

# cXML ユーザー・ガイド

バージョン 1.2.014

2005 年 10 月

岸 和孝 抄訳

## 訳注

… [～] は, ～に対する訳が不明なまたは未確定な箇所です。

便宜上, 見出しに章節番号を付け, 図に見出しを付けました。

原書の「第 1 章 cXML とは」と「第 2 章 cXML の基礎」の抄訳です。

## 序

この文書は、電子商取引と関連するデータの通信のための cXML (commerce eXtensible Markup Language) をどのように使うかについて記述します。

### 読者と必要条件

この文書の読者は、cXML 対応アプリケーションを設計するアプリケーション開発者を意図しています。cXML は、次のような処理条件のためのオープンな多目的の言語です。

- ・ ネットワーク電子商取引ハブ [Network e-commerce hubs]
- ・ 電子的製品カタログ [Electronic product catalogs]
- ・ パンチアウト・カタログ [PunchOut catalogs]
- ・ 購買アプリケーション [Procurement applications]
- ・ 購買者 [Buyers]
- ・ 供給業者 [Suppliers]
- ・ 電子商取引サービス提供者 [E-commerce service providers]

読者は、電子商取引の概念、HTTP インターネット通信の標準、XML 形式についての実用的知識を持っているとします。

この文書は、特定の調達アプリケーションや商取引ネットワーク・ハブをどのように用いるかについて記述しません。

## どの章を読むべきか

- ・ 電子商取引ビジネス管理者 [E-commerce Business Managers]

cXML の能力の概要については、第 1 章の「cXML とは」を読んでください。

- ・ ウェブ・プログラマー [Web Programmers]

電子商取引サイトを実現するウェブ・プログラマーは、すべての章を読むべきです。

- ・ カタログ作成者 [Catalog Creators]

cXML カタログを作る供給業者は第 12 章「カタログ」を読むべきです。

- ・ パンチアウト・サイト開発者 [PunchOut Site Implementors]

パンチアウト・ウェブサイトを作るウェブ・プログラマーは、第 4 章の「パンチアウトの処理」を読むべきです。

## タイポグラフィー

cXML の要素と属性は、等幅フォントで示します。cXML の要素と属性の名前は、大文字・小文字が区別されます。両方共に大文字と小文字の組合せで、要素は大文字で始まり、属性は小文字で始まります。例えば、MyElement は cXML の要素で、myAttribute は cXML の属性です。次の表は、本書で使ったタイポグラフィーの取り決めに記述しています。

書体または記号	意味	例
<AaBbCc123>	変更の必要があるテキストは“<”と“>”の間に現れるイタリック体です。	http://<server>:<port>/inspector
AaBbCc123	ユーザー・インターフェース制御、メニュー、メニュー項目の名前。	File メニューから Edit を選ぶ。
AaBbCc123	ファイル名とディレクトリ名。パラメーター。CSV ファイルのフィールド。コマンド・ライン。コード例。	システムの各レポートに関する ReportMeta.csv の 1 行がある。
<i>AaBbCc123</i>	本の名前。	より詳細な情報については、「 <i>Acme Configuration Overview</i> 」を参照してください。

## 第1章 cXML とは

この章では、電子商取引 [electronic-commerce transactions] のための cXML (commerce eXtensible Markup Language) を紹介します。

この章では、次のことについて記述します。

- ・ cXML, XML の実現
- ・ cXML の能力
- ・ cXML を用いるアプリケーションの種類
- ・ 内容配送の戦略
- ・ cXML の文書型定義
- ・ プロファイル・トランザクション
- ・ XML のユーティリティ

## 1.1 cXML, XML の実現

XML (eXtensible Markup Language) は、言語の構文を作るために使われるメタ・マークアップ言語です。また、それはアプリケーション、とりわけインターネットにおける通信でデータをやりとりするための基準です。

XML 文書はタグと値が対になった形の、例えば、次のようなデータを含んでいます。

```
<DeliverTo>金太郎</DeliverTo>
```

XML は、XML の親にあたるメタ言語である、SGML の実現である HTML (HyperText Markup Language) に似た構造を持っています。しかし、XML ではすべてのデータにその目的に応じてマークを付けられるので、アプリケーションは、HTML より容易に XML 文書からのデータを引き出し利用できます。HTML がデータと表示情報を含むのに対して、XML はデータだけを含んでいます。

各々の cXML 文書は、XML 文書型定義 (DTD) に基づいて構成されます。テンプレートとして働く DTD は、cXML 文書の内容モデル、例えば、要素の妥当な順序と入れ子、属性のデータ型、を定義します。

cXML の DTD は、ウェブ・サイト「[www.cXML.org](http://www.cXML.org)」で利用可能なファイルです。さらに詳細な情報については、第 1 章の「cXML の DTD の入手」を参照してください。

## 1.2 cXML の能力

cXML は、購買者 [buying organizations] , 供給業者, サービス提供者, 仲介業者がオープンな一つの標準の言語を使って伝達することを許します。

成功している企業間電子商取引 (B2B e-commerce) ポータルは、柔軟で広く採用されたプロトコルに依存します。cXML は、企業間電子商取引のために特に設計された、明確で堅牢な言語です。そしてそれは多数の購買者, 供給業者の選択です。

cXML 取引は、あらかじめ定義されたタグによって囲まれた値を含む単純なテキストファイルである文書から成り立ちます。

cXML 文書のほとんどの種類は、伝統的にビジネスで利用されるハードコピー文書に類似しています。

cXML 文書の一般に利用される種類は、次のとおりです。

- ・ カタログ [Catalogs]
- ・ パンチアウト [PunchOut]
- ・ 購買注文 [Purchase Orders]

次節でそれらの cXML 文書について記述します。

## 1.2.1 カタログ

カタログは、製品とサービス内容を購買者へ伝えるファイルです。それらは、供給業者によって提供された製品とサービス、それらの価格について記述します。さらに、それらは、供給業者から彼らの顧客への主要な通信チャンネルです。

調達アプリケーションを使う購買者がそれらの製品とサービス提供を確認し購買できるように、供給業者はカタログを作ります。調達アプリケーションは、カタログを読み、データベースへ内部的にそれらを蓄えます。

購買者がカタログを承認した後、その内容は利用者に目に見えるようになり、品目を選択でき、購買要求にそれらを追加できるようになります。

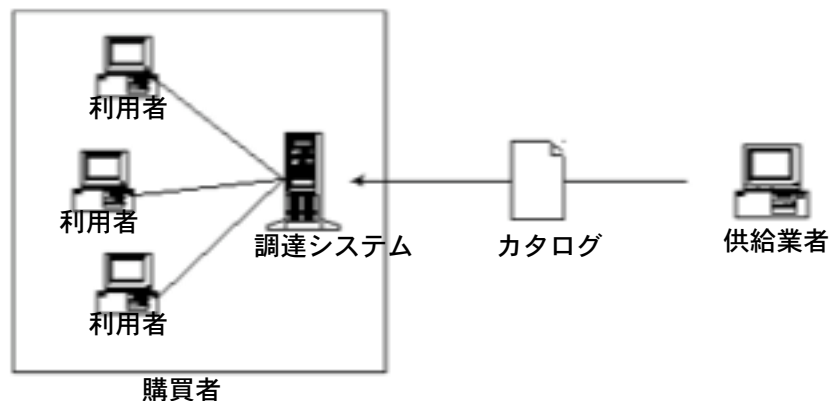


図 1.2.1 製品とサービスの内容を供給業者から購買者へ送信する

供給業者は、どのような製品やサービスのカタログも、それがどのように評価され、値踏みされ、引き渡されるかにかかわらず、作ることができます。

カタログにおける各々の品目において、基本的な情報は必須で、選択的な情報は、多言語記述のような、先進的なカタログ機能を可能にします。

## 1.2.2 パンチアウト

パンチアウトは、インターネットで管理される対話型セッションのための実現が容易なプロトコルです。パンチアウトは、リアルタイムの同期式の cXML メッセージを使って、遠隔地における利用者の相互作用を継ぎ目無く与えることで、アプリケーション間の通信を可能にします。

パンチアウトには、次の 3 種類があります。

- ・ 調達パンチアウト [Procurement PunchOut]
- ・ パンチアウト連鎖 [PunchOut chaining]
- ・ 提供者パンチアウト [Provider PunchOut]

### 1.2.2.1 調達パンチアウト

調達パンチアウトは、静的なカタログ・ファイルに代わるものを供給業者に与えます。パンチアウト・サイトは、ウェブサイトで稼働しているライブの対話型カタログです。電子商取引のウェブサイトを持つ供給業者は、パンチアウトをサポートするためにそれらを変更できます。パンチアウト・サイトは、cXML を使うことによってインターネット上で調達システムと通信します。

パンチアウト・サイトにおいて、調達アプリケーションは、製品や価格付けの詳細の代わりにボタンを表示します。利用者がそのボタンをクリックする時、ウェブ・ブラウザは供給業者のローカルなウェブ・サイトからページを表示します。供給業者がどのようにこれらのページを実現するかによって、利用者は、製品の選択を閲覧し、形状を指定し、そして配送方法を選択できます。利用者が選択している品目を選んでボタンをクリックすると、調達アプリケーションに注文情報が向けられます。完全に構成された製品とその価格は利用者の購買要求の範囲内に現れます。

さらに詳細な情報については第 4 章の「パンチアウトの処理」を参照してください。

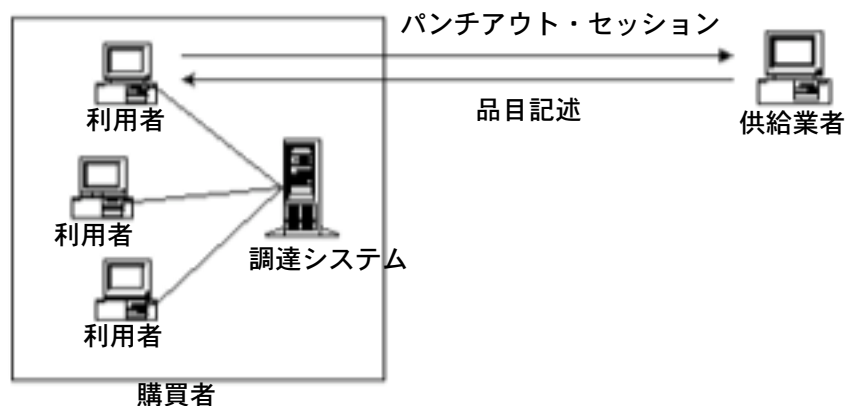


図 1.2.2.1 利用者と供給業者のウェブ・サイト間の対話型パンチアウト・セッション

供給業者のウェブ・サイトは、合意の上の契約製品と価格を前もって提示できます。

### 1.2.2.2 パンチアウト連鎖

パンチアウト連鎖は、一つ以上のパンチアウトを含む調達パンチアウトです。cXML パス・ルーティング [cXML Path Routing] は、この相関関係を可能にします。



図 1.2.2.2 消費者，市場，供給業者

cXML パス・ルーティングは、市場へ向けられる注文とその他の結果として起こるメッセージ、見積りの提出に従事する供給業者、を許します。パス・ルーティングは、最後の注文についてすべての当事者に通知します。そして、結果として起こるどのようなパンチアウトも、市場に代わってどのように注文を分割するかを調達アプリケーションに対して指定します。

### 1.2.2.3 提供業者パンチアウト

提供業者パンチアウトは、クレジット・カードの認証、利用者証明、自己登録のような、開始アプリケーションにサービスを提供する遠隔アプリケーションに対して状況を記録する [to punch out] アプリケーションを可能にします。

### 1.2.3 購買注文

購買者は、契約遂行を依頼するために供給業者に対して購買注文を送ります。

さらに詳細な情報については第 5 章の「購買注文」を参照してください。

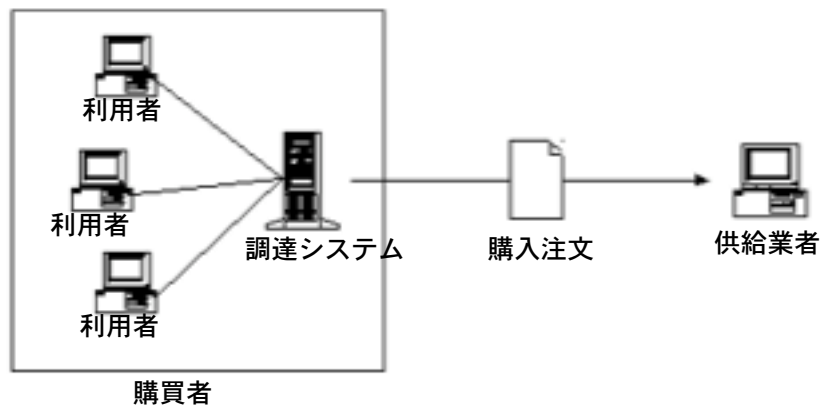


図 1.2.3 供給業者に対して送られる購買注文

cXML は、柔軟で、実現が安価で、しかもデータの広い配列と添付をサポートするので、(ANSI X12 EDI 850 のような) 他の形式よりも購買注文の通信に適しています。

## 1.3 cXML を用いるアプリケーションの種類

cXML は、どのような電子商取引アプリケーションでも利用できます。現在、cXML は、購買機関 [buying organizations]，垂直方向かつ水平方向の購買共同体 [buying communities]，供給業者 [suppliers]，アプリケーション販売業者 [application vendors] によって利用されています。次の節では、現時点における cXML を使うアプリケーションの主要な種類について述べます。

### 1.3.1 調達アプリケーション

Ariba Buyer と Ariba Marketplace (ネットワーク版) のような調達アプリケーションは、外部の処理のために cXML を使います。

Ariba Buyer は、イントラネット上で従業員によって使用されるための、大組織によって主催された企業アプリケーションです。

Ariba Marketplace (ネットワーク版) は、小規模から中規模の多くのビジネスから成り立つ購買共同体の創設を許すインターネット・ベースのサービスです。

それらのアプリケーションは、購買管理者 [purchasing managers] によって承認されている販売業者から契約製品とサービスを購入する利用者の共同体を許します。最初に、共同体の管理者は、依頼された購入を承認します。次に、承認されている購入注文は、幾つかの可能なチャンネルを通して、インターネット上で cXML を含んで、供給業者に伝えられます。

### 1.3.2 商業ネットワーク・ハブ

Ariba Supplier Network のような商業ネットワーク・ハブは、購買者と供給業者をつなぐためのウェブ・ベースのサービスです。これらのウェブ・サービスは、カタログの検証と版改訂 [catalog validation and versioning] , カタログの出版と署名 [catalog publishing and subscription] , 自動化した購買注文のルーティング [automated purchase order routing] , 購買注文の履歴 [purchase order history] といった機能を提供します。

商業ネットワーク・ハブは、多様な組織から、あるいは組織への依頼 [requests] と回答 [responses] を証明し、ルートを定める仲介者として実行できます。それらの組織の間の通信は、インターネット上で cXML を介して完全に行えます。

### 1.3.3 パンチアウト・カタログ

前節で記述したように、パンチアウト・カタログは、供給業者のウェブ・サイトにおいて利用できる対話型のカタログです。パンチアウト・カタログは、購買者のパンチアウト・セッションを管理する ASP (Active Server Pages) , JavaScript, CGI (Common Gateway Interface) のようなプログラミング言語で書かれたウェブ・サーバー・アプリケーションによって可能にされます。

パンチアウト・カタログは、調達アプリケーションからパンチアウト依頼 [PunchOut requests] を受け取り、購買者を確認し、適切な製品と価格を HTML 形式で表示します。それから、適切であれば、利用者は品目を選んで、それらを構成し、オプションを選択します。

パンチアウト・セッションの終わりにおいて、パンチアウト・サイトは、利用者の選択の記述を cXML 形式で調達アプリケーションに送ります。

さらに詳細な情報については第 4 章の「パンチアウトの処理」を参照してください。

### 1.3.4 受注システム

受注システム [Order-receiving systems] は購買者から送られた購買注文を受付けて処理する、供給業者サイトにおけるアプリケーションです。

受注システムは、在庫管理システム、注文即遂行システム [order-fulfillment systems] , 注文処理システム [order-processing systems] のような、どのような自動システムにもできます。

cXML 購買注文から情報を引き出すことは単純ですので、既存の受注システムがそれらを受け付けることを可能にする、アダプターを作ることは比較的容易です。

さらに詳細な情報については第 5 章の「購買注文」を参照してください。

## 1.4 内容配達戦略

調達アプリケーションは、製品とサービス内容を利用者に対して示します。供給業者は、顧客が製品やサービスを見る方法を制御することを望みます。なぜならばプレゼンテーションが彼らの営業プロセスにとって重要だからです。購買者は、高い契約承諾を保証するために、内容を容易に届きやすく検索可能にすることを望みます。

購買者と供給業者は、製品とサービス内容を配達するための複合的な方法を選択できません。使う特有の方法は、購買者と供給業者の間の契約と、製品の性質が取り引きされたサービスによって決定されます。

次の表では、一般に入手した製品とサービスの事例のカテゴリーと、それらの望まれた内容配達の方法を列挙します。

必需品	性質	内容配達の方法
オフィス必需品, 内部的な必需品	静的な内容, 安定した価格付け	静的なカタログ
製造工場必需品, MRO (保守, 修繕, 運転), 電子部品	正常化に有用なことを要求する。	垂直方法の必需品ポータルへのパンチアウト
書籍, 化学薬品	非常に多い品目	供給業者の主催するサイトへのパンチアウト
コンピューター, ネットワーク設備, 周辺装置	多くの可能な構成	供給業者の主催する構成ツールへのパンチアウト
サービス, 印刷物	内容には極めて変わりやすい属性がある。	供給業者サイトにおける電子的フォームへのパンチアウト

購買者は、パンチアウトを介して、購買機関内で局所的に内容を貯蔵でき、あるいは、インターネット上で遠隔的に内容を呼び出せます。cXML カタログは、両方の記憶戦略をサポートします。

この表が示すように、パンチアウトは、どの供給業者に対しても、彼らの必需品や顧客に依存して、柔軟な枠組みを提供し、カスタマイズされた内容を提供できます。この内容戦略の目的は、道理にかなう方法でカタログデータを交換することを購買者と供給業者に対して許すことです。

## 1.5 cXML の文書型定義

cXML は XML 言語ですので、ひとそろいの文書型定義 (DTD) はそれを完全に定義します。それらの DTD は、cXML 要素の正確な構文と順序を記述するテキストファイルです。DTD は、cXML を読んだり書いたりするアプリケーションにそれらの検証を可能にします。

各々の cXML 文書のヘッダーは、文書を定義する DTD の URL を含みます。cXML アプリケーションは、DTD を検索でき、そして文書を検証するためにそれを使用できます。

最も堅牢な取引処理のために、受入れたすべての cXML 文書を検証してください。エラーが検出される場合、適切なエラーコードが生じますので、送り手は再送できます。cXML アプリケーションは、検証は勧められていますが、受入れた cXML 文書を検証することを要求されません。しかし、すべての cXML 文書は妥当でなければならず、さらに次の節で記述される cXML の DTD を参照しなければなりません。

### 1.5.1 cXML の DTD の入手

cXML のすべてのバージョンのための DTD は、cXML.org で利用できます。cXML 文書のさまざまな種類は、DTD の大きさを減少させるために複合的な DTD で定義されます。それは幾つかのパージャーにおいてより速い検証を可能にします。

文書	DTD
基本の cXML 文書	<a href="http://xml.cXML.org/schemas/cXML/&lt;version&gt;/cXML.dtd">http://xml.cXML.org/schemas/cXML/&lt;version&gt;/cXML.dtd</a>
確認書 [Confirmation] と 出荷通知書 [Ship Notice]	<a href="http://xml.cXML.org/schemas/cXML/&lt;version&gt;/Fulfill.dtd">http://xml.cXML.org/schemas/cXML/&lt;version&gt;/Fulfill.dtd</a>
送り状 [Invoice]	<a href="http://xml.cXML.org/schemas/cXML/&lt;version&gt;/InvoiceDetail.dtd">http://xml.cXML.org/schemas/cXML/&lt;version&gt;/InvoiceDetail.dtd</a>
型定義 [Type Definition]	<a href="http://xml.cXML.org/schemas/cXML/&lt;version&gt;/Catalog.dtd">http://xml.cXML.org/schemas/cXML/&lt;version&gt;/Catalog.dtd</a>
支払い送金 [Payment Remittance]	<a href="http://xml.cXML.org/schemas/cXML/&lt;version&gt;/PaymentRemittance.dtd">http://xml.cXML.org/schemas/cXML/&lt;version&gt;/PaymentRemittance.dtd</a>

ここで、<version>は、1.2.014 のように、完全な cXML バージョンです。

cXML アプリケーションは、すべての入出力する文書 [all incoming and outgoing documents] を検証するためにそれらの DTD を用います。

## 1.5.2 DTD の貯蔵

最も良い性能のために、cXML アプリケーションは、局所的に DTD を貯えるべきです。cXML の DTD ファイルが発表された後は、それらは決して変更されませんので、あなたは無期限にそれらを貯蔵できます (DTD の各々の新しいバージョンには新しい URL があります)。cXML アプリケーションが cXML 文書を解析する時、それらは文書ヘッダーで SYSTEM 識別子を検査すべきで、それがすでに局所的に貯蔵されていない場合、その DTD を検索すべきです。

DTD の局所的貯蔵は、より速い文書検証の利点と cXML.org サイトへのより少ない依存を提供します。

幾つかの環境では、cXML アプリケーションは、それらが新しい文書を受け取る時に、DTD を自動的に検索することを許さないかもしれません。これらの環境では、DTD を手で検索し、それらを局所的に貯蔵し、アプリケーションに cXML.org でそれらを局所的に検索する指示をしなければなりません。しかし、生成された cXML 文書は、局所的な DTD でなく、cXML.org で DTD を指さなければなりません。

## 1.6 プロファイル・トランザクション

プロファイル・トランザクション [Profile transaction] は、特有の cXML サーバーがどのようなトランザクションを受け付けできるかについての基本的な情報を伝達します。すべての cXML サーバーは、このトランザクションをサポートしなければなりません。それは、クライアント・システムで利用できる cXML サーバーの能力を作ることで、アプリケーション間のバックエンド統合 [back-end integrations] を意図しています。

このトランザクションは、二つの文書、すなわち ProfileRequest と ProfileResponse から成り立ちます。それらは一緒に、サポートされた cXML バージョン、サポートされたトランザクション、それらのトランザクションに対する選択を含むサーバー能力を検索します。

注：すべての cXML 1.1 以上のサーバーは、プロファイル・トランザクションを受け付けできなければなりません。

### 1.6.1 ProfileRequest

ProfileRequest 文書は、内容を持ちません。それは、指定された cXML サーバーへのルートを単純に定めます。

### 1.6.2 ProfileResponse

サーバーは、それがサポートする cXML トランザクション、それらの場所、文字列値を持つ名前の付いた選択を列挙する、ProfileResponse 文書に応答します。

## 1.7 サービス状態応答

POST された cXML を受け付けた URL からの 200 という状態コードを持つ応答は、上りの処理中です。HTTP GET がサービス場所へ送られる時、そのサービスは、妥当な動的に生成された cXML Response 書類を応答します。サービスは、cXML Request 文書を受け取られる場所においてどのような HTTP URL であってもかまいません。

## 1.8 XML のユーティリティ

XML ファイルを編集したり検証したりするためのユーティリティはウェブにおいて無償あるいは有償で利用可能です。

それらのユーティリティの幾つかを次に述べます。

ユーティリティ	開発元	機能	サイト
Internet Explorer	Microsoft	DTD に対して XML ファイルを検証できる XML 対応ウェブブラウザ	<a href="http://www.microsoft.com/windows/ie/default.htm">www.microsoft.com/windows/ie/default.htm</a>
Turbo XML	TIBCO Software	XML アセットの生成, 検証, 変換, 管理のための統合開発環境 (IDE)	<a href="http://www.tibco.com/software/business_integration/turboxml.jsp">www.tibco.com/software/business_integration/turboxml.jsp</a>
XML Spy	Altova	グリッド, ソース, ビューを持つ, DTD と XML ファイルを保守するためのツール	<a href="http://www.altova.com">www.altova.com</a>
XMLwriter	Wattle	SoftwareXML プロジェクトを管理するために設計された分かりやすい XML オーサリング・ツール	<a href="http://www.xmlwriter.net">www.xmlwriter.net</a>

さらに詳細な XML ツールについては、次のウェブ・サイトを参照してください。

[www.xmlsoftware.com](http://www.xmlsoftware.com)

[www.xml.com](http://www.xml.com)

## 第 2 章 cXML の基礎

この章では、cXML の基本プロトコルとデータ形式について記述します。これは、すべてのトランザクションを実現するために必要とされる情報を含んでいます。

この章では、次のことについて記述します。

- ・ プロトコルの仕様
- ・ 基本要素

### 2.1 プロトコルの仕様

cXML トランザクションに関する二つの通信モデル、すなわち Request-Response（要求-応答）モデルと片方向モデルがあります。これらの二つのモデルが操作を厳密に指定するので、それらは単純な実現を可能にします。一つのモデルが適切でないような状況がありますので、両方のモデルは必要とされます。

### 2.1.1 Request-Response (要求-応答) モデル

Request-Response トランザクションは、HTTP か HTTPS 接続だけで実行されます。次の図は、当事者 A と B の間の Request-Response の相互作用におけるステップを説明します。



図 2.1.1 Request-Response の相互作用

このトランザクションは、次のようなステップを含んでいます。

1. サイト A は、サイト B のアドレスを表わす、前もって決定された URL 上のサイト B と HTTP/1.x 接続を開始する。
2. サイト A は、HTTP 接続を介して cXML 文書を送るために POST 操作を使う。次にサイト A は回答を待つ。
3. サイト B は、ステップ 1 で用いた URL によって指定された資源へ HTTP Request を発送する HTTP/1.x-compliant サーバーを持つ。この資源は、サイト B の HTTP サーバーを知っているどのような妥当な場所でもありえる。例えば、CGI プログラムや ASP ページ。
4. ステップ 3 で識別されたサイト B の資源は、cXML 文書内容を読み、その要求に関して適切なハンドラーを Request に対応づける。
5. cXML Request のためのサイト B のハンドラーは、Request が指定し cXML Response 文書を生成する仕事を行なう。
6. サイト B は、ステップ 1 で確立した HTTP 接続を介してサイト A へ cXML Response を送る。
7. サイト A は、cXML Response を読み、それを Request を開始した処理に返す。
8. サイト A は、ステップ 1 で確立した HTTP 接続を閉じる。

さらに、この処理は、Request-Response サイクルを反復します。上記の過程における仕事を簡単にするために、cXML 文書は二つの部分に分けられます。

- ・ ヘッダー

認証情報とアドレッシングを含みます。

- ・ 要求または応答データ

具体的な要求か回答，渡されるべき情報を含みます。

これらの要素の両方が親のエンベロープ要素で運ばれます。次の例は、cXML Request 文書の構造を示します。

```
<cXML>
  <Header>
    ヘッダー情報
  </Header>
  <Request>
    要求情報
  </Request>
</cXML>
```

次の例は、cXML Response 文書の構造を示します。

```
<cXML>
  <Response>
    応答情報
  </Response>
</cXML>
```

Response 構造は、Header 要素を用いません。Response が Request と同じ HTTP 接続でいつも伝わりますので、それは必要ありません。

## 2.1.2 cXML の規則

cXML は、伝統的なビジネス文書の性質である個別の項目を記述するために要素を用います。さらに、要素は、明白な下位区分、それらの下位区分間の関係を持つ情報を記述します。例えば、国、市、番地から構成される住所。

また、cXML は、要素を修飾したり文脈を与えたりする属性を用います。

要素と属性の名前は、大文字と小文字を区別し、単語の接続にはハイフンではなく頭文字を用います。要素名は大文字で始め、属性名は小文字で始めます。例えば、次のようです。

### 要素

Sender, Credential, Payment, ItemDetail

### 属性

payloadID, lineNumber, domain

任意の要素がどのような内容も持たない（つまり、空）場合、それらを完全に取り去ってください。欠けている値が幾つかのパージャーに影響を与えることがありますので、空か空白の要素を避けてください。

DTD ファイルとその文書において、記号はトランザクションで要素が何回出現できるか示すために用いられます。「+」は、要素が 1 個以上出現できるということを意味し、「?」は、要素が 0 個または 1 個出現できるということを意味し、「\*」は、0 個または 1 個以上出現できるということを意味します。

### 2.1.3 cXML 文書

cXML 要素は、cXML 文書の本体です。一つの文書は、次のように始まるでしょう。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML xml:lang="en-US" payloadID="1234567.4567.5678@buyer.com"
      timestamp="2002-01-09T01:36:05-08:00">
```

cXML 文書の最初の文字は、「<?」または「<！」でなければなりません。文書は、空白文字またはタブから始まってはなりません。例えば、PunchOutOrderMessage 文書を含む HTML フォームは、開始の引用符と「<」の間にいかなる文字も挿入してはなりません。

cXML 文書の 2 行目は、DOCTYPE 文書型宣言を含まなければなりません。これは cXML 文書に現われることができる唯一の外部実体です。この行は、cXML DTD を参照しています。cXML DTD に関するより詳細な情報については第 1 章の「cXML の文書型定義」を参照してください。

cXML 文書は、cXML, Supplier, Contract, Index という最上位要素のいずれか一つを持ちます。cXML 要素は、「商取引」のデータに関するものです。その他の要素は、静的な内容を記述します。

## 2.1.4 ラッピング層

通常、cXML 文書は、text/xml の MIME (Multipurpose Internet Mail Extensions) メディア型を指定する HTTP ヘッダーと、cXML 文書における符号化と一致する文字集合パラメーターを持つ、HTTP を介して伝送されます。

HTTP は完全な 8 ビットですので、受け付けるパーサーによってサポートされるどのような文字符号化も base64 か quotedprintable のような内容転送の符号化 [content-transfer encoding] なしに利用されます。

すべての XML パーサーは、すべての US-ASCII を含む、すべての Unicode 文字を含む UTF-8 (Universal Transformation Format) 符号化をサポートします。したがって cXML 文書を伝送している時、アプリケーションは UTF-8 を使うべきです。

注：

IETF RFC 2376 「XML のメディア型」によると、MIME 文字集合パラメーターは XML 宣言で指定されたどのような符号化も無効にします。さらに、text/xml メディア型のための省略時符号化は、XML 仕様の 4.3.3 節で述べられている UTF-8 でなく、us-ascii です。明らかにするために、cXML 文書は、XML 宣言に明白な符号化を含めるべきです。MIME エンベロープは、text/xml に関して一致する文字集合パラメーターを使うべきです。また、application/xml メディア型も利用できます。そのメディア型は、XML 宣言を無効にしません。あるいは受け手の解読している記録に影響を及ぼしません。さらに文字集合パラメーターを必要としません。

cXML 文書の HTTP 伝送は、次のような MIME と HTTP ヘッダーを含むかもしれません。

```
POST /cXML HTTP/1.0
Content-type: text/xml; charset="UTF-8"
Content-length: 1862
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
User-Agent: Java1.1
Host: localhost:8080

Connection: Keep-Alive
<?xml version="1.0" encoding="UTF-8"?>
...
```

## 2.1.5 添付

cXML プロトコルは、cXML 文書に対してどのような種類の外部ファイルの添付もサポートします。例えば、購買者は購買注文を説明するためにメモ、図面、ファックスを伴う必要がときどきあります。別の例は、添付としてのカタログ・ファイルを含む CatalogUploadRequest 文書です。

cXML 文書によって参照されたファイルは、受け手によってアクセス可能なサーバー上か、cXML 文書自身も含むエンベロープ内に存在できます。一つのエンベロープの中の cXML 文書に外部ファイルを添付するために、多目的インターネット・メール拡張 (MIME—Multipurpose Internet Mail Extensions) を用いてください。cXML 文書は、送られた外部の部分への参照を multipart MIME エンベロープ内に含みます。

### 2.1.5.1 添付の包含

このエンベロープに関する cXML 条件 (IETF RFC 2046 「多目的インターネット・メール拡張 第 2 部 メディア型」で記述された条件) は、各々の添付されたファイルの持つ Content-ID ヘッダーの包含です。

含まれる URL は、大きな伝送の範囲内で参照される添付のための識別子である cid を伴い始めなければなりません。cid:identifier は、転送中の文書を含んでいる MIME 伝送の一部 (かつ唯一) の Content-ID ヘッダーと一致しなければなりません。

次の例は、JPEG 画像を添付した cXML 文書の必須な骨組みを示します。

```
POST /cXML HTTP/1.0
Content-type: multipart/mixed; boundary=唯一の何か
--唯一の何か
Content-type: text/xml; charset="UTF-8"
<?xml version="1.0" encoding="UTF-8"?>
...
<Attachment>
  <URL>cid:uniqueCID@sender.com</URL>
</Attachment>
...
--唯一の何か
Content-type: image/jpeg

Content-ID: <uniqueCID@sender.com>
...
--唯一の何か--
```

この骨組みは、受け取っている MIME パーザーが処理できなければならないすべてです。骨組みが強化される場合、RFC 2387「MIME 多くの部分に分かれた・関連した内容型 [The MIME Multipart/Related Content-type]」で記述されたメディア型を使うアプリケーションは、より多くの情報を得るでしょう。

```
POST /cXML HTTP/1.0
Content-type: multipart/related; boundary=唯一の何か;
type="text/xml"; start=<uniqueMainCID@sender.com>
--唯一の何か
Content-type: text/xml; charset="UTF-8"
Content-ID: <uniqueMainCID@sender.com>
<?xml version="1.0" encoding="UTF-8"?>
...
<Attachment>
  <URL>cid:uniqueAttachmentCID@sender.com</URL>
</Attachment>
...
--唯一の何か
Content-type: image/jpeg
Content-ID: <uniqueAttachmentCID@sender.com>
...
--唯一の何か--
```

multipart/related メディア型を理解しない、受け取っている MIME パーザーは、上記の二つの例を同じく扱わなければなりません。さらに、MIME 伝送の各部分には Content-transfer-encoding を持て、その符号化を利用できます。この追加は、HTTP 伝送には必要ではありません。Content-description と Content-disposition のヘッダーは有用な文書化を与えますが、それらは、cXML プロトコルの範囲内で任意です。

## 2.1.5.2 添付の例

次の例は、カタログを添付した CatalogUploadRequest を示します。

```

POST /cXML HTTP/1.0
Content-type: multipart/related; boundary=kdf1kajfdksadjfk;
type="text/xml"; start="<part1.PC028.975@saturn.workchairs.com>"
--kdf1kajfdksadjfk
Content-type: text/xml; charset=UTF-8
Content-ID: <part1.PC028.975@saturn.workchairs.com>
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML timestamp="2000-12-28T16:56:03-08:00" payloadID="12345666@10.10.83.39">
  <Header>
    <From>
      <Credential domain="DUNS">
        <Identity>123456789</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="DUNS">
        <Identity>123456789</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
    </Sender>
  </Header>
  <Request>
    <CatalogUploadRequest operation="new">
      <CatalogName xml:lang="en">Winter Prices</CatalogName>
      <Description xml:lang="en">premiere-level prices</Description>
      <Attachment>
        <URL>cid:part2.PC028.975@saturn.workchairs.com</URL>
      </Attachment>
    </CatalogUploadRequest>
  </Request>
</cXML>

```

```
--kdf1kajfdksadjfk
Content-type: text/plain; charset=US-ASCII
Content-Disposition: attachment; filename=PremiereCatalog.cif
Content-ID: <part2.PC028.975@saturn.workchairs.com>
Content-length: 364

CIF_I_V3.0
LOADMODE: F
CODEFORMAT: UNSPSC
CURRENCY: USD
SUPPLIERID_DOMAIN: DUNS
ITEMCOUNT: 3
TIMESTAMP: 2001-01-15 15:25:04
DATA
942888710,34A11,C11,"Eames Chair",11116767,400.00,EA,3,"Fast MFG",,,400.00
942888710,56A12,C12,"Eames Ottoman",11116767,100.00,EA,3,"Fast MFG",,,100.00
942888710,78A13,C13,"Folding Chair",11116767,25.95,EA,3,"Fast MFG",,,25.95
ENDOFDATA
--kdf1kajfdksadjfk--
```

Content-ID か Content-Type ヘッダーの ID を「<」と「>」で囲んでください。しかし、URL 要素で ID を参照する場合、これらのブラケットを無視してください。

cid URL の特殊文字は、16 進数へ符号化 (%hh 形式) されなければなりません。

cXML 文書にテキストファイル、PDF、画像、その他の文書を添付する場合は、Attachment 要素を使ってください。別の cXML 文書を添付している場合、cXML 文書が添付それ自身を含むかどうかにかかわらず、cXMLAttachment 要素を使ってください。cXMLAttachment 要素は、受け取るシステムに敏感になるように対応します。追加の cXML 処理で添付を扱うことを要求されるかもしれません。

次の例は、cXMLAttachment 要素を使って添付を伴う cXML 文書を転送している CopyRequest を示します。CopyRequest に関するより詳細な情報については第 6 章の「CopyRequest」を参照してください。

```
Content-Type: Multipart/Related; boundary=outer-boundary
[Other headers]
--外側の境界--
Content-Type: text/xml; charset=UTF-8
Content-ID: <111@sendercompany.com>
[Other headers]
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML version="1.0" payloadID="123@sendercompany.com"
  timestamp="2003-11-20T23:59:45-07:00">
  <Header>
    <From>
      <!-- 送り手 -->
      <Credential domain="AribaNetworkUserId">
        <Identity>sender@sendercompany.com</Identity>
      </Credential>
    </From>
    <To>
      <!-- 受取人 -->
      <Credential domain="AribaNetworkUserId">
        <Identity>recipient@recipientcompany.com</Identity>
      </Credential>
    </To>
    <Sender>
      <!-- 送り手 -->
      <Credential domain="AribaNetworkUserId">
        <Identity>sender@sendercompany.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Sender Application 1.0</UserAgent>
    </Sender>
  </Header>
  <Request deploymentMode="production">
    <CopyRequest>
      <cXMLAttachment>
        <Attachment>
          <URL>cid:222@sendercompany.com</URL>
        </Attachment>
      </cXMLAttachment>
    </CopyRequest>
  </Request>
</cXML>
```

```
</cXMLAttachment>
</CopyRequest>
</Request>
</cXML>
--外側の境界--
Content-Type: Multipart/Related; boundary=inner-boundary
Content-ID: <222@sendercompany.com>
[Other headers]
--内側の境界--
Content-Type: text/xml; charset=UTF-8
Content-ID: <333@sendercompany.com>
[Other headers]
[Forwarded cXML]
--内側の境界--
[Attachment 1 of the forwarded cXML]
--内側の境界--
[Attachment 2 of the forwarded cXML]
--inner-boundary--
--外側の境界--
```

### 2.1.5.3 MIME についてのさらなる情報

さらに詳細な MIME 標準の情報については、次のウェブ・サイトを参照してください。

[www.hunnysoft.com/mime](http://www.hunnysoft.com/mime)

[www.ietf.org/rfc/rfc1341.txt](http://www.ietf.org/rfc/rfc1341.txt)

[www.ietf.org/rfc/rfc2046.txt](http://www.ietf.org/rfc/rfc2046.txt)

[www.ietf.org/rfc/rfc2387.txt](http://www.ietf.org/rfc/rfc2387.txt)

購買注文に添付する外部ファイルのさらに詳細な情報については、第 5 章の「添付」を参照してください。

## 2.1.6 cXML エンベロープ

cXML 要素は、cXML 文書のルートであり、それはすべての他の要素を含みます。cXML 要素は、どの cXML トランザクションにも存在しています。次の例は、完全に指定された cXML 要素を示します。

```
<cXML xml:lang="en-US"
      payloadID=1234567.4567.5678@buyer.com
      timestamp="1999-03-31T18:39:09-08:00">
```

cXML は、次の属性を持っています。

version (推奨されない)	この属性は、cXML 1.2.007 で反対されましたので、新しい cXML 文書では使ってはなりません。 cXML プロトコルのバージョンを指定してください。 検証する XML パーザーは、参照された DTD から version 属性を確定できます。このバージョン番号も cXML 文書で SYSTEM 識別子に現われるので、この属性は除外すべきです。
xml:lang (任意)	すべての自由なテキストのために利用される場所は、この文書の範囲内で送られます。受け手は応答するか、同じか類似した場所で情報を表示すべきです。 例えば、要求で「xml:lang="en-UK"」を指定しているクライアントは応答で「en」データを受け取るかもしれません。 可能な限りの説明的で具体的な場所を指定してください。
payloadID	失われたか、問題があったかもしれない文書を識別するためにログ目的で用いられる空間と時間に関するユニークな番号。 この値は再度試みるために変わるべきではありません。 勧められている実現は、次のとおりです。 datetime.process id.random number@hostname
timestamp	ISO 8601 形式によるメッセージが送信された日時。 この値は再度試みるために変わるべきではありません。 その形式は、次のとおりです。 YYYY-MM-DDThh:mm:ss-hh:mm (例：1997-07-16T19:20:30+01:00)
signatureVersion	存在する場合、デジタル式で署名された文書であることを暗示します。つまり、Request 要素か Response 要素か Message 要素の直

	後に妥当な ds:Signature 要素を含む文書です。 属性のための唯一の妥当な値は 1.0 であり、他の値は、将来の使用のために予約されています。 より詳細な情報については、第 16 章「cXML のデジタル署名」を参照してください。
--	--

### 2.1.6.1 xml:lang によって指定された場所

xml:lang 属性は、Description と Comments のような、ほとんど自由なテキスト要素で現れます。ある要素に関する場所を省略することを XML 仕様は認めていますが、そうした省略は、能率が悪い文書ツリーの問い合わせに終わります。cXML は影響を受けた文字列と共に locale 識別子を保とうと試みます。説明的で具体的な既知の場所がこの属性で指定されるべきです。

cXML プロトコルの全体にわたって現れる xml:lang 属性は、数値、日付、時間のよう な書式化されたデータに対してどのような効果も持ちません。次の節での timestamp 属性の記述のように、timestamp 属性の分離した値は、それらのデータ型にしたがって書式化されます。機械処理を意図していない長い文字列（参照されたウェブ・ページ）は、近くの xml:lang 属性に一致する場所特性の数か日付の形式を含むかもしれません。

## 2.1.6.2 日付, 時間, その他のデータ型

cXML における timestamp 属性とすべてのその他の日時は, ISO 8601 の制限された部分集合で書式付けされなければなりません。これは, W3C (Word Wide Web Consortium) の「日時の形式 (Date and Time Formats)」という表題の覚え書きで記述されており, 「[www.w3.org/TR/NOTE-datetime-970915.html](http://www.w3.org/TR/NOTE-datetime-970915.html)」で利用可能です。

timestamp 属性は, 年月日, 時間, 分, 秒を組み合わせた最小限で完全なデータを必要とします。秒の端数は任意です。このプロトコルは, UTC (世界時間。グリニッジ標準時間として知られている) から時間帯オフセットをでローカルな時間で表現された時間を必要とします。時間帯指示子「Z」は許されません。

例えば, `2002-04-14T13:36:00-08:00` は `April 14, 2002, 1:36 p.m., U.S. Pacific`

`Standard Time` に対応します。

cXML で用いられる日時とその他のデータ型の形式のさらに詳細な参照先は, 次のとおりです。

- ・ Microsoft 社の XML データ型リファレンス  
[msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk30/htm/xmrefxmldatatypes.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk30/htm/xmrefxmldatatypes.asp)
- ・ W3C に対する元の XML Data 草案  
[www.w3c.org/TR/1998/NOTE-XML-data-0105](http://www.w3c.org/TR/1998/NOTE-XML-data-0105)

## 2.1.7 特殊文字

cXML では、XML と同様に、すべての文字をキーボードからタイプできません。例えば、登録商標の記号のような文字はタイプできません。また「<」と「&」のような文字も XML で特殊な意味を持っています。それらの文字は、文字実体を使って符号化されなければなりません。

XML は、次のような組込みの文字実体を定義しています。

実体参照	文字実体
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

符号化の外側にある文字については、その文字の Unicode 番号（「#」に続けて 10 進数か 16 進数で表わす）を用いてください。例えば、「&#174;」と「&#xAE;」は、登録商標の記号を表わします。

例えば、

```
<Description xml:lang="en-US">The best prices for software®</Description>
```

は、

```
<Description xml:lang="en-US">The best prices for software&#174;</Description>
```

のように符号化できます。

一重引用符「'」や二重引用符「"」は、属性値の内側ではそれを区切り子として使うことを避けなければなりません。内容が引用符を含まないのであれば、属性を区切るために一重引用符だけを使うことが勧められています。

文書で特殊文字を扱うために、

1. 属性を区切るために一重引用符を使うだけのテンプレートを使ってください。
2. 次の一つを行なうことによってテンプレートに値を加えてください。

\* 文書が `cxml-urlencoded` で符号化された見えないフィールドによって伝送される `PunchOutOrderMessage` であれば、US-ASCII 符号化を使っているテンプレ

ートの値を満たしてください。この符号化は、その符号化を超えるすべての文字のための XML 文字実体を必要とします。例えば、前述したように、US-ASCII で利用できない登録商標の記号は「&#174;」として入力してください。

- \* さもなければ、UTF-8 符号化を使っている文書の中の値を満たしてください。UTF-8 は、HTTP Post によって直接送られたり、cXML-base64 で符号化された見えないフィールドに埋め込まれたりする、すべての文書のために利用されるべきです。UTF-8 は、すべての US-ASCII を含みます。

3. あなたが cXML 文書を作るように、XML は属性値と要素の内容をエスケープします。エスケープしなければならない文字は「&」, 「'」, 「<」, 「>」です。

あなたが PunchOutOrderMessage で文書を伝送する場合、次のステップが必要とされます。

4. ブラウザーが解釈するすべての文字に注意を払ってください。
  - a. あなたが cxml-urlencoded で符号化された見えないフィールドを使っているならば、すべての二重引用符を「&#34;」に変えてください。
  - b. さらに、cxml-urlencoded で符号化されたフィールドのために、HTML にとって重要な意義を持つ文脈に現われるすべてのアンパサンド「&」を「&amp;」でエスケープしてください。安全のために、すべてのアンパサンド「&」をエスケープできます。例えば、「&amp;」は「&amp;amp;」のように、「&apos;」は「&amp;apos;」のようにエスケープしてください。登録商標の記号「&#174;」は「&amp;#174;」のようにエスケープしてください。
  - c. さもなければ、あなたが cxml-base64 で符号化された見えないフィールドを使っているならば、base64 は cXML 文書全体を符号化します。
5. 文字列値のまわりを二重引用符で囲み、HTML フォームに文書に埋め込んでください。例えば、内容が「©®'""&<>>」で、属性の値が「©®'""&<>>」である Money 要素を送るために、XML 文書は次のように現れるかもしれません。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Money SYSTEM 'SpecialChars.dtd'>
<Money alternateAmount='&#174;#xAE;'&apos;""&#34;#quot;&amp;&lt;>>';'>
  &#174;#xAE;'&apos;""&#34;#quot;&amp;&lt;>>';'>
</Money>
```

は、次のように符号化されるべきです。

```
<!-- Recommendation for cXML-urllencoding: Uses double quotes to delimit the -->
<!-- field value and single quotes for the contained attributes. -->
<Input type="Hidden" name="cXML-urllencoded" value="<?xml version='1.0'
encoding='UTF-8'?>
<!DOCTYPE Money SYSTEM 'SpecialChars.dtd'>
<Money alternateAmount='&#174;&#174;&#xAE;&#apos;#34;&#34;
&#quot;&#amp;&#amp;lt;&#gt;&#gt;'&#174;&#174;&#xAE;'&#apos;
&#34;&#34;&#quot;&#amp;&#amp;lt;&#gt;&#gt;'/>
<!-- Best choice: Base64 encode the value. Don't have to worry about what -->
<!-- the browser interprets. -->
<Input type="Hidden" name="cXMLbase64"
value="PD94bWwgdMVyc2lrbj0nMS4wJyBlbmNvZGluZz0nVVRGLTgnPz4K
PCFET0NUWVBFIE1vbmV5IFNZU1RFTSAnU3B1Y2lhbENoYXJzLmR0ZCc+CjxNb
25leSBhbHRlcm5hdGVBbW91bnQ9JyYjMTc00yYjeEFF0yZhcG9z0yImIzM00yZxd
W900yZhbXA7Jmx00z4mZ3Q7Jz4KJiMxNzQ7JiN4QUU7JyZhcG9z0yImIzM00yZx
dW900yZhbXA7Jmx00z4mZ3Q7PC9Nb25leT4K">
```

前述の例は、cXML-urllencoded フィールドの符号化に代わるものを説明しています。それらは、すべての文脈において XML に特有でない二、三の文字、例えば、「<」と「>」をエスケープしている XML を避けます。前のステップの直接の実現は、次のような HTML フィールドに結果としてなるでしょう。

```
<Input type="Hidden" name="cXML-urllencoded" value="<?xml version='1.0'
encoding='UTF-8'?>
<!DOCTYPE Money SYSTEM 'SpecialChars.dtd'>
<Money alternateAmount='&#174;&#174;&#apos;#34;&#34;&#34;
&#amp;&&#amp;lt;&#gt;&#gt;'&#174;&#174;'&#34;&#34;&#34;
&#amp;&&#amp;lt;&#gt;&#gt;'/>
```

または、次のような XML 文書となります。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Money SYSTEM 'SpecialChars.dtd'>
<Money alternateAmount='&#174;&#174;&#apos;""&#lt;&#gt;&#gt;'
&#174;&#174;'""&#lt;&#gt;&#gt;
</Money>
```

## 2.1.8 Header 要素

Header (ヘッダー) 要素は、アドレスと認証の情報を含みます。Header 要素は、具体的な Request か Response にかかわらず、cXML メッセージの本体内で同じです。アプリケーションは、要求者 [requestor] の同一性を必要としますが、同一性として与えられた情報が正しいという確証はありません。

次の例は、Header 要素を示します。

```
<Header>
  <From>
    <Credential domain="AribaNetworkUserId">
      <Identity>admin@acme.com</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="DUNS">
      <Identity>012345678</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="AribaNetworkUserId">
      <Identity>sysadmin@buyer.com</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>
    <UserAgent>Network Hub 1.1</UserAgent>
  </Sender>
</Header>
```

From 要素と To 要素は、SMTP メール・メッセージの From と To と同義です。それらはメッセージの論理的なソースと目的地です。Sender 要素は、HTTP 接続を開き、cXML 文書を送る当事者です。

Sender 要素は、Credential 要素を含みます。それは、送っている当事者を証明するために、受け取る当事者を認めます。

この Credential 要素は、公開鍵 [public-key] の、端末どうしをむすぶ [end-to-end] デジタル認証基盤 [digital certificate infrastructure] を必要することなしに強い認証を認めます。利用者の名前とパスワードだけは、Request を行なう送っている当事者を許すために、受け取る当事者によって発せられる必要があります。

文書が最初に送られた時、Sender 要素と From 要素は同じですが、cXML 文書が電子商取引ネットワーク・ハブを介して伝わる場合、Sender 要素は、現在送っている当事者を示すために変わります

### From 要素

From 要素は、cXML 要求の作成者 [originator] を識別します。

### To 要素

To 要素は、cXML 要求の目的地 [destination] を識別します。

### Sender 要素

Sender 要素は、HTTP 接続を開いた当事者を認証するために、受け取る当事者を許します。これは、受け取る当事者は、誰が仕事を行なうように頼んでいるかを証明しなければならないので、From 要素か To 要素の一つより強い認証の Credential 要素を含みます。

### UserAgent 要素

cXML 対話を行なっている UserAgent を表わしている原文の文字列。これは唯一の？ [per-product] 文字列で、理想的には？ [per-version] であるべきです。HTTP 対話のための UserAgent に類似しています。

## Credential 要素

Credential 要素は、身元確認 [identification] と認証値 [authentication value] を含みます。

Credential 要素は、次の属性を持ちます。

domain	信用証明書の種類を指定してください。この属性は、複合的な認証ドメインに関する信任の複合的な種類を含む文書を許します。Ariba Supplier Network 上で送られたメッセージについて、例えば、ドメインは、電子メール・アドレス、D-U-N-S 番号に関する DUNS、事前に割り当てられた ID に関する NetworkId を示すために、AribaNetworkUserId とすることができます。
type (任意)	市場からの要求または市場への要求は、市場とメンバー会社を From 要素か、To 要素と Credential 要素で確認します。この場合、市場のための信用証明書は、値を「marketplace」に設定した type 属性を使います。

Credential 要素は、Identity 要素と、任意な SharedSecret 要素か CredentialMac 要素を含みます。任意の認証 [authentication] 要素が当事者の同一性を実証するのに対して、Identity 要素は、Credential 要素が誰を表わすかを述べます。

## SharedSecret 要素

SharedSecret 要素は Sender 要素が要求者が識別するパスワードを持っている場合に利用されます。

注：一方通行の通信を介して送られた文書で認証要素を使ってはなりません。一方通行の伝送は、利用者のブラウザを介してルートを定めます。それで、利用者は Credential 要素を含む文書ソースを確認することができます。

## CredentialMac 要素

CredentialMac 要素は、Message Authentication Code (MAC) 認証方法のために利用されます。この認証方法は、信頼される第三者による共有された秘密 [shared secret] によってそれが証明されるということを、送り手が受け手に対して証明しなければならない状況において利用されます。直接のパンチアウト要求は、ネットワーク・コマ

ース・ハブを通り抜けることなく供給業者へ消費者から直接伝達できます。なぜならば、それが供給業者がそれを証明することを許す（ネットワーク・コマーシ・ハブによって生成された）MAC を含んでいるからです。

信頼される第三者は、MAC を計算して、Profile トランザクションを介して送り手へそれを伝送します。MAC は、送り手に対して不透明です（それは安全かつ非可逆です）。どのように信頼される第三者から送り手へ MAC が伝えられるかについては、第 3 章の「ProfileResponse」を参照してください。

受け手は、信頼される第三者として同じ入力を用いて MAC を計算し、cXML 文書で受けられた MAC とそれを比較します。二つの値が一致すれば、文書は確実です。

どのように MAC 値を計算するかについては、第 15 章の「Message Authentication Code(MAC)」を参照してください。

CredentialMac 要素は、次の属性を持ちます。

type	証明されているデータと認証のためにそれがフォーマット化された方法を識別します。 唯一サポートされる値は「FromSenderCredentials」です。
algorithm	データで用いた MAC アルゴリズムを識別します。 唯一サポートされる値は「HMAC-SHA1-96」です。
creationDate	MAC が生成した日付と時間を指定します。
expirationDate	この MAC がもはや妥当でない日付と時間を指定します。 受け手は、expirationDate の後に受け取られた MAC を拒絶しなければなりません。受け手は、期限満了になっていない MAC を任意に拒絶できます。例えば、受け手は、1 時間以内に期限が切れて無効になることが予定されている MAC を拒絶するかもしれません。

次の例は、CredentialMac 要素を含む Credential 要素を示します。

```
<Sender>
  <Credential domain="NetworkId">
    <Identity>AN9900000100</Identity>
    <CredentialMac type="FromSenderCredentials"
      algorithm="HMAC-SHA1-96"
      creationDate="2003-01-15T08:42:46-0800"
      expirationDate="2003-01-15T11:42:46-0800">
      MnXkusp8Jj0lw3mf
    </CredentialMac>
    <UserAgent>Procurement Application 8.1</UserAgent>
  </Credential>
</Sender>
```

## 複合的な Credential 要素

From 要素, To 要素, Sender 要素は, 各々任意に複合的な Credential 要素を含むことができます。複合的な信任状 [credentials] を供給する目的は, 異なるドメインを使って一つの組織を識別することです。例えば, 組織は, DUNS 番号と NetworkId 番号を一緒に含むことによって識別されるかもしれません。

受け手は, 受け手が識別するドメインですべての信任状を検証すべきです。そして, 識別されたドメインのどの信任状も受け手が知っている組織と一致しないならば, 受け手は文書を拒絶すべきです。

また, 同じ From セクションか To セクションか Sender セクションにおいてどの二つの信任状も異なる実体を参照するように見えるならば, 受け手は文書を拒絶すべきです。

From セクションか To セクションか Sender セクションにおける, 異なる値であるが同じドメインではない複合的な信任状がある場合, 受け手は文書を拒絶すべきです。

## 2.1.9 Request 要素

クライアントは操作のために要求を送信します。たった一つの Request（要求）要素が各々の cXM エンベロープ要素で許されます。それは、サーバーの実現を簡単にします。なぜならば、cXML 文書を読んでいる時にデマルチプレキシング [demultiplexing] が生じる必要がないからです。Request 要素は、ほぼどのような種類の XML データも含むことができます。

典型的な Request 要素は、次の通りです。

- ・ OrderRequest
- ・ ProfileRequest
- ・ PunchOutSetupRequest
- ・ StatusUpdateRequest
- ・ GetPendingRequest
- ・ ConfirmationRequest
- ・ ShipNoticeRequest
- ・ ProviderSetupRequest
- ・ PaymentRemittanceRequest

Request 要素は、次の属性を持ちます。

deploymentMode (任意)	要求がテスト要求か製造要求であるかどうかを示します。許される値は「production」（省略時）か「test」です。
Id	この属性は、デジタル署名のための目標として要素とすべてのその子要素を呼び出すために利用できます。より詳細なデジタル署名については、第 16 章「cXML Digital Signatures」を参照してください。

## 2.1.10 Response 要素

サーバーは、操作の結果をクライアントに知らせるために送信します。幾つかの要求の結果にはどのようなデータもないかもしれませんが、Response（応答）要素は、Status 要素のほかには何も含みません。Response 要素は、どのようなアプリケーション・レベルのデータも含むことができます。例えば、パンチアウトの間に、アプリケーション・レベルのデータは、PunchOutSetupResponse 要素に含まれます。

典型的な Response 要素は、次のとおりです。

- ・ ProfileResponse
- ・ PunchOutSetupResponse
- ・ GetPendingResponse

Response 要素は、次の属性を持ちます。

Id	この属性は、デジタル署名のための目標として要素とすべてのその子要素を呼び出すために利用できます。より詳細なデジタル署名については、第 16 章「cXML Digital Signatures」を参照してください。
----	--

## Status 要素

Status 要素は、要求操作の成功か、一時的な失敗か、永久的な失敗かを知らせます。

Status 要素は、次の属性を持ちます。

code	要求の状態コード。例えば、200 は、成功した要求を表わします。後述するコードの表を参照してください。
text	状態のテキスト。このテキストは、ログについての利用者可読性を助け、エラーについての公式に認められた英語の文字列です。
xml:lang (任意)	Status 要素におけるデータの言語。cXML 1.0 と互換性のある選択。cXML の将来のバージョンで要求されるかもしれません。

Status 要素の属性は、何が要求で起きたかを示します。例えば、次のようです。

```
<Status xml:lang="en-US" code="200" text="OK">
</Status>
```

Status 要素の内容は、要求者によって必要とされるどのようなデータでもよいので、エラーを記述すべきです。cXML 200/OK 状態コードについては、データは無いかもしれませんが。しかし、cXML 500/Internal Server Error 状態コードか他の類似したコードについては、実際の XML 解析エラーかアプリケーション・エラーが現われられることが強く勧められています。このエラーは、よりよい一方的なデバッグと相互運用性検査を許します。例えば、次のようです。

```
<Status code="406" text="Not Acceptable">
  cXML が検証できなかった。大きな問題だ！
</Status>
```

次の表は、cXML 状態コードの範囲を記述しています。

範囲	意味
2xx	成功。
4xx	永久的なエラー。クライアントは再試行すべきではありません。このエラーは、受け付けられていることからの要求を妨げます。
5xx	一時的なエラー。典型的な伝送のエラー。クライアントは再試行すべきです。勧められている再試行の回数は、1 時間の周期当たり 10 回です。最小限度で、1 時間当たり 6 回の再試行が勧められています。殺到する注文のように、高い優先的な要求では、あなたは再試行周期を増やしたいと思うかもしれません。

状態コードが cXML 200 の範囲（例えば、cXML 200/OK）になれば、サーバーは、追加の Response 要素（例えば、PunchOutSetupResponse 要素）を含むべきではありません。

大半の場合、cXML は HTTP 上に層を成しているので、HTTP 404/Not Found のような多くのエラーは、伝送によって扱われます。すべての伝送エラーは、一時的に扱われるべきで、あたかも cXML 500 の範囲の状態コードが受けられたかのように、クライアントは再試行すべきです。HTTP 404/Not found と HTTP 500/Internal Server Error の状態コードを含み、妥当な cXML 内容を含まないすべての HTTP の応答は、伝送エラーだと見なされます。その他の共通の伝送問題は、時間切れ、（接続拒否のような）TCP エラー、

(不明なホストのような) DNS エラーを含みます。Request 文書を解析している際の検証エラーは、通常 400 の範囲の cXML の永久的なエラー、好んで 406/Not Acceptable となります。

次の表は、起こりうる cXML 状態コードを含んでいます。

状態	テキスト	意味
200	OK	The server was able to execute the request or deliver it to the final recipient. The returned Response might contain application warnings or errors: the cXML Request itself generated no errors or warnings, however, this status does not reflect any errors or warnings that might be generated afterward by the application itself. You will receive no further status updates, unless an error occurs during later processing.
201	Accepted	The request has been accepted for forwarding by an intermediate hub, or has been accepted by its ultimate destination and not yet been examined. You will receive updates on the status of the request, if a mechanism to deliver them is available. As mentioned in "StatusUpdateRequest" on page 206, the client should expect later StatusUpdate transactions.
204	No Content	All Request information was valid and recognized. The server has no Response data of the type requested. In a PunchOutOrderMessage, this status indicates that the PunchOut session ended without change to the shopping cart (or client requisition).
280		The request has been forwarded by an intermediate hub. You will receive at least one more status update. This status could mean that the request was delivered to another intermediary or to the final recipient with 201 status, or that it was forwarded via a reliable non-cXML transport.
281		The request has been forwarded by an intermediate hub using an unreliable transport (such as email). You might receive status updates; however, if you do not received status updates, there is not necessarily a problem.

状態	テキスト	意味
400	Bad Request	Request unacceptable to the server, although it parsed correctly.
401	Unauthorized	Credentials provided in the Request (the Sender element) were not recognized by the server.
402	Payment Required	This Request must include a complete Payment element.
403	Forbidden	The user has insufficient privileges to execute this Request.
406	Not Acceptable	Request unacceptable to the server, likely due to a parsing failure.
409	Conflict	The current state of the server or its internal data prevented the (update) operation request. An identical Request is unlikely to succeed in the future, but only after another operation has executed, if at all.
412	Precondition Failed	A precondition of the Request (for example, a PunchOut session appropriate for a PunchOutSetupRequest edit) was not met. This status normally implies the client ignored some portion of a previous transmission from a server (for example, the operationAllowed attribute of a PunchOutOrderMessageHeader).
417	Expectation Failed	Request implied a resource condition that was not met. One example might be a SupplierDataRequest asking for information about a supplier unknown to the server. This status might imply lost information at the client or server.
450	Not Implemented	The server does not implement the particular Request. For example, PunchOutSetupRequest or the requested operation might not be supported. This status normally implies the client has ignored the server's profile.
475	Signature Required	The receiver is unwilling to accept the document because it does not have a digital signature.
476	Signature Verification Failed	The receiver is unable to validate the signature, possibly because the document was altered in transit, or the receiver does not support one or more algorithms used in the signature.
477	Signature Unacceptable	The signature is technically valid, but is not acceptable to the receiver for some other reason. The signature policies or

		certificate policies may be unacceptable, the type of certificate used may be unacceptable, or there may be some other problem.
--	--	---

状態	テキスト	意味
500	Internal Server Error	Server was unable to complete the Request.
550	Unable to reach cXML server	Unable to reach next cXML server to complete a transaction requiring upstream connections. An intermediate hub can return this code when a supplier site is unreachable. If upstream connections complete, intermediate hubs should return errors directly to the client.
551	Unable to forward request	Unable to forward request because of supplier misconfiguration. For example, an intermediate hub failed to authenticate itself to a supplier. Clients cannot rectify this error, but this error might be resolved before the client retries.
560	Temporary server error	For example, a server might be down for maintenance. The client should retry later.

カタログ更新と関連がある状態コードについては、第 12 章の「Response」を参照してください。

認識されたコードを受け取っている時、cXML クライアントは、それらのクラスに応じてそれらを扱わなければなりません。それゆえ、より古いクライアントは、すべての新しい 2xx コードを 200（成功）として、4xx コードを 400（永久的な失敗）として、5xx コードを 500（一時的なエラー）として扱うべきです。このふるまいは、相互運用性を失うことなく、cXML プロトコルのさらなる拡張とサーバー特性コードの両方について許します。

### 2.1.11 一方通行（非同期）モデル

Request-Response トランザクションと違って、一方通行メッセージ [One-Way messages] は、HTTP 伝送に制限されません。一方通行メッセージは、HTTP チャンネル（同期の Request-Response 型操作）が適切でない状況のためにあります。次の図は、A と B が Request-Response トランザクションの代わりにメッセージをどのように通信するか例を示します。



図 2.1.11 一方通行メッセージ（非同期）

この場合、次のようなシナリオになるでしょう。

1. A は、B が理解する伝送において cXML 文書をフォーマット化し符号化します。
2. A は、既知の伝送を使って文書を送ります。A は、B から応答が戻るのを積極的に待ちません（待てません）。
3. B は、cXML 文書を受け取って、それを伝送ストリームの外で復号化します。
4. B は、文書进行处理します。

一方通行モデルにおいて、A と B は、明白な Request-Response 答サイクルがありません。例えば、一方通行メッセージの間に、他の当事者からのメッセージが到着するかもしれません。そして他の対話が起こりえます。

また、一方通行のトランザクションを十分に指定するために、メッセージのために使われる伝送は、文書化されなければなりません。一方通行のアプローチを使う cXML トランザクションのために伝送と符号化が指定されます。一方通行を使うトランザクションについて共通する例は、PunchOutOrderMessage です。

一方通行メッセージは、Request-Response モデルに対して次のような類似した構造を持ちます。

```
<cXML>
  <Header>
    ヘッダー情報…
  </Header>
  <Message>
    メッセージ情報…
  </Message>
</cXML>
```

Header 要素は、それが Request-Response の場合に正確に扱われます。cXML 要素は、前述した cXML エンベロープと全く同一です。一方通行メッセージと Request-Response メッセージの間の相違を知る最も容易な方法は、(Request 要素か Response 要素の代わりに) Message 要素の存在です。次の節では、Message 要素についてより詳細に議論します。

一方通行メッセージの中の Header 要素は、送り手の信任状における共有された秘密情報を含むべきではありません。BuyerCookie を使って認証は行なわれます。これは Request-Response ヘッダーとは異なります。

## 2.1.12 Message 要素

Message 要素は、cXML メッセージですべての本文レベルの情報を持ち運びます。それは、Response 要素で見つけたそれと同じ任意の Status 要素を含むことができます。それは Request メッセージへの論理的な応答であるメッセージで用いられるでしょう。

Message 要素は、次のような属性を持ちます。

deploymentMode (任意)	要求がテスト要求か製造要求であるかどうかを示します。許される値は、「production」（製造）か「test」です。
inReplyTo (任意)	この Message がどの Message に対して応答するかを指定します。inReplyTo 属性の内容は、より早く受け取られた Message の payloadID になります。これは多くのメッセージを持つ双方向の対話を構成するために利用されるでしょう。
Id	この属性は、デジタル署名のための目標として要素とすべてのその子要素を呼び出すために利用できます。より詳細なデジタル署名については、第 16 章「cXML Digital Signatures」を参照してください。

inReplyTo 属性は、初期の Request 文書か Response 文書の payloadID も参照できます。Request-Response トランザクションが複合的な一方通行の相互作用を介して「対話」を始める時、最初のメッセージは、他の方向に行った最近の関連する Request か Response の payloadID を含むことができます。例えば、PunchOutOrderMessage を含んでいる Message 要素は、PunchOut セッションを開始させた PunchOutSetupRequest の payloadID を持っている inReplyTo 属性を含んでいるかもしれません。PunchOut 文書に含められた BuyerCookie は、inReplyTo 属性のそれに類似した機能を行いません。

### 2.1.13 伝送オプション

一方通行のメッセージのために用いられる共通の二つの伝送、すなわち HTTP と URL-Form-Encoding があります。これらは、今日明確に定義された、ただ二つの伝送です。今後、より多くのものがサポートされることになるでしょう。

#### HTTP

調達アプリケーションは、一方通行の HTTP 通信を使って情報を引っ張ります。一方通行の HTTP 通信を使うトランザクションの種類は「GetPendingRequest」で議論されます。伝送データを安全のために暗号化するので HTTPS が望まれます。

#### URL-Form-Encoding

URL-Form-Encoding は、遠隔のウェブ・サイトと調達アプリケーションの間で統合を可能にします。またそれは、インターネットを介して直接アクセス可能な消費者のシステムでリスニング・サーバー [listening server] の必要を避ける方法の役目を果たします。この伝送は、どのように PunchOutOrderMessage トランザクションが働くかを調べることによって最もよく理解されます。

遠隔のウェブ・サイトは、調達アプリケーションに対して cXML の PunchOutOrderMessage 文書を直接送りません。その代わりに、HTML の Form (フォーム) 要素の見えないフィールドとしてそれらを符号化して、PunchOutSetupRequest の BrowserFormPost 要素で指定された URL にそれらを位置にポストします。利用者が買物の後にウェブ・サイトで「チェックアウト」ボタンをクリックする時、ウェブ・サイトは、HTML の Form の送信 [Submit] として調達アプリケーションに対してデータを送ります。

次の図は、何が起こるかを説明しています。

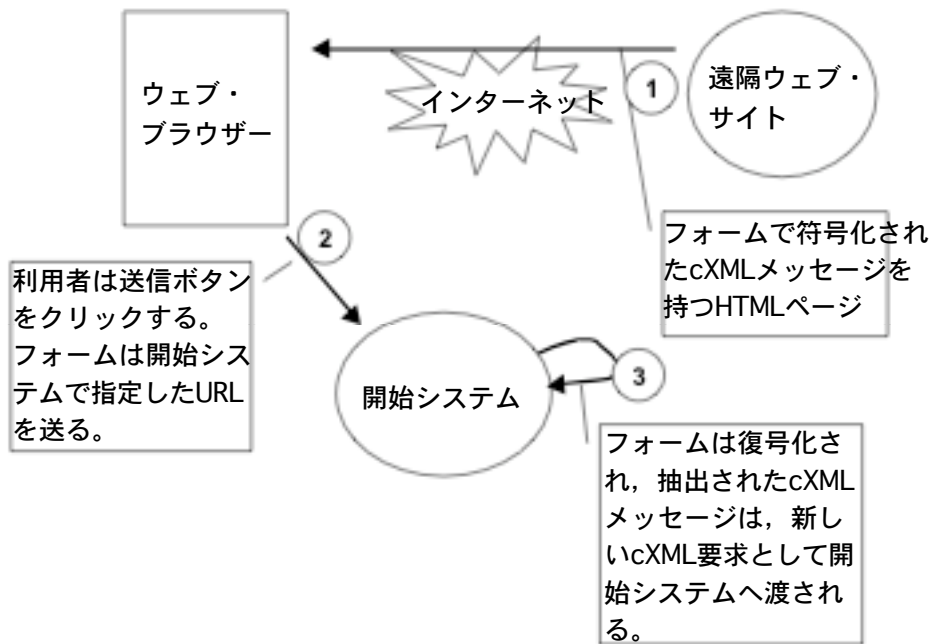


図 2.1.13 遠隔のウェブ・サイトと調達アプリケーションの間の動き

パックとアンパックの意味は、次に記述されます。

## Form 要素のパック

遠隔のウェブ・サイトは、cXML-urlencoded か cXML-base64 と名付けられた Form 要素の見えないフィールドに各々の PunchOutOrderMessage 文書を割り当てます。それらは、HTML の Form 要素に、METHOD 属性を ACTION とし、かつ ACTION 属性を PunchOutSetupRequest の BrowserFormPost 要素で渡された URL として割り当てます。例えば、次のようです。

```
<FORM METHOD=POST
  ACTION="http://workchairs.com:1616/punchoutexit">
  <INPUT TYPE=HIDDEN NAME="cXML-urlencoded"
    VALUE="Entire URL-Encoded PunchOutOrderMessage document">
  <INPUT TYPE=SUBMIT VALUE="Proceed">
</FORM>
```

ページ上の追加の HTML タグは、買物バスケットの内容の詳細を記述するために前述した部分を含むかもしれない。

注：ウェブ・サーバーが cXML-urlencoded フィールドを送信する時、それはまだ符号化された URL ではありません。Form がウェブ・ブラウザによって送信される（利用者が前述の例で「チェックアウト」をクリックする）時に、この符号化が必要とされます。ウェブ・ブラウザは、この条件に自ら応じます。ウェブ・ブラウザは、利用者のために Form を適切に表示するように、フィールドの値、エスケープする引用符、その他の特殊文字を HTML として符号化 [HTML-encode] しなければなりません。

cXML-urlencoded と cXML-base64 では、大文字と小文字を区別しません。

## cXML-urlencoded

cXML-urlencoded フィールドは、ウェブ・サーバーや供給業者によってでなく、ウェブ・ブラウザによって（HTTP 仕様によって）符号化された URL です。利用者が前述の例で「チェックアウト」をクリックする時のように、Form がウェブ・ブラウザによって送信される時だけ符号化が必要とされます。しかし、ウェブ・サーバーは、フィールドの値、エスケープする引用符、その他の特殊文字を HTML として符号化しなければなりません。それでフォームは正しく展示されるでしょう。

注：供給業者は、cXML-urlencoded フィールドを決して URL 符号化すべきではありません。このフィールドは、ウェブ・ブラウザによって自動的に URL 符号化されます。

cXML-urlencoded データに関して、受け取るパーサーは、メディア型「text/xml」の省略時値を超えて文字集合パラメーター [charset parameter] を仮定できません。POST されたデータについてのどのような文字符号化情報も HTTP POST で運ばれません。受け取るウェブ・サーバーは、見えないフィールド [hidden field] を含んでいる HTML ページの符号化を確定できません。それゆえ、この流儀で転送された cXML 文書は、us-ascii 文字符号化を使わなければなりません。XML ソース文書で見つけたどのような文字（「%XX」のように「URI 符号化」されたものも含む）も「us-ascii」集合になければなりません。その他の Unicode 記号はそのソース文書で文字実体を使って符号化できます。

## cXML-Base64

cXML-base64 の見えないフィールドは、国際的文書をサポートします。「us-ascii」外の記号を含んでいる cXML 文書は、cXML の URL として符号化 [cXML-urlencoded] された見えないフィールドの代わりにこのフィールドを使うべきです。その代替のものは、ほとんど同じ意味です。しかし、文書全体は、伝送を介して base64 で符号化され、受け取るサーバーに対しては HTML 符号化されず、ウェブ・ブラウザに対しては URL 符号化されます。Base64 符号化については、RFC 2045 の「Multipurpose Internet Mail Extensions (MIME) 第 1 部 インターネット・メッセージの本体」で記述されています。

ブラウザを介した遠隔ウェブ・サイトからクライアントの受け取るウェブ・サーバーへの Base64 符号化は、cXML 文書の元の文字符号化を保持します。いかなる文字集合パラメーターも POST で送られた情報で到着しないが、（伝送の符号化が取り除かれた後の）復号化された文書は、メディア型「application/xml」として扱うことができます。この符号化は、XML 宣言で指定されたどのような encoding 属性も守ることを受け取るパーサーにまかせます。（どのような application/xml 文書についても）このフィールドでは、省略時の文字符号化は UTF-8 です。

これらの見えないフィールド（cXML-urlencoded か cXML-base64）のどちらも、調達アプリケーションに POST で送られたデータに現われなければなりません。受け手はデータの cXML-base64 を最初に探すべきですが、両方のフィールドの送信は無駄です。

## Form 要素のアンパックと処理

適切な URL を前もって与えた調達アプリケーションは、前述したような Form データを含んでいる HTML Form POST を受け取ります。最初に、Form POST プロセッサは、cXML-base64 変数を探し、値を引き出し、そしてその内容を base64 で複合化します。そのフィールドがデータに存在しない場合、Form POST プロセッサは、cXML-urlencoded 変数を探し、URL 符号化された cXML メッセージを取り出し、そしてそれを URL 復号化します。それで、フィールドの複合化された内容は、まるでそれが正常な HTTP Request/Response サイクルを通して受け取られたかのように処理されます。

復号化後の文書の暗黙的なメディア型は、異なる可能な文字符号化を持ち、次のように異なります。

- cXML-urlencoded 変数は、charset 属性を持たずメディア型「text/xml」です。

従って、それは us-ascii 文字符号化に制限されます。ブラウザが符号化を変えるかもしれないので、受け取るパーサーは、cXML 文書の XML 宣言のどのような encoding 属性も無視しなければなりません。

- ・ cXML-base64 変数は、メディア型「application/xml」であり、したがって、どのような文字符号化（たとえ、含まれた XML 宣言の encoding 属性によって示されるとしても）どのような application/xml 文書に関しても、省略時文字符号化は UTF-8 です。

このトランザクションと正常な Request-Response トランザクションの間の主要な違いは、それを送るどのような HTTP 接続もないので、生成されたどのような応答もないことです。

## 2.1.14 サービス状態応答

このトランザクションは、特定のサービスが現在利用できるかどうかを決めます。HTTP GET がサービス場所に送られる時、そのサービスは、妥当な、動的に生成された cXML Response 文書で反応します。サービスは、cXML Request 文書が受け取られるどのような HTTP URL でもかまいません。

例えば、HTTP GET が `https://service.ariba.com/service/transaction/cxml.asp` へ送信すると、次のような応答が生じます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE cXML "http://xml.cXML.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML timestamp="2001-01-08T10:47:01-08:00"
      payloadID="978979621537--4882920031100014936@206.251.25.169">
  <Response>
    <Status code="200" text="OK">Ping Response Message</Status>
  </Response>
</cXML>
```

注：伝送（HTTP）とプロトコル（cXML）レベルのこの組み合わせは、前述した場合にだけ利用されるべきです。

## 2.2 基本要素

次の実体と要素は、cXML 仕様の全体にわたって用いられます。ここで列挙された定義のほとんどは、記述された高度なビジネス文書 [higher-order business document] が持つ基本的なボキャブラリーです。最も低いレベルのオブジェクトを表わしている共通の型実体と共通の要素は、ここで定義されます。

## 2.2.1 型実体 Type Entities

これらの定義のほとんどは、ワールドワイド・ウェブ・コンソーシアム (W3C) に対する XML-Data 覚え書提案からのものです。また、ここで定義される二、三のより高いレベルの型実体は、XML-Data からのものではありません。これらの型は「2.1.6 cXML エンベロープ」で議論されています。

### isoLangCode

ISO 639 標準からの ISO 言語コード。

### isoCountryCode

ISO 3166 標準からの ISO 国コード。

### xmlLangCode

XML 1.0 ([www.w3.org/TR/1998/REC-xml-19980210.html](http://www.w3.org/TR/1998/REC-xml-19980210.html)) によって定義された言語コード。大半の場合、これは ISO 639 の言語コードと、(任意に) ハイフンで区切られた ISO 3166 の国コードを含みます。完全な XML 勧告と違って、IANA か非公式の言語コードは、cXML で利用すべきではありません。IANA と非公式のサブコードは、妥当な ISO 3166 の国コードの後に来るべきですが、許されます。

勧められる cXML 言語コードの形式は、xx[-YY[-zzz]\*]? です。ここで、xx は ISO 639 の言語コード、YY は ISO 3166 の国コード、zzz は IANA か、質問における言語についての非公式のサブコードです。また、国コードの使用は常に勧められています。取り決めによって、言語コードは小文字で、国コードは大文字です。これはコードの正しい一致のために必要とされません。

### UnitOfMeasure 要素

UnitOfMeasure 要素は、どのように製品が梱包されたり出荷されたりするかを記述します。それは Measure Common Codes の UN/CEFACT 単位に従わなければなりません。さらに詳細な UN/CEFACT コードについては、[www.unetrades.net](http://www.unetrades.net) を参照してください。

## URL

HTTP/1.1 標準によって定義された URL (Uniform Resource Locator) 。

## 2.2.2 共通要素

仕様の全体に使われるこれらの要素は、Name 要素と Extrinsic 要素のような一般的なものから、Money 要素のような具体的なものまでわたります。

### Money 要素

Money 要素は、三つの可能な属性、すなわち、currency、alternateAmount、alternateCurrency を持ちます。currency 属性と alternateCurrency 属性は、3 文字の ISO 4217 の通貨コードでなければなりません。Money 要素と alternateAmount 属性の内容は、数値であるべきです。例えば、次のようです。

```
<Money currency="USD">12.34</Money>
```

任意の alternateCurrency 属性と alternateAmount 属性は、代わりの通貨で数量を指定するために一緒に利用されます。これらは、euro のような通貨条件をサポートするために利用できます。例えば、次のようです。

```
<Money currency="USD" alternateCurrency="EUR" alternateAmount="14.28">  
  12.34  
</Money>
```

注：1000 の区切りとしてのコンマを任意に使えます。小数の区切りとしてコンマを使ってはなりません。

### *Country*

場所の国名を含みます。PostalAddress 要素によって含まれます。

### *CountryCode*

国のコードのための国際的 ITU ダイヤルコードを含みます。それは、その国へ達するために電話機のキーパッド上でエスケープ・コードの後に入力できます。Phone 要素と Fax 要素によって用いられます。

## Contact 要素

Contact 要素は、現在のトランザクションにとって重要な連絡先に関するどのような情報も含みます。例えば、次のようです。

```
<Contact>
  <Name xml:lang="en-US">Mr. Smart E. Pants</Name>
  <Email>sepants@workchairs.com</Email>
  <Phone name="Office">
    ...
  </Phone>
</Contact>
```