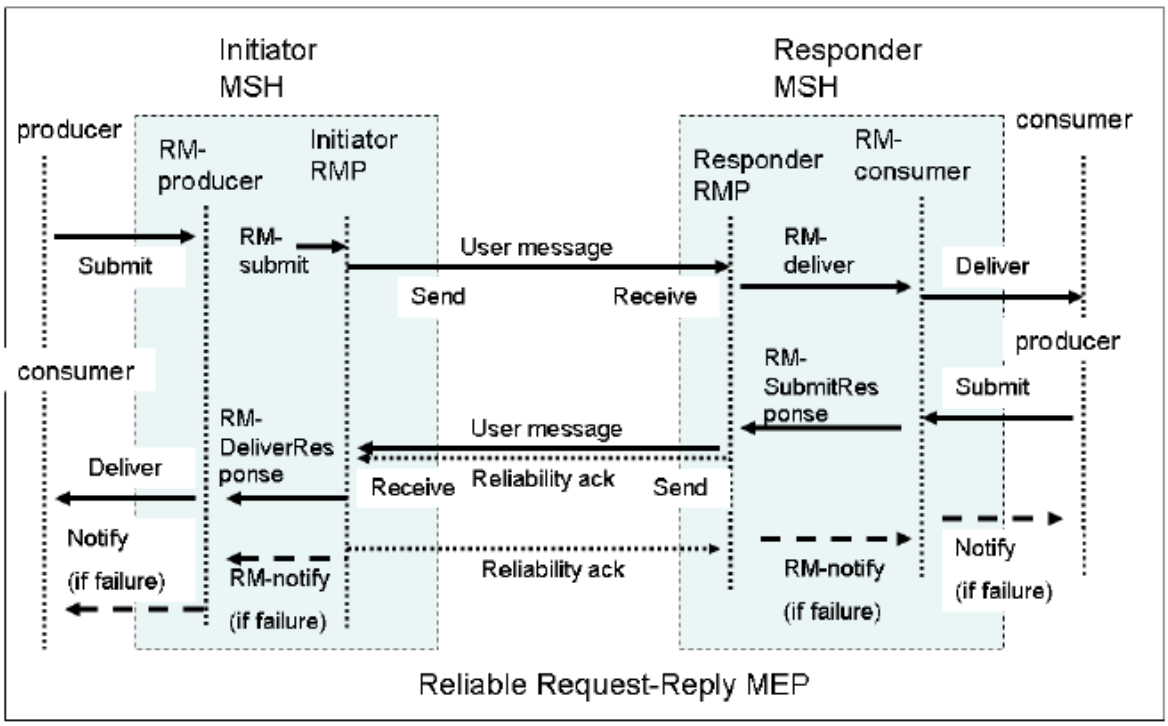

図 13：信頼性要求応答 MEP

MEP 応答が信頼性合意を結んでいる場合、一方向の Pull MEP で説明した MEP 要求の信頼性との依存性と同じ依存性がここでも適用される。

ebXML による次世代 EDI 促進報告書

平成１８年　３月　発行

発　行　　次世代電子商取引推進協議会

販　売　　財団法人　日本情報処理開発協会
電子商取引推進センター
東京都港区芝公園三丁目５番８号
機械振興会館３階
TEL：０３（３４３６）７５００

この資料は再生紙を使用しています。