2022-06-30

cXML の使用

目次

cXML 使用許諾契約

重要: cXML 仕様 (以下「本仕様」という) をご使用の前に、この cXML 使用許諾契約 (以下「本使用許諾」という) を必ずお読みください。本仕様をご使用いただくことによって、お客様は本使用許諾条件に拘束されることに同意するものとします。お客様が本使用許諾の条件にご同意いただけない場合、お客様は本仕様をご使用になること、または本仕様にアクセスすることはできません。使用許諾供与者は、本契約の新バージョン (改訂版も含む) を cXML サイト (www.cxml.org) で随時公開することがあります。本仕様に関して本使用許諾の下に付与される権利は、お客様がダウンロードまたはアクセスした時点で有効だった本契約のバージョンの条件に従うものとします。

1. 公開性。cXML は、電子商取引を促進するためのオープン標準として設計および意図されています。お客様は、本標準を導入およびご使用になり、コメント、提言、提案を cXML.org 宛に自由に寄稿することができます。お寄せいただいたご意見は十分に検討させていただき、最終的に cXML に採用される可能性があります。
2. 使用許諾。本契約条件によって、使用許諾供与者は、お客様に対して、本仕様を使用するための無期限、非独占、使用料無料の、国際的な権利および使用許諾を、使用許諾供与者の知的財産権の下で供与します。この知的財産権の範囲は、(a) 変更されていない仕様の使用、複製、公開、および配布 (別のコンピュータプログラムの一部として配布する場合なども含まれますが、これに限定されません) を目的として、および (b) 本仕様に含まれるスキーマガイドラインに準拠したコンピュータプログラムの作成、配布、販売、または販売以外の譲渡を行うために本仕様 (本仕様に含まれる cXML タグおよびスキーマガイドラインも含む) を実装し使用する目的として、本仕様を実装するために必要なものとします。お客様が、変更されていない仕様を使用、公開、または配布する場合、仕様を「cXML」と称することができます。
3. 制限。お客様が本使用許諾条件に準拠しない場合、本使用許諾の下でのお客様の権利は、使用許諾供与者からの通告なく自動的に終了します。
4. 使用許諾供与者は、本仕様の資料および内容に含まれる可能性がある、その他すべての権利を明白に留保します。お客様は、すべての仕様に対するすべての権利、資格、および権益を使用許諾供与者が所有することを認識および同意するものとします。ただし、お客様が作成するコンピュータプログラムまたは関連ドキュメントを使用許諾供与者が所有することはありません。また、cXML が導出される基となった、XML の知的財産または Ariba 以外の知的財産を所有することもありません。お客様は、本仕様の実装または使用により使用許諾供与者またはその他の団体に対する必然的な侵害となる可能性がある知的所有権を、本仕様の前記の実装または使用に関して主張しないことに同意するものとします。ただし、お客様が本仕様を実装または使用したことにより知的所有権が侵害されたと主張するあらゆる団体 (使用許諾供与者も含む) に対しては、前記の主張を行わないというお客様の同意は適用されなくなります (お客様が本契約に違反したという主張の一部として、使用許諾供与者またはその他の団体が、知的所有権をお客様に対して主張している場合は除きます)。お客様が本仕様を公開、複製、または配布する場合は、本使用許諾を添付する必要があります。お客様が使用許諾供与者になんらかのコメントまたはご提案を提供され、使用許諾供与者がお客様のそのご意見に基づいて本仕様を変更したとしても、使用許諾供与者は、本仕様の変更バージョンを所有するものとします。
5. 無保証。仕様に関する、お客様によるいかなる使用も、お客様ご自身の責任において行われるということについて、お客様は認識および同意するものとします。本仕様は、「現状のまま」いかなる保証もなく使用に供されるものとします。使用許諾供与者およびそのサプライヤは、商業性の目的、特定な目的の適正および非侵害性の保証を含む、またそれに限定されることのない、明示的、法的、あるいは暗黙的ないかなる種類のすべての保証を否認します。本仕様の使用中止が、本仕様のお客様による使用に関連した唯一かつ排他的措置となります。
6. 責任の限定。法律に許可される責任限度において、いかなる状況においても、本使用許諾またはお客様の本仕様の使用に関連するいかなる損害 (付随的損害、特別損害、懲罰的損害、直接損害、間接損害、または結果的損害を含みますがこれらに限定されることなく) にも、申し立てが不法行為、規約、またはその他の責任に関する見解に基づいているとしても、またそのような損害が発生する可能性が使用許諾供与者に忠告されていた場合にも、使用許諾供与者はいかなる責任も負わないものとします。お客様の裁判権が上記の損害免責を認められない場合、本使用許諾

の下でのすべての損害に対する使用許諾供与者による損害賠償合計額は、最高 10 ドル ($10.00) であることに合意するものとします。6.政府機関のエンドユーザー。本仕様が合衆国政府に供給される場合、本仕様は FAR 52.227-19 条項で定義される「制限付コンピュータソフトウェア」に分類されます。本仕様への合衆国政府の権利は、FAR 52.227-19 条項で規定されているとおりです。

7. 法律条項の論争を除いて、本使用許諾はカリフォルニア州および適用される米国連邦法に準拠し作成、解釈されるものとします。本使用許諾に関連するいかなる法律行為または法的な処理は、カリフォルニア州サンフランシスコ郡、サンタクララ郡、サンマテオ郡の州法廷または連邦法廷で実施されるものとし、本使用許諾内のいずれの当事者もこれらの郡の人的裁判権に同意するものとします。正当な法管轄権を有する法廷が、規定およびその一部を施行不能と判断した場合、本使用許諾のそれ以外の部分は完全なる効力および法的強制力を有するものとします。
8. お客様は、本仕様の使用から生じるすべてのリスクを負うものとします。
9. 完全な契約。本使用許諾は完全かつ排他的な説明および当事者による相互理解を絶対的に統合したものであり、本使用許諾の内容に関連するこれまでのあらゆる書面および口頭による同意および伝達に優先し、それを取り消すものです。お客様は、お客様による本使用許諾の内容に対するいかなる重大な違反も、使用許諾供与者に修復至難な損害をもたらし、法的な改善措置が不十分であることを認めるものとします。したがって、すべての法的、または公平な改善措置に加えて、使用許諾供与者は本使用許諾の違反を是正するために必要な差し止めを求める権利を有しています。Ariba, Inc. が使用許諾供与者としてみなされます。
10. 通知。使用許諾供与者へ宛てられたいかなる通知も、書面で comments@cxml.org に送られる必要があります。

7-19-04

1 はじめに

本マニュアルでは、cXML (commerce eXtensible Markup Language) を使用して電子商取引に関連したデータをやり取りする方法について説明します。

対象者および前提条件 [6 ページ]

本マニュアルの文字スタイル [6 ページ]

1.1 対象者および前提条件

本マニュアルは、cXML を使用して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対象に作成されています。

cXML は汎用性のあるオープンスタンダードな言語で、以下のトランザクションで使用します。

- 電子商取引ネットワークハブ
- 電子製品カタログ
- パンチアウトカタログ
- 購買アプリケーション
- バイヤー
- サプライヤ
- 電子商取引サービスプロバイダ

読者は電子商取引の概念、HTTP インターネット通信規格、および XML 形式に関する実践的な知識が必要です。

本マニュアルでは、特定の購買アプリケーションまたはネットワークハブの使用方法については説明しません。

1.2 本マニュアルの文字スタイル

cXML 要素 (element) および cXML 属性は、モノタイプのフォントで表記します。これらの名前には大文字と小文字の区別があります。どちらの名前にも大文字と小文字が混在していますが、要素名は大文字で始まり、属性名は小文字で始まります。たとえば、`MyElement` は cXML 要素で、`myAttribute` は cXML 属性です。

次の表に、本マニュアルで使用している表記規則を示します。

書体または記号	意味	例
AaBbCc123	ユーザーが置換しなければならないテキストは等幅のイタリックで表記します。	http://*server*:*port*/inspector
AaBbCc123	ユーザーインターフェイスとして表示されるコントロール名、メニュー名、およびメニュー項目名	[ファイル] メニューから [編集] を選択します。

書体または記号	意味	例
`AaBbCc123`	ファイル名、ディレクトリ名、パラメータ、CSV ファイルのフィールド、コマンド行、およびコード例	`ProfileRequest` ドキュメントは、バイヤーからネットワークハブに送信されます。
AaBbCc123	マニュアルのタイトル	詳細については、『*Acme* 設定の概要』を参照してください。

2 cXML の概要

このセクションでは、電子商取引トランザクションで使用する cXML (commerce eXtensible Markup Language) について紹介します。

2.1 XML を実装した cXML

XML (eXtensible Markup Language) はメタマークアップ言語で、文書やデータの構造を記述します。また、XML はアプリケーション間でデータの受け渡しをする際の標準言語で、特にインターネットを経由した通信に使用します。

XML ドキュメントでは、タグと値が対になった形式のデータを記述します。次に例を示します。

```
<DeliverTo>Joe Smith</DeliverTo>
```

XML の構造は HTML (HyperText Markup Language) の構造と類似しています。HTML は SGML で定義されたもので、SGML は XML のベースとなったメタ言語です。しかし、XML ドキュメントではすべてのデータがその意味に基づいてタグ付けされているため、HTML ドキュメントと比較して、アプリケーションによるデータの使用と抽出が容易に行えます。XML にはデータ情報しか含まれていませんが、HTML にはデータ情報と表示方法に関する情報の両方が含まれています。

各 cXML ドキュメントは、XML DTD (文書型定義) に基づいて作成されます。DTD は、テンプレートとして機能することにより、要素 (element) の正しい順序や入れ子などといった、cXML ドキュメントの内容モデルと属性のデータ型を定義します。

cXML 対応の DTD ファイルは、www.cXML.org Web サイトで入手できます。

2.2 cXML の機能

cXML を使用することで、バイヤー企業、サプライヤ、サービスプロバイダ、および仲介組織は、オープンスタンダードな単一の言語による通信が可能となります。

企業間の電子商取引 (B2B e コマース) が成功するためには、汎用性が高く柔軟なプロトコルが鍵となります。cXML は明確で厳密に定義された言語で、電子商取引を目的として設計されました。cXML は大量の製品を扱うバイヤー企業やサプライヤに特に適しています。

cXML によるトランザクションは、ドキュメントで構成されます。これらのドキュメントは、定義済みのタグの間に値を記述する単純なテキストファイルです。cXML ドキュメントは、そのほとんどがビジネスで従来から使用されているハードコピーのドキュメントに類似しています。

最も一般的な cXML ドキュメントの種類は、次のとおりです。

- カタログ [9 ページ]
- パンチアウト [10 ページ]
- 注文書 [12 ページ]

以下の項で、これらの cXML ドキュメントについて説明します。

2.2.1 カタログ

カタログは、製品およびサービスの内容をバイヤー企業に伝達するためのファイルです。カタログには、サプライヤが提供する製品およびサービス、さらにその価格が記載されており、サプライヤからバイヤーに対して行う伝達手段の中心となるものです。

サプライヤがカタログを作成すると、調達アプリケーションを使用するバイヤー企業は、そのサプライヤが提供する製品およびサービスを参照し、購入できるようになります。調達アプリケーションはカタログを読み込み、調達アプリケーションのデータベースに保存します。バイヤー企業がカタログを承認すると、カタログコンテンツはユーザーが参照できるようになり、ユーザーは品目を選択して、購入申請を作成します。

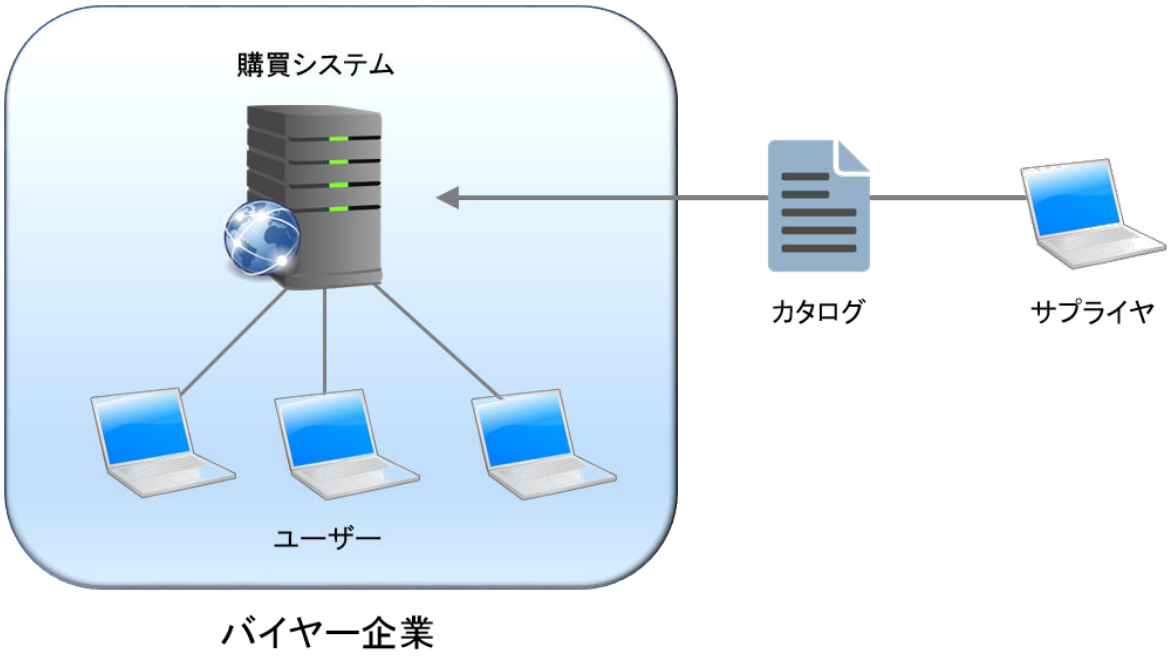


図 1: 製品およびサービスのコンテンツをバイヤー企業に送信

サプライヤは、数量単位、価格設定、配送方法に関係なく、どのような製品またはサービスに対してもカタログを作成できます。

カタログの各品目には、必須の基本情報と、複数の言語による説明などの高度なカタログ機能を使用する任意設定情報を記載できます。

2.2.2 パンチアウト

パンチアウトは、実装が容易な、インターネット経由で動作する対話型セッションのためのプロトコルです。パンチアウトでは、同期式の cXML メッセージを使用してリアルタイムでアプリケーション間の通信を行うため、リモートサイトの間でシームレスな対話が実現できます。

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 [10 ページ]
- パンチアウト連鎖 [11 ページ]
- プロバイダ パンチアウト [12 ページ]

調達パンチアウト

調達パンチアウトは、サプライヤにとって静的なカタログファイルに代わるものです。パンチアウトサイトは、Web サイト上で動作するリアルタイムの対話型カタログです。

電子商取引の Web サイトを運営するサプライヤは、サイトを変更してパンチアウトに対応させることができます。パンチアウトサイトは、cXML を使用して、インターネット経由で調達システムと通信します。

パンチアウトサイトの場合、調達アプリケーションでは、製品や価格設定の詳細が表示される代わりにボタンが表示されます。ユーザーがこのボタンをクリックすると、サプライヤの Web サイトのページがユーザーのブラウザに表示されます。サプライヤがこのページをどのように実装するかにより異なりますが、ユーザーは、製品の参照、設定内容の指定、配達方法の選択などを行うことができます。このページでの操作が完了したら、ユーザーはその注文情報を調達アプリケーションに転送するボタンをクリックします。購入する製品の構成や価格は、ユーザーの購入申請に表示されます。

次の例は、ユーザーとサプライヤ Web サイト間の対話型パンチアウトセッションを示しています。

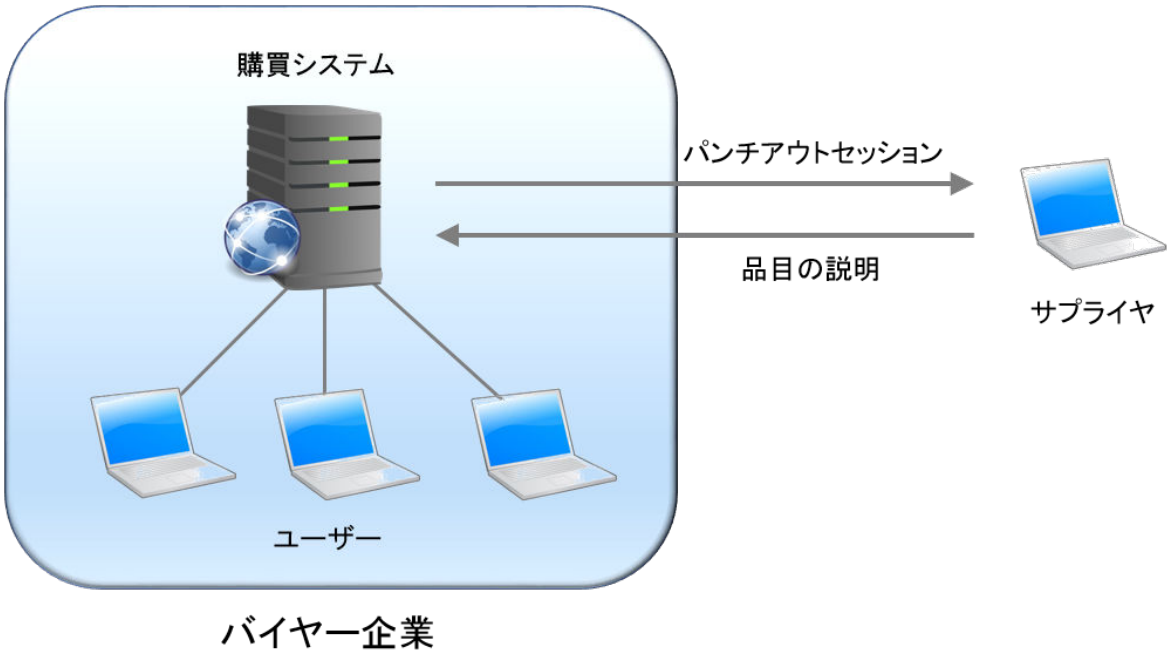


図 2: ユーザーとサプライヤ Web サイト間の対話型パンチアウトセッション

サプライヤの Web サイトには、あらかじめ合意した個別契約に基づく製品および価格が表示できます。

パンチアウト連鎖

パンチアウト連鎖とは、複数のパンチアウトがかかわっている調達パンチアウトのことです。cXML のパスルーティングにより、この機能が実現されました。

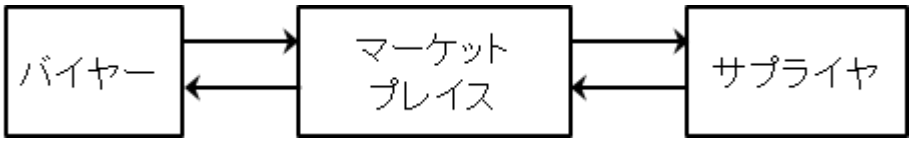


図 3: パンチアウト連鎖

cXML のパスルーティングを使用することにより、オーダーとそれに伴うメッセージを、見積り作成に関与するマーケットプレイスとサプライヤに返すことができます。パスルーティングは最終オーダーをすべての関係組織に通知し、それに続くパンチアウトは、マーケットプレイスに代わり、オーダーを分割する方法を調達アプリケーションに指示します。

プロバイダ パンチアウト

プロバイダ パンチアウトを使用すると、アプリケーションはリモートアプリケーションにパンチアウトすることができ、リモートアプリケーションはクレジットカードの検証、ユーザー認証、新規登録などのサービスをパンチアウトしたアプリケーションに提供できます。

2.2.3 注文書

バイヤー企業は、サプライヤに注文書を送信して契約の履行を要求します。

次の例は、サプライヤに送信される注文書を示しています。

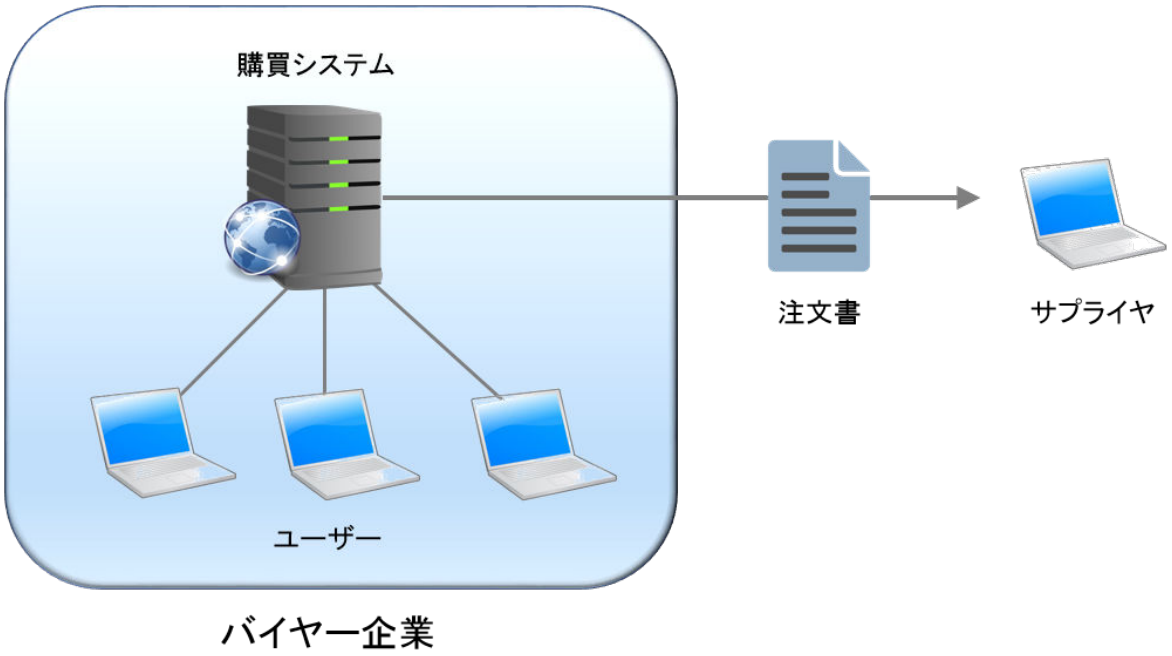


図 4: サプライヤに送信される注文書

cXML は、柔軟性があり、実装コストが低く、さまざまなデータや添付ファイルを広範にサポートしているため、注文書の受け渡し形式として、ほかの形式 (ANSI X12 EDI 850 など) よりも優れています。

2.3 cXML を使用す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使用できます。現在、バイヤー企業、さまざまな購買グループ、サプライヤ、およびアプリケーションベンダによって使用されています。以下の項では、現在 cXML を使用している主要なアプリケーションの種類について説明します。

購買アプリケーション

SAP Ariba Buying、SAP Ariba Buying and Invoicing、Ariba Buyer などの購買アプリケーションでは、外部トランザクションに cXML を使用しています。

購買コミュニティのユーザーは、購買マネージャにより承認されれば、これらのアプリケーションを使用して、契約に基づく製品およびサービスをサプライヤから購入できます。購入申請は、最初に購買コミュニティのマネージャによって承認され、承認済みの注文書は cXML ドキュメントとしてインターネット上でいくつかの仲介組織を介してサプライヤに転送されます。

ネットワークハブ

Ariba Network などのネットワークハブは、バイヤーとサプライヤを接続する Web ベースのサービスです。これらの Web サービスは、カタログの検証とバージョン管理、カタログの公開と受信登録、注文書の自動ルーティング、および注文書履歴などの機能を提供します。

ネットワークハブは、さまざまな組織間で受け渡しされる Request と Response を認証し、ルーティングすることにより、その仲介機能を果たします。これらの組織間の通信は、すべてインターネット上で cXML を使用して行われます。

パンチアウトカタログ

前項で説明したように、パンチアウトカタログはサプライヤの Web サイトで使用する対話型カタログです。パンチアウトカタログは、バイヤーのパンチアウトセッションを管理する ASP (Active Server Pages)、JavaScript、または CGI (Common Gateway Interface) などのプログラミング言語で記述された Web サーバーアプリケーションにより実現されます。

パンチアウトカタログは、購買アプリケーションからパンチアウト要求を受け取り、バイヤー企業を識別してから、該当する製品と価格を HTML フォーマットで表示します。次にユーザーは品目を選択して、設定を行い、必要であればオプションを選択します。

パンチアウトセッションの終了時に、パンチアウトサイトはユーザーが選択した内容を cXML フォーマットで購買アプリケーションに送信します。

オーダー受信システム

オーダー受信システムは、バイヤー企業から送信された注文書を受け付けて処理するサプライヤサイトのアプリケーションです。オーダー受信システムには、在庫管理システム、オーダーフルフィルメントシステム、またはオーダー処理システムなどのような自動システムの機能も持たせることができます。

cXML の注文書から情報を抽出するのは単純な処理のため、比較的容易に既存のオーダー受信システムにこれらの機能を付加することができます。

2.4 コンテンツの提供方法

調達アプリケーションは、自社のユーザーに対して製品およびサービスの内容を表示します。サプライヤの販売プロセスにおいて、バイヤーに対して製品やサービスをいかに表示するかは重要なため、顧客がどのように製品やサービスを参照するかに関して、サプライヤは制御したいと考えます。また、バイヤー企業は、契約に確実に準拠した上で、コンテンツへのアクセスと検索を容易に行えるようにしたいと考えます。

バイヤー企業とサプライヤは、製品とサービスの内容の表示に関して、複数の方法から選択できます。使用する個別の方法は、バイヤー企業とサプライヤとの間で達した合意と、取引される製品またはサービスの性質によって決定されます。

次の表に、一般的に調達される製品とサービスのカテゴリ例、およびそれらのコンテンツの好ましい提供方法を一覧表示します。

商品	特性	コンテンツの提供方法
事務用品、 社内用品	静的なコンテンツ、安定した価格	静的カタログ
ラボサプライ、 MRO (保守、修理、運用)、 電子部品	使い勝手のよい規格化が必要	垂直商品ポータルへのパンチアウト
書籍、 化学薬品	多品目	サプライヤが運営するサイトにパンチアウト
コンピュータ ネットワーク装置、 周辺機器	さまざまな構成が可能	サプライヤが提供する設定ツールにパンチアウト
サービス、 印刷物	高度に可変な属性を持つコンテンツ	サプライヤサイトの電子フォームにパンチアウト

バイヤー企業は、コンテンツを組織内にローカルに格納することも、パンチアウトを使用してインターネット経由でリモートコンテンツにアクセスすることもできます。cXML カタログは、その両方の手段をサポートしています。

この表が示すように、パンチアウトは柔軟性のある枠組みを提供します。この枠組みの中で、サプライヤは商品またはバイヤーに応じてカスタマイズされたコンテンツを提供できます。このような方法でコンテンツを取り扱う目的は、バイヤーとサプライヤが最も合理的な方法でカタログデータをやり取りできるようにすることです。

2.5 cXML DTD

cXML は XML 言語の 1 つであるため、文書型定義 (DTD) によって完全に定義されます。これらの DTD は、cXML 要素の構文と順序を明確に記述したテキストファイルです。アプリケーションは DTD を使用することにより、読み出したり書き込んだりする cXML データを検証できます。

各 cXML ドキュメントのヘッダーには、そのドキュメントを定義している DTD の URL が含まれます。cXML アプリケーションはその DTD を入手し、それを使用してドキュメントを検証することができます。

トランザクションを最も確実なものにするためには、受信したすべての cXML ドキュメントを検証します。エラーが検出された場合、適切なエラーコードを発行して、送信者が再送信できるようにします。cXML アプリケーションで、受信したすべて

の cXML ドキュメントを検証しなければいけないわけではありませんが、検証することを推奨します。しかし、すべての cXML ドキュメントは正しく、かつ以下で説明する cXML DTD に準拠している必要があります。

cXML DTD の入手方法

あらゆるバージョンの cXML に対応した DTD は、cXML.org から入手できます。さまざまな種類の cXML ドキュメントが複数の DTD で定義されています。これは、一部のパーサーで高速な検証を可能にするために、サイズを縮小した DTD を使用しているためです。

ドキュメント	DTD
基本 cXML ドキュメント	http://xml.cXML.org/schemas/cXML/*version*/cXML.dtd
確認と出荷通知	http://xml.cXML.org/schemas/cXML/*version*/Fulfill.dtd
請求書	http://xml.cXML.org/schemas/cXML/*version*/InvoiceDetail.dtd
タイプ定義	http://xml.cXML.org/schemas/cXML/*version*/Catalog.dtd
支払送金	http://xml.cXML.org/schemas/cXML/*version*/PaymentRemittance.dtd
見積依頼書	http://xml.cXML.org/schemas/cXML/*version*/Quote.dtd
契約	http://xml.cXML.org/schemas/cXML/*version*/Contract.dtd
物流	http://xml.cXML.org/schemas/cXML/*version*/Logistics.dtd

ここで、*version* は cXML の完全なバージョン番号です。

cXML アプリケーションでは、これらの DTD を使用して、送受信するすべてのドキュメントが検証できます。

DTD のキャッシュ

最高のパフォーマンスを得るためには、cXML アプリケーションは DTD をローカルにキャッシュする必要があります。cXML の DTD ファイルは公開後に変更されることはないため、期限を設定せずにキャッシュしておくことができます。(DTD は、更新されると新しい URL が割り当てられます。)cXML アプリケーションは、cXML ドキュメントの解析時にドキュメントヘッダーの SYSTEM 識別子を調べ、DTD がまだローカルに格納されていない場合は、その DTD を取得するようにします。

ローカルに DTD をキャッシュしておくと、ドキュメントの検証をより高速に行うことができ、cXML.org サイトへのアクセスも減少するという利点があります。

環境によっては、cXML アプリケーションが新しいドキュメントを受信する際に、自動的に DTD を取得することが許可されていない場合があります。これらの環境では、DTD を手動で取得してローカルに保存し、cXML.org ではなくローカルの DTD を参照するよう、アプリケーションを設定する必要があります。ただし、cXML ドキュメントを生成する場合は、ローカルの DTD ではなく cXML.org の DTD を参照するようにしておく必要があります。

2.6 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では、特定の cXML サーバーがどのトランザクションを受信できるかについての基本情報をやり取りします。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する必要があります。それは、アプリケーション間のバックエンド統合を行うためで、それにより cXML サーバーの機能がクライアントのシステムで使用可能になります。

このトランザクションは、`ProfileRequest` および `ProfileResponse` という 2 つのドキュメントで構成されます。この 2 つのドキュメントを組み合わせて使用し、サポートする cXML バージョンとトランザクション、およびそれらのトランザクションに対するオプションを含むサーバー機能を検索します。

> **i 注記**
>
> cXML 1.1 以降のすべてのサーバーは、プロファイルトランザクションを受け付けなければなりません。

ProfileRequest

この `ProfileRequest` ドキュメントにはコンテンツがありません。このドキュメントは、cXML サーバーにそのままルーティングされます。

ProfileResponse

サーバーは、`ProfileResponse` ドキュメントを使用して応答します。ProfileResponse ドキュメントには、cXML サーバーがサポートするトランザクション、その所在地 (URL)、および指定されたオプションを文字列で記述します。

2.7 サービス状況応答

送信された cXML を受け付けた URL から、状況コード 200 を持つ Response が作成されて、返信されます。HTTP GET がサービスサイトに送られると、サービスは動的に作成された有効な cXML Response ドキュメントで応答します。サービスの HTTP URL は、cXML Request ドキュメントを受信できればどこでも可能です。

2.8 XML ユーティリティ

XML ファイルを編集して検証するためのユーティリティは、Web 上で無償または有償で入手できます。以下は、これらいくつかのユーティリティについて説明したものです。

- **Internet Explorer** (Microsoft 社製)DTD を使用して XML ファイルの妥当性を検証することができる XML 対応の Web ブラウザ。
  www.microsoft.com/windows/ie/default.htm
- **Turbo XML** (TIBCO Software 社製) (注: 英語)XML 資産を作成、検証、変換、および管理するための統合開発環境 (IDE)。
  www.tibco.com/software/metadata/turboxml.jsp
- **XML Spy** (Altova 社製) (注: 英語)グリッド、ソース、およびブラウザ表示機能を備えた、DTD と XML ファイルを保守するためのツール。
  www.altova.com
- **XMLwriter** (Wattle Software 社製) (注: 英語)XML プロジェクトを管理するために設計されたグラフィカル XML オーサリングツール。
  www.xmlwriter.net

そのほかに、以下の Web サイトには、さらに多くの XML ツールが一覧表示されています (注: 英語)。

- www.xml.com
- http://www.ibm.com/developerworks/xml/

3 cXML の基本

このセクションでは、cXML の基本プロトコルとデータフォーマットについて説明します。この章には、すべてのトランザクションの実装に必要な情報が記載されています。

プロトコルの仕様 [18 ページ]

基本要素 [43 ページ]

3.1 プロトコルの仕様

cXML トランザクションには、Request/Response モデルと One-Way モデルの 2 つの通信モデルがあります。これら 2 つのモデルは操作が厳密に定義されているため、実装は容易に行えます。状況によっては 1 つのモデルで対応できない場合があるため、両方のモデルが必要となります。

3.1.1 Request/Response モデル

Request/Response トランザクションは、HTTP または HTTPS 接続でのみ実行できます。以下の図は、組織 A と B の間における Request/Response トランザクションの対話の手順を示しています。

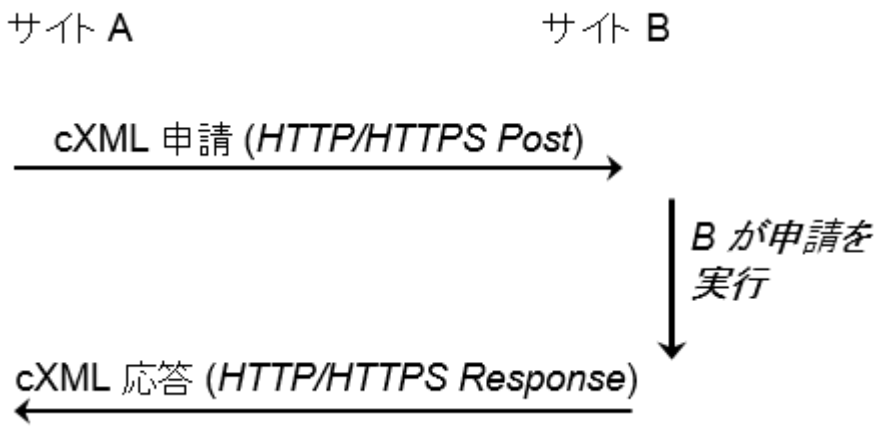


図 5: Request/Response トランザクション

このトランザクションは、以下の手順で処理されます。

1. サイト A は、サイト B のアドレスとして指定されている URL を参照して、サイト B との HTTP/1.x 接続を開始します。
2. サイト A は、POST 操作を行い、cXML ドキュメントを HTTP 接続経由で送信します。その後サイト A は、Response 待ちとなります。

3. サイト B の HTTP/1.x 準拠のサーバーは、手順 1 で参照された URL が示すリソースに HTTP `Request` を送信します。このリソースは、CGI プログラムや ASP ページなどで、サイト B の HTTP サーバーが認識できる有効なネットワーク所在地に存在する必要があります。
4. 手順 3 で識別されたサイト B のリソースは、cXML ドキュメントの内容を読み取り、その `Request` を適切なハンドラにマッピングします。
5. cXML `Request` に対応するサイト B のハンドラは、その `Request` に従って処理を行い、cXML `Response` ドキュメントを生成します。
6. サイト B は、手順 1 で確立した HTTP 接続により、cXML `Response` をサイト A に送信します。
7. サイト A は、cXML `Response` を読み取り、Request を発信したプロセスにその `Request` を返します。
8. サイト A は、手順 1 で確立した HTTP 接続を切断します。

さらに `Request/Response` サイクルを続行する場合は、このプロセスが繰り返されます。

上記手順の作業を単純化するために、cXML ドキュメントは次の 2 つの部分に分割されます。

- Header - アドレス指定と認証情報で構成されます。
- `Request` または `Response` データ - 特定の要求または応答、およびやり取りされる情報が含まれます。

これらの要素 (element) の両方とも、親エンベロープ要素に入れて転送されます。次の例は、cXML `Request` ドキュメントの構造を示しています。

```
<cXML>
    <Header>
        Header information
    </Header>
    <Request>
        Request information
    </Request>
</cXML>
```

次の例は、cXML `Response` ドキュメントの構造を示しています。

```
<cXML>
    <Response>
        Response information
    </Response>
</cXML>
```

`Response` 構造では、`Header` 要素が使用されません。`Response` は常に `Request` と同じ HTTP 接続で送信されるため、これは必要ありません。

3.1.2 cXML 変換

cXML では、要素を使用して、従来のビジネスドキュメントでのプロパティに当たる個々の項目を記述します。たとえば、住所が子要素である番地、市区町村、国で構成されるような場合、要素は、子要素が必要な情報やそれらの子要素間の関係も記述できます。

また、cXML は属性を使用して、要素の変更や内容の提供を行います。

要素や属性の名前では大文字と小文字が区別され、名前の中では大文字を使用して語句を区切り、ハイフンは使用しません。要素名は大文字で始まり、属性名は小文字で始まります。たとえば、次のようになります。

要素: `Sender, Credential, Payment, ItemDetail` 属性:`payloadID, lineNumber, domain`

必須ではない要素の内容が空の場合 (null の場合)、要素全体を削除します。これは、空の要素や空白文字の要素が一部のパーサーに影響を及ぼす恐れがあるためです。

DTD ファイルおよび cXML ドキュメントでは、トランザクション内で出現する要素の回数を示す記号が使用されます。「+」は、その要素が 1 回以上出現することを表し、「?」はその要素が 0 回または 1 回出現することを表し、「*」はその要素が 0 回以上出現することを表します。

3.1.3 cXML ドキュメント

`cXML` 要素は cXML ドキュメントのボディです。以下に、ドキュメントの先頭部分の例を示します。

```
<?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 ドキュメントの最初の文字は、`<?` または `<!` でなければなりません。ドキュメントを空白文字やタブで始めることはできません。たとえば、HTML フォームの `PunchOutOrderMessage` ドキュメントでは、始めの引用符と左山カッコの間に文字を挿入することはできません。

cXML ドキュメントの 2 行目には、DOCTYPE 文書型宣言を入れる必要があります。これは、cXML ドキュメントで設定する唯一の外部エンティティです。この行は cXML の DTD を参照します。

cXML ドキュメントには、最上位レベルの要素である `cXML`、`Supplier`、`Contract`、および `Index`。`cXML` 要素は、「トランザクション的」なデータの場合に使用します。その他の 3 つの要素では、静的なコンテンツを記述します。

関連情報

cXML DTD [14 ページ]

3.1.4 ラッピングレイヤ

cXML ドキュメントは、通常は HTTP ヘッダーを持ち、HTTP を使用して転送されます。この HTTP ヘッダーでは、`text/xml` という MIME (Multipurpose Internet Mail Extensions) メディアタイプを指定し、さらに cXML ドキュメントのエンコード方式を示す `charset` パラメータを指定します。

HTTP は 8 ビットクリーンであるため、受信パーサーによってサポートされる任意の文字エンコードが、`base64` や quoted-printable などの content-transfer encoding を必要とせずに使用できます。すべての XML パーサーは、US-ASCII を含むすべての Unicode 文字が入っている `UTF-8` (Universal Transformation Format) へのエンコードをサポートしています。そのため、cXML ドキュメントを転送する場合、アプリケーションでは UTF-8 を使用してください。

> **i 注記**
>
> IETF RFC2376 の「XML Media Types」に従い、MIME の `charset` パラメータは XML 宣言で指定されたエンコードよりも優先されます。さらに、`text/xml` メディアタイプの通常のエンコードは `us-ascii` であり、XML 仕様の第

> 4.3.3 項で述べられているように `UTF-8` ではありません。明確にするために、cXML ドキュメントの XML 宣言では明示的にエンコード方式を記述してください。MIME エンベロープでは、`text/xml` . の場合、適切な `charset` パラメータを使用する必要があります。また、application/xml メディアタイプも使用できます。このメディアタイプでは XML 宣言が優先され、また受信者側のデコードにも影響しません。さらにこのメディアタイプでは charset パラメータも指定する必要はありません。

cXML ドキュメントの HTTP 送信には、次の MIME と HTTP ヘッダーが含まれます。

```
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"?>
…
```

3.1.5 添付ファイル

cXML プロトコルでは、任意のタイプの外部ファイルを cXML ドキュメントに添付することができます。たとえば、バイヤーが注文書の内容を明確にするために、補足メモ、図面、または FAX を添付する場合があります。別の例として、カタログファイルを添付する `CatalogUploadRequest` ドキュメントがあります。

cXML ドキュメントで参照されるファイルは、受信者がアクセス可能なサーバー上に置くか、または cXML ドキュメント自身も含むエンベロープに入れることができます。cXML ドキュメントに外部ファイルを添付して 1 つのエンベロープにするには、MIME (Multipurpose Internet Mail Extensions) を使用します。cXML ドキュメントには、マルチパート MIME エンベロープで送信される外部パートへの参照が含まれます。

添付ファイルを含める

このエンベロープに対する cXML の必要条件 (IETF RFC 2046 の「Multipurpose Internet Mail Extensions Part Two: Media Types」で規定) として、`Content-ID` ヘッダーが添付ファイルごとに必要です。

cXML で指定する URL は、`cid:` で開始する必要があります。これはより大規模な送信で添付ファイルを参照するための識別子です。`cid:` 識別子は、送信されるドキュメントを含む MIME 送信の 1 パート (1 つのみ) の Content-ID ヘッダーと一致していなければなりません。

次の例は、JPEG 画像を添付する場合の cXML ドキュメントに必要なスケルトンを示します (ただし上述の HTTP ヘッダーは含んでいません)。

```
POST /cXML HTTP/1.0
Content-type: multipart/mixed; boundary=something unique
--something unique
Content-type: text/xml; charset="UTF-8"
<?xml version="1.0" encoding="UTF-8"?>
…
   <Attachment>
      <URL>cid:uniqueCID@sender.com</URL>
   </Attachment>
```

```
…
--something unique
Content-type: image/jpeg
Content-ID: <uniqueCID@sender.com>
…
--something unique--
```

さらに、このスケルトンには、受信 MIME パーサーで処理する必要があるすべてのものが含まれています。RFC 2387 の『The MIME Multipart/Related Content-type』で規定されているメディアタイプを使用するアプリケーションは、このスケルトンの内容を拡張することによって、より多くの情報を取得できます。

```
POST /cXML HTTP/1.0
Content-type: multipart/related; boundary=something unique;
    type="text/xml"; start=<uniqueMainCID@sender.com>
--something unique
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>
…
--something unique
Content-type: image/jpeg
Content-ID: <uniqueAttachmentCID@sender.com>
…
--something unique--
```

`multipart/related` メディアタイプを解読できない受信 MIME パーサーは、上記の 2 つの例を同様に処理する必要があります。MIME 転送の各パートに Content-transfer-encoding を追加して、そのエンコードを使用することもできます。HTTP 転送には、この追加は必要ありません。cXML プロトコルでは、Content-description ヘッダーと Content-disposition ヘッダーの設定は任意ですが、これらのヘッダーを使用すると文書の利用価値を高めることができます。

Attachment の例

次の例は、カタログを添付する `CatalogUploadRequest` を示します。

```
POST /cXML HTTP/1.0
Content-type: multipart/related; boundary=kdflkajfdksadjfk;
    type="text/xml"; start="<part1.PCO28.975@saturn.workchairs.com>"
<--! begin first MIME body part header -->
--kdflkajfdksadjfk
Content-type: text/xml; charset=UTF-8
Content-ID: <part1.PCO28.975@saturn.workchairs.com>
<--! end first MIME body part header -->
<?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>
            <!-- ID of MIME attachment follows -->
                <URL>cid:part2.PCO28.975@saturn.workchairs.com</URL>
            </Attachment>
        </CatalogUploadRequest>
    </Request>
</cXML>
<--! begin second MIME body part header -->
--kdflkajfdksadjfk
Content-type: text/plain; charset=US-ASCII
Content-Disposition: attachment; filename=PremiereCatalog.cif
Content-ID: <part2.PCO28.975@saturn.workchairs.com>
Content-length: 364
<--! end second MIME body part header -->
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
<!-- MIME trailer follows -->
--kdflkajfdksadjfk--
```

`Content-ID` または `Content-Type` ヘッダー内の ID は山カッコ (< >) で囲みます。ただし、`URL` 要素内で ID を参照する場合は、これらのカッコを省略します。また、`URL` 要素内でメッセージ ID の前に `cid:` を付加します。ただし、MIME ヘッダー内では付加しません。

cid: URL の特殊文字は、16 進コード (`%hh` フォーマット) にする必要があります。

テキストファイル、PDF、画像などを cXML ドキュメントに添付する場合は、`Attachment` 要素を使用します。ほかの cXML ドキュメントを添付する場合は、`cXMLAttachment` を使用します。その cXML ドキュメントにさらに添付ファイルが含まれるかどうかは関係ありません。`cXMLAttachment` 要素は、添付ファイルを処理するために追加の cXML 処理が必要であることを受信システムに通知します。

次の例は、`cXMLAttachment` を使用してファイルを添付した cXML ドキュメントを転送する `CopyRequest` を示します。

```
Content-Type: Multipart/Related; boundary=outer-boundary
[Other headers]
--outer-boundary
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 payloadID="123@sendercompany.com"
```

```
            timestamp="2003-11-20T23:59:45-07:00">
    <Header>
        <From>
            <!-- Sender -->
            <Credential domain="AribaNetworkUserId">
                <Identity>sender@sendercompany.com</Identity>
            </Credential>
        </From>
        <To>
            <!-- Recipient -->
            <Credential domain="AribaNetworkUserId">
                <Identity>recipient@recipientcompany.com</Identity>
            </Credential>
        </To>
        <Sender>
            <!-- Sender -->
            <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>
--outer-boundary
Content-Type: Multipart/Related; boundary=inner-boundary
Content-ID: <222@sendercompany.com>
[Other headers]
--inner-boundary
Content-Type: text/xml; charset=UTF-8
Content-ID: <333@sendercompany.com>
[Other headers]
[Forwarded cXML]
--inner-boundary
[Attachment 1 of the forwarded cXML]
--inner-boundary
[Attachment 2 of the forwarded cXML]
--inner-boundary--
--outer-boundary--
```

MIME の詳細情報

MIME 規格の詳細については、次の Web サイトを参照してください。

- www.hunnysoft.com/mime
- www.ietf.org/rfc1341.txt
- www.ietf.org/rfc/rfc2046.txt
- www.ietf.org/rfc/rfc2387.txt

3.1.6 cXML エンベロープ

cXML 要素は cXML ドキュメントのルート要素で、その他のすべての要素はこの中で定義します。cXML 要素は、すべての cXML トランザクションで定義されます。次に、cXML 要素の有効な例を示します。

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

cXML には次の属性があります。

属性	説明
version (非推奨)	この属性は、cXML 1.2.007 以降のバージョンでは推奨されていません。新規の cXML ドキュメントでは使用しないでください。 cXML プロトコルのバージョンを指定します。検証を行う XML パーサーも、参照元の DTD からバージョン属性を確認できます。 このバージョン番号は cXML ドキュメントの SYSTEM 識別子にも表示されるため、この属性は使用しないでください。
xml:lang	この地域情報は、このドキュメントで送信されるすべてのフリーテキストで使用されます。受信者は同一、または類似の地域情報で応答や表示を行ってください。たとえば、Request の中で xml:lang="en-UK" を指定したクライアントは、Response として "en" データを受信します。最も明確で特定化できる地域情報を指定してください。
payloadID (必須)	消失した、または問題が発生したドキュメントを特定するためにロギング目的で使用される、スペースと時間に関する一意の数字。この値を変更して再試行しないでください。 次のような実装を推奨します。 datetime.process id.random number@hostname
timestamp (必須)	メッセージが送信された日付と時刻。ISO 8601 フォーマットで記述されます。この値を変更して再試行しないでください。 フォーマットは、YYYY-MM-DDThh:mm:ss-hh:mm となります (2015-07-14T19:20:30+01:00 など)。
signatureVersion	存在する場合には、ドキュメントが電子署名付きであることを示します。このドキュメントには、Request、Response、または Message 要素の直後に有効な ds:Signature が 1 つ以上定義されています。この属性の有効な値は 1.0 のみです。その他の値は将来の使用に備えて予約されています。

関連情報

cXML 電子署名 [55 ページ]

3.1.6.1 xml:lang で指定する地域情報

`xml:lang` 属性は、ほとんどのフリーテキスト要素 (`Description` や `Comments` など) で指定できます。XML 仕様では、ある要素に地域情報が指定されていない場合、親要素に指定された地域情報がその要素の通常の地域情報となりますが、このようにしてドキュメントツリーの検索を行って通常の値を決定すると効率が低下します。cXML では、地域情報識別子をそれに関連する文字列とともに保持するようにします。最も明確で特定化できる既知の地域情報を、この属性に指定してください。

cXML プロトコルのさまざまな要素で指定できる `xml:lang` 属性は、番号、日付および時刻などのフォーマット済みデータには影響を与えません。次項の timestamp 属性で説明するように、`timestamp` 属性に関しては、それぞれの値はデータタイプに従ってフォーマットされます。マシン処理に配慮していない長い文字列 (および参照される Web ページ) には、近くにある `xml:lang` 属性に一致する地域固有の数値または日付フォーマットが使用される場合があります。

3.1.6.2 日付、時刻およびその他のデータタイプ

`timestamp` 属性と、cXML 内にあるその他すべての日付と時刻は、ISO 8601 で制限されたサブセットでフォーマットされなければなりません。この情報は、『Word Wide Web Consortium (W3C) Note』の「Date and Time Format」に記述されており、www.w3.org/TR/NOTE-datetime-970915.html で入手できます。

timestamp には、少なくとも完全な形式の日付および時、分、秒が含まれている必要があります。秒の小数部の設定は任意です。このプロトコルでは、UTC (協定世界時、グリニッジ標準時としても知られます) を基準とした時間帯で現地時間を表す必要があります。「Z」時間帯識別子は使用できません。

たとえば、`2015-04-14T13:36:00-08:00` は、米国太平洋標準時の西暦 2015 年 4 月 14 日午後 1 時 36 分に相当します。

> **i 注記**
>
> `timestamp` 属性は cXML DTD で必要ですが、値のフォーマットの検証はアプリケーションに依存します。

cXML で使用する日付、時刻、およびその他のデータタイプのフォーマットに関する詳細情報は、次のサイトで参照できます。

- Microsoft 社の XML Data Types Reference: msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/b24aafc2-bf1b-4702-bf1c-b7ae3597eb0c.asp
- W3C (Word Wide Web Consortium) に対する XML データ提案の原書: www.w3c.org/TR/1998/NOTE-XML-data-0105

3.1.6.3 特殊文字

cXML では XML と同様に、すべての文字がキーボードから入力できるわけではありません。たとえば登録商標記号 (R) などがこれに該当します。また、たとえば < や & は、XML では特別な意味を持ちます。これらの文字は、文字エンティティを使用してエンコードする必要があります。

XML では、次の組み込み文字エンティティが定義されています。

エンティティ	文字
<	<
>	>
&	&
"	"
'	'

使用するエンコード以外の文字に関しては、シャープ記号 (#) に続けてその文字の Unicode 番号 (10 進数または 16 進数) を使用します。たとえば、® および ® は、登録商標記号 ((R)) を表します。

次に例を示します。

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

これは、次のようにエンコードします。

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

一重引用符 (') または二重引用符 (") が属性値の中で使用されており、その属性値がこれらの引用符で囲まれている場合、属性値の中で使用されている引用符は、エスケープする必要があります。属性値の中に引用符が含まれる場合、属性は一重引用符のみを使用して囲むことを推奨します。

3.1.6.3.1 ドキュメントにおける特殊文字の処理

1. 属性の区切り文字として一重引用符のみを使用しているテンプレートを使用します。
2. 次のいずれか 1 つを実行して、そのテンプレートに値を追加します。
   - ドキュメントが、cxml-urlencoded 隠しフィールドによって転送された PunchOutOrderMessage である場合、US-ASCII エンコードを使用してテンプレートに値を入力します。このエンコードでは、エンコードできないすべての文字に対して、XML 文字エンティティを使用します。たとえば、登録商標記号 (®) は、US-ASCII ではサポートされていないため、® と入力します。
   - それ以外の場合は、UTF-8 エンコードを使用してドキュメントの値を入力します。HTTP Post により直接送信されたドキュメント、または cXML-base64 隠しフィールドに埋め込まれたドキュメントのすべてで、UTF-8 を使用します。UTF-8 には、US-ASCII コードがすべて含まれています。
3. cXML ドキュメント作成時に、XML で属性値と要素の内容をエスケープします。エスケープする必要がある文字は、&、'、<、および > です。
   PunchOutOrderMessage ドキュメントを転送する場合は、次の手順が必要です。
4. ブラウザが解読するすべての文字について、次のことに注意してください。
   - cxml-urlencoded 隠しフィールドを使用している場合、すべての二重引用符を " に変換します。
   - さらに (cxml-urlencoded フィールドの場合)、HTML で有効なコンテキストで記述されているすべてのアンパサンドを、& に変換してください。安全を期すために、すべてのアンパサンドをエスケープしてもかまいません。たとえば、アンパサンド (&) は & としてエスケープし、アポストロフィ (') は ' としてエスケープします。登録商標記号 (®) は ® としてエスケープします。
   - これ以外の場合で、cxml-base64 隠しフィールドを使用している場合、cXML ドキュメント全体に対して base64 エンコード方式を使用します。

5. 文字列を二重引用符で囲んで、ドキュメントを HTML フォームに埋め込みます。たとえば、値が ®®'''"""&<>>" で、属性の値が ®®'''"""&<>> の Money 要素を送信する場合、XML ドキュメントは次のようになります。

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

これは、次のようにエンコードしてください。

```
<!-- Recommendation for cXML-urlencoding: Uses double quotes to delimit the -->
<!-- field value and single quotes for the contained attributes. -->
<Input type="Hidden" name="cXML-urlencoded" value="<?xml version='1.0'
encoding='UTF-8'?>
<!DOCTYPE Money SYSTEM 'SpecialChars.dtd'>
<Money
alternateAmount='MoneyalternateAmount='&amp;#174;&amp;#xAE;&amp;apos;&#34;&amp;#3
4;
&amp;quot;&amp;amp;&amp;lt;>&amp;gt;'>&amp;#174;&amp;#xAE;'&amp;apos;
&#34;&amp;#34;&amp;quot;&amp;amp;&amp;lt;&amp;gt;&amp;gt;'</Money>">
<!-- Best choice: Base64 encode the value. Don't have to worry about what -->
<!-- the browser interprets. -->
<Input type="Hidden" name="cXML-
base64"value="PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nVVRGLTgnPz4K
PCFET0NUWVBFIE1vbmV5IFNZU1RFTSAnU3BlY2lhbENoYXJzLmR0ZCc+CjxNb
25leSBhbHRlcm5hdGVBbW91bnQ9JyYjMTc0OyYjeEFFOyZhcG9zOyImIzM0OyZxd
W90OyZhbXA7Jmx0Oz4mZ3Q7Jz4KJiMxNzQ7JiN4QUU7JyZhcG9zOyImIzM0OyZx
dW90OyZhbXA7Jmx0Oz4mZ3Q7PC9Nb25leT4K">
```

上記の例は、cXML-urlencoded フィールドをエンコードする別の方法です。ここでは XML が、山カッコなどのいくつかの文字をエスケープしないようにしていますが、これは XML での特別な処理というわけではありません。前記の手順を直接実装すると、次のような HTML フィールドになります。

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

または、次のような XML ドキュメントになります。

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

3.1.7 Header

Header 要素は、アドレス指定と認証情報で構成されます。cXML メッセージのボディに特定の Request または Response がある場合も、Header 要素は同じです。アプリケーションは申請者の ID 情報が必要ですが、ID 情報の内容が正しいかどうかの検証は行いません。

次の例は、Header 要素を示しています。

```
<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。これらは、メッセージの論理的な発信元と宛先です。`Sender` 要素に定義する情報は、HTTP 接続を開始して cXML ドキュメントを送信する組織または人です。

`Sender` 要素には `Credential` 要素が含まれており、この要素を使用して、受信者は送信者を認証します。この認証方法は、公開鍵によるデジタル認証のためのインフラストラクチャを必要とせずに、強力な認証を可能にします。送信者は、受信者側が発行したユーザー名とパスワードを使用することで、`Requests` を実行できます。

ドキュメントが最初に送信される時は `Sender` と `From` は同一の組織を示していますが、cXML ドキュメントがネットワークハブを経由して転送されると、`Sender` 要素は変更され、送信した直前の組織を示すようになります。

3.1.7.1 From

この要素は、cXML Request の発信元を識別します。

3.1.7.2 実行する処理

この要素は、cXML Request の宛先を識別します。

3.1.7.3 送信者

この要素によって、受信者側は HTTP 接続を開始した相手を識別して認証します。受信者側は作業の要求者を認証する必要があるため、この要素内では、`From` または `To` 要素の場合よりも強力な認証機能を持つ `Credential` 要素を使用します。

3.1.7.4 UserAgent

cXML で対話を行う `UserAgent` を表す、テキスト文字列です。この文字列は、製品ごとに固有にする必要があり、可能であればバージョンごとにも固有にしてください。これは、HTTP の `UserAgent` に類似しています。

3.1.7.5 Credential

この要素には、識別値と認証値を定義します。

`Credential` には次の属性があります。

属性	説明
`domain` (必須)	認証情報のタイプを指定します。この属性を使用して、複数の認証ドメインに対する複数のタイプの認証情報をドキュメントに含めることができます。 たとえば、Ariba Network 上で送信されたメッセージについて、domain に `AribaNetworkUserId` が指定されている場合は電子メールアドレス、`DUNS` の場合は DUNS ナンバー、`NetworkId` の場合は割り当て済みの ID を表します。
`type`	マーケットプレイスからの、またはマーケットプレイスへの要求は、マーケットプレイスとメンバー企業の両方を `From` または `To Credential` 要素で識別します。この場合、マーケットプレイスに対する認証情報には `type` 属性を使用し、その属性は「`marketplace`」という値に設定します。

`Credential` には、`Identity` 要素を含め、任意で `SharedSecret` 要素または `CredentialMac` 要素を含めます。`Identity` 要素は `Credential` の対象である組織について記述し、任意設定の認証要素はその組織の ID 情報を記述します。

SharedSecret

`SharedSecret` 要素は、申請者が認識するパスワードが `Sender` に含まれる場合に使用します。

> **i 注記**
>
> One-Way 転送によって送信されるドキュメントには、認証要素を使用しないでください。One-Way 転送はユーザーのブラウザを介してルーティングされるため、`Credential` 要素を含むドキュメントのソースをユーザーに参照される可能性があります。

CredentialMac

`CredentialMac` 要素は、メッセージ認証コード (MAC) による認証で使用されます。この認証方法は、信頼のおけるサードパーティにより共有シークレットを使用して認証されたことを、送信者が受信者に証明する必要がある状況で使用され

ます。たとえば、ダイレクトパンチアウト要求は、サプライヤによる認証が可能な MAC (ネットワークハブによって生成されたもの) が含まれるため、バイヤーからサプライヤへ、ネットワークハブを介さずに直接送信できます。

信頼のおけるサードパーティが MAC を計算し、プロファイルトランザクションを介して送信者へ転送します。送信者は MAC の値しか見ることはできません (安全で逆変換できません)。MAC は、`ProfileResponse` オブジェクトを使用し、信頼のおけるサードパーティから送信者に送信されます。

受信者は、信頼のおけるサードパーティと同じ入力データを使用して MAC を計算し、cXML ドキュメントで受信した MAC と比較します。この 2 つの値が一致するとドキュメントは認証されたことになります。

MAC 値の計算方法については、メッセージ認証コード (MAC) [47 ページ] を参照してください。

`CredentialMac` には、次の属性があります。

属性	説明
`type` (必須)	認証されるデータとその認証用のフォーマット方法を識別します。サポートされる値は `"FromSenderCredentials"` のみです。
`algorithm` (必須)	データで使用された MAC アルゴリズムを識別します。サポートされる値は `"HMAC-SHA1-96"` のみです。
`creationDate` (必須)	MAC が生成された日時を指定します。
`expirationDate` (必須)	MAC の有効期限を示す日時を指定します。受信者は expirationDate 後に受け取った MAC を却下する必要があります。受信者は有効期限前の MAC を、任意で却下することもできます。たとえば、受信者は 1 時間以内に有効期限が切れる MAC を却下する場合があります。

次の例は、`CredentialMac` 要素が使用されている `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>
```

複数の認証情報

`From` 要素、`To` 要素、および `Sender` 要素には、それぞれに複数の `Credential` 要素を任意で設定できます。複数の認証情報を提供する目的は、さまざまなドメインを使用している 1 つの組織を識別することです。たとえば、DUNS ナンバーおよび NetworkId ナンバーの両方を含めることで、組織が識別される場合があります。

受信者は、受け取ったドメインのすべての認証情報を検証する必要があります。使用されたドメインの認証情報が、認識している組織と一致しない場合、そのドキュメントを却下してください。同じ `From`、`To`、または `Sender` セクションの 2 つの認証情報が別々のエンティティを参照している場合にも、ドキュメントは却下されます。

異なる値が使用されているものの、同じドメインが使用されている `To`、`From`、または `Sender` セクションに複数の認証情報が存在する場合には、ドキュメントは却下されます。

3.1.7.6 Correspondent

From および To 要素には、任意設定の Correspondent 要素をそれぞれ含めることができます。Correspondent 要素は、発信側または受信側の組織が、もう一方の組織または接続ハブにとって未知である場合に使用されます。送信者、受信者、または接続ハブは、Correspondent 要素内の情報を使用して未知の組織を識別できます。

Correspondent には次の属性があります。

属性	説明
preferredLanguage	組織の優先言語（わかっている場合）

Correspondent には以下の要素が含まれます。

要素	説明
Contact （必須）	オーダーをフォローアップするための連絡先情報が含まれます。「連絡先 [46 ページ]」を参照してください。
Routing	対応するルーティング宛先を定義します。「Routing [32 ページ]」を参照してください。
Extrinsic	この組織に関連する追加情報が含まれます。

3.1.7.6.1 Routing

外部ビジネスパートナーの対応するルーティング宛先を定義します。Routing には次の属性があります。

属性	説明
destination （必須）	ルーティング宛先の名前。使用可能な値は次のとおりです。 • peppol • fieldglass

以下の例は、外部ビジネスパートナーの Routing 要素を示します。

```
<Header>
  <From>
    <Credential domain="BusinessPartnerId">
      <Identity>
        <IdReference domain="iso6523" identifier="9925:BE12345678"/>
      </Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="BusinessPartnerId">
      <Identity>
        <IdReference domain="iso6523" identifier="9925:BE3456789" />
      </Identity>
    </Credential>
    <Correspondent preferredLanguage="de">
      <Contact role="correspondent">
        <Name xml:lang="en-US">SupplierTradingName Ltd.</Name>
        <PostalAddress>
          <Street>Street</Street>
          <City>City</City>
```

```
          <State>State</State>
          <PostalCode>04726010</PostalCode>
          <Country isoCountryCode="BE" />
        </PostalAddress>
        <Phone name="work">
          <TelephoneNumber>
            <CountryCode isoCountryCode="BE" />
            <AreaOrCityCode />
            <Number>1151869655</Number>
          </TelephoneNumber>
        </Phone>
      </Contact>
      <Routing destination="peppol" />
    </Correspondent>
  </To>
  <Sender>
    <Credential domain="NetworkID">
      <Identity>AN01000000001</Identity>
    </Credential>
    <UserAgent>Ariba Network</UserAgent>
  </Sender>
</Header>
```

3.1.8 依頼

クライアントは、Requestドキュメントを送信して特定の操作を要求します。cXMLドキュメントの読み取り処理を分割する必要がないよう、各 cXML エンベロープ要素で使用できる `Request` 要素は 1 つだけです。このため、サーバーの実装は単純になります。`Request` 要素には、事実上すべてのタイプの XML データを含めることができます。

一般的な `Request` 要素として次のものがあります。

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

`Request` には次の属性があります。

属性	説明
`deploymentMode`	Request がテストモードであるか本稼動モードであるかを示します。値は、「production」(通常) または「test」です。
`Id`	この属性は要素およびそのすべての子要素を電子署名の対象として呼び出すために使用します。

関連情報

cXML 電子署名 [55 ページ]

3.1.9 Response

サーバーはクライアントに Response を送信して、操作結果を通知します。一部の Request の結果にはデータがない場合があるため、`Response` 要素は、任意で `Status` 要素のみで構成することもできます。さらに、`Response` 要素にはどのようなアプリケーションレベルのデータも含めることができます。たとえば PunchOut セッションでは、そのアプリケーションレベルのデータが `PunchOutSetupResponse` 要素に含まれています。

一般的な `Response` 要素として次のものがあります。

- `ProfileResponse`
- `PunchOutSetupResponse`
- `GetPendingResponse`

`Response` には次の属性があります。

属性	説明
`Id`	この属性は要素およびそのすべての子要素を電子署名の対象として呼び出すために使用します。

関連情報

cXML 電子署名 [55 ページ]

3.1.9.1 Status

この要素は要求された操作が成功したか、一時エラーか、または永久エラーかを通知します。

`Status` には次の属性があります。

属性	説明
`code` (必須)	要求の状況コード。たとえば、200 は要求が成功したことを示します。下記のコード表を参照してください。
`text` (必須)	状況のテキスト。このテキストは、英語で記述された標準的なエラー表記で、ユーザーが判読できるログです。
`xml:lang`	`Status` 要素で使用されるデータの言語。cXML 1.0 との互換性を目的としたオプションです。cXML の将来のバージョンで必要となる場合があります。

Status 要素の属性は、要求の結果を表します。以下に例を示します。

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

Status 要素の内容には、申請者が必要なあらゆるデータを記述することができますが、エラーに関する情報を記述するようにしてください。200/OK 状況コードには、データが存在しない場合があります。ただし、500/Internal Server Error 状況コードまたはその他の類似するコードには、実際の XML 解析エラーまたはアプリケーションエラーを含めることを強く推奨します。このエラーは、一方向のデバッグや相互運用性のテストをする際に役立ちます。以下に例を示します。

```
<Status code="406" text="Not Acceptable">cXML did not validate. Big Problem!</
Status>
```

次の表は、cXML 状況コードの範囲を示します。

範囲	意味
2xx	成功
4xx	永久エラー。クライアントは、再試行しないでください。このエラーの場合、申請は承認されていません。
5xx	一時エラー。一般的には転送エラーです。クライアントは、再試行してください。推奨する再試行回数は、1 時間おきに 10 回です。少なくとも 6 回の再試行を推奨します。緊急なオーダーなど、重要度の高い申請に関しては、再試行頻度を上げてもかまいません。

cXML 状況コードが 200 番台 (cXML 200/OK など) でない限り、サーバーには追加の Response 要素 (PunchOutSetupResponse 要素など) を含めないでください。

ほとんどの場合、cXML は HTTP の上の階層にあるため、多くのエラー (HTTP 404/Not Found など) はトランスポートによって発生しています。すべてのトランスポートエラーは、cXML 500 番台の状況コードを受け取った時と同様に一時エラーとして取り扱い、クライアントは再試行する必要があります。HTTP 404/Not found や HTTP 500/Internal Server Error 状況コードなどの、有効な cXML コンテンツを含まない HTTP Response は、すべて転送エラーであるとみなされます。転送に関するその他の一般的な問題には、タイムアウト、TCP エラー (「connection refused」など)、および DNS エラー (「host unknown」など) があります。Request ドキュメントの構文解析における検証エラーの場合、一般的には 400 番台、多くは 406/Not Acceptable の cXML 永久エラーが発生します。

次の表に、発生し得る cXML 状況コードを示します。

状況	テキスト	意味
200	OK	サーバーは Request を実行しました。または、最終受信者へ結果を送信しました。返された Response には、アプリケーションの注意またはエラーが含まれる可能性があります。cXML Request 自身はエラーも注意も生成しませんでしたが、この状況はその後のアプリケーションで発生し得るいかなるエラーや注意も、反映されません。後の処理でエラーが発生しない限り、この状況の更新情報は受け取りません。
201	Accepted	中間ハブによって転送の Request が受け入れられました。または、その最終の宛先によって受信されましたが、解析はされていません。送信するメカニズムが利用可能であれば、Request に関する状況の更新情報を受け取ります。 クライアントはこの後 StatusUpdate トランザクションを受信します。

状況	テキスト	意味
204	No Content	すべての `Request` 情報は有効で、認識されました。サーバーには、要求されたタイプの `Response` データがありません。 `PunchOutOrderMessage` において、この状況はパンチアウトセッションがショッピングカート (または、クライアントの購入申請) に対する変更なしに終了したことを示します。
211	OK	バイヤーはこの状況コードを使用して、サプライヤが把握する必要があるイベント (休日のスケジュール、生産設備の閉鎖、特定の活動の完了 (たとえば、計画実行完了) など) を通知するために、サプライヤにブロードキャストメッセージを送信することができます。
280		中間ハブにより Request が転送されました。少なくともう 1 つの状況更新情報を受け取ります。この状況は、Request が別の中間転送者、または、状況 201 で最終受信者に送信されたか、信頼できる非 cXML のトランスポートを経由して転送されたことを意味する場合があります。
281		非確実なトランスポート (電子メールなど) を使用して、中間ハブにより Request が転送されました。状況更新情報を受け取る可能性があります。ただし、状況更新情報を受け取らない場合、必ずしも問題があるというわけではありません。
400	Bad Request	構文解析では問題がありませんが、サーバーが Request を受け入れられませんでした。
401	Unauthorized	`Request` (`Sender` 要素) の中で指定されている Credential が、サーバーに認識されませんでした。
402	Payment Required	`Request` には、完全な `Payment` 要素が含まれていなければなりません。
403	Forbidden	ユーザーが、`Request` を実行するための十分な権限を持っていません。
406	Not Acceptable	`Request` はサーバーに受け入れられませんでした。構文解析の失敗によるものと推測されます。
409	Conflict	サーバーの現在の状況またはその内部データにより、(更新) 操作要求が中断されました。同様の `Request` は、別の操作の実行後以外では、今後も正常に実行される可能性はありません。
412	Precondition Failed	`Request` の前提条件 (`PunchOutSetupRequest edit` に対する適切なパンチアウトセッションなど) が満たされていませんでした。この状況は、一般的には、サーバーからの以前の転送の一部をクライアントが無視したことを意味します (`PunchOutOrderMessageHeader` の `operationAllowed` 属性など)。
417	Expectation Failed	`Request` のリソース条件が満たされなかったことを示します。1 つの例として、サーバーが未知のサプライヤに関する情報を要求する `SupplierDataRequest` があります。この状況は、クライアントまたはサーバーで消失した情報があることを示します。
450	Not Implemented	サーバーは指定の `Request` を実装していません。たとえば、`PunchOutSetupRequest`、もしくは要求された操作がサポートされていない可能性があります。一般的に、この状況はクライアントがサーバーのプロファイルを無視したことを示します。
475	Signature Required	ドキュメントに電子署名がないため、受信者はドキュメントを受け付けようとしません。
476	Signature Verification Failed	転送中にドキュメントが変更、または署名で使用されたアルゴリズムの 1 つまたは複数を受信者がサポートしないなどの原因により、受信者は署名を検証することができません。

状況	テキスト	意味
477	Signature Unacceptable	署名は技術的に有効ですが、その他の理由により受信者に承認されません。署名規定または証明書規定が承認されないか、使用された証明書の種類が承認されないか、またはその他の問題が存在する可能性があります。
500	Internal Server Error	サーバーが `Request` を完了できませんでした。
550	Unable to reach cXML server	次の処理のための接続が必要なトランザクションは、次の cXML サーバーに到達できず完了できません。サプライヤサイトに到達できないとき、中間ハブがこのコードを返すことがあります。次への接続が完了した場合、中間ハブはエラーをクライアントに直接返さなければなりません。
551	Unable to forward request	サプライヤの設定ミスにより、Request を転送できません。たとえば、中間ハブがサプライヤに対し自身の認証に失敗しました。クライアントはこのエラーを修正できませんが、クライアントが再試行する前にこのエラーが解決される場合があります。
560	Temporary server error	たとえば、サーバーは保守などで停止することがあります。クライアントは後で再試行してください。

下の表は、CatalogUploadRequest に対する可能な状況コードを一覧にしたものです。

状況	テキスト	意味
200	Success	CatalogUploadRequest は正常に処理されました。
201	Accepted	CatalogUploadRequest を処理中です。
461	Bad Commodity Code	カタログに割り当てられている商品分類コードが無効です。
462	Notification Error	通知方法 (電子メールまたは URL) が指定されていません。
463	Bad Catalog Format	ZIP ファイルが無効です。
464	Bad Catalog	カタログが添付されていないか、複数のカタログが添付されています。
465	Duplicate Catalog Name	同じカタログ名が存在します。
466	No Catalog to Update	更新するカタログが存在しません。
467	Publish Not Allowed	以前に公開されていないカタログを公開しようとしました。
468	Catalog Too Large	アップロードするファイルのサイズが 4MB の制限を超えています。カタログをアップロードする前に、ZIP 形式で圧縮してください。
469	Bad Catalog Extension	カタログのファイル名の拡張子は、.cif、.xml、または.zip でなければなりません。
470	Catalog Has Errors	メッセージが、カタログの状況を示しています。(HasErrors)
499	Document Size Error	cXML ドキュメントのサイズが大きすぎます。
561	Too Many Catalogs	1 時間当たり一定数以上のカタログをアップロードすることはできません。
562	Publish Disabled	定期保守作業に伴い、カタログの公開を一時停止しています。指定した日時までに復旧する見通しです。
563	Catalog Validating	以前のバージョンのカタログの検証が終了する前に、カタログを更新しようとしました。

認識されないコードを受信したとき、cXML クライアントはコードのクラスに従ってそれらのコードを処理する必要があります。したがって、古いクライアントは、すべての新しい 2xx コードを 200 (成功)、4xx コードを 400 (永久エラー)、および

5xx コードを 500 (一時エラー) として処理する必要があります。この動作は、今後、相互運用性を損なわずに、cXML プロトコルとサーバー固有のコードが拡張されることを考慮しています。

3.1.10 One-Way (非同期式) モデル

Request/Response トランザクションと異なり、One-Way メッセージは HTTP 転送に限定されません。One-Way メッセージは、HTTP チャネル (同期式の Request/Response タイプの操作) で対応できないような場合に使用します。次の図は、Request/Response トランザクションを使用せず A および B がメッセージを通信する方法を示しています。

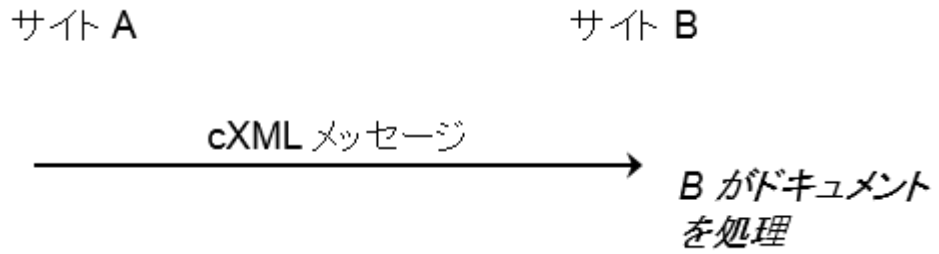


図 6: One-Way メッセージ (非同期式)

この場合、以下のシナリオが考えられます。

1. サイト A は、サイト B が認識する転送方法で cXML ドキュメントをフォーマットしてエンコードします。
2. サイト A は、その転送方法を使用してドキュメントを送信します。サイト A は、サイト B からの Response を待つことはしません (待ち状態になることはできません)。
3. サイト B は、cXML ドキュメントを受信した後、それを転送ストリームからデコードします。
4. サイト B はドキュメントを処理します。

One-Way モデルでは、サイト A とサイト B の間に明示的な Request/Response サイクルは存在しません。たとえば、One-Way メッセージによる通信中に別の相手からメッセージが到着して、そちらの対話が始まる可能性もあります。

One-Way トランザクションを完全に明記するには、メッセージの転送方法も記述します。たとえば、One-Way アプローチを使用する cXML トランザクションでは、転送方法とエンコード方式を指定します。One-Way トランザクションを使用する一般的な例は、`PunchOutOrderMessage` です。

One-Way メッセージの構造は、Request/Response モデルの構造と類似しています。

```
<cXML>
    <Header>
        Header information here…
    </Header>
    <Message>
        Message information here…
    </Message>
</cXML>
```

`Header` 要素は、Request/Response ドキュメントと同様に処理されます。`cXML` 要素も、cXML エンベロープ [25 ページ] に説明した内容と同じです。One-Way メッセージと Request/Response メッセージとの違いを見分ける最も簡単な方法は、`Request` または `Response` 要素の代わりに使用する `Message` 要素の有無を確認することです。次のセクションでは、`Message` 要素についてさらに詳しく説明します。

One-Way メッセージの Header 要素の送信者認証情報に、共有シークレット情報を含めないでください。認証は、BuyerCookie を使用して行われます。これは、Request/Response の Header とは異なります。

3.1.11 Message

この要素には、cXML メッセージのすべてのボディの情報を定義します。この要素には、任意設定の Status 要素を含めることができます (Response 要素の Status 要素と同じです)。Status 要素は、Request メッセージに対する Response メッセージ中で使用します。

Message には次の属性があります。

属性	説明
deploymentMode	Request がテストモードであるか本稼動モードであるかを示します。値は、「production」(通常) または「test」です。
inReplyTo	この Message が応答する Message を指定します。inReplyTo 属性の値は、以前に受信した Message の payloadID です。この属性は、多数のメッセージによる双方向の対話を行う場合に使用します。
Id	この属性は要素およびそのすべての子要素を電子署名の対象として呼び出すために使用します。

inReplyTo 属性は、以前の Request または Response ドキュメントの payloadID を参照することもできます。Request/Response トランザクションが、複数の One-Way メッセージを介して「対話」を開始するとき、最初のメッセージには、逆方向へ最後に送信された適切な Request または Response の payloadID を含めることができます。たとえば、PunchOutOrderMessage を含む Message には、パンチアウトセッションを開始した PunchOutSetupRequest の payloadID を持つ inReplyTo 属性が指定されている場合があります。パンチアウトドキュメントに含まれる BuyerCookie には、この inReplyTo 属性と同様の機能があります。

関連情報

cXML 電子署名 [55 ページ]

3.1.12 転送オプション

One-Way メッセージには、HTTP と URL フォームエンコードの 2 つの一般的に使用される転送があります。現在明確に定義されている転送は、これら 2 つだけです。将来、さらに多くの転送がサポートされる可能性があります。

HTTP

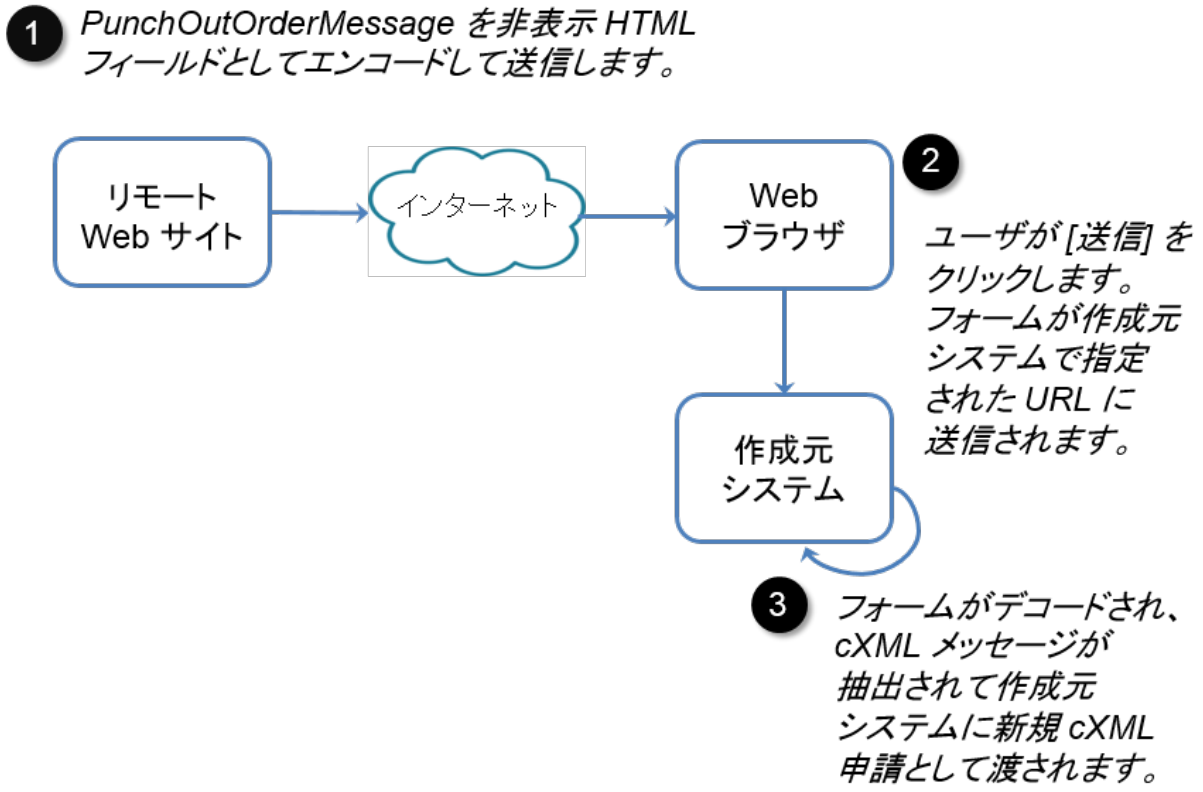
調達アプリケーションは、One-Way HTTP 通信を使用して情報を引き出します。One-Way HTTP 通信を使用するトランザクションの 1 つに、GetPendingRequest があります。

セキュリティのため転送データが暗号化されるため、HTTPS が推奨されます。

URL フォームエンコード

URL フォームエンコードを使用することにより、リモート Web サイトと調達アプリケーションの統合が可能になります。さらに、URL フォームエンコードでは、バイヤーシステムの受信サーバーを通す必要がなく、バイヤーシステムにインターネットを経由して直接アクセスできます。`PunchOutOrderMessage` トランザクションがどのように動作するかの説明は、この転送の仕組みを理解するのに役立ちます。

リモート Web サイトは、cXML `PunchOutOrderMessage` ドキュメントを調達アプリケーションに直接送信するのではなく、このドキュメントを HTML Form の隠しフィールドとしてエンコードし、`PunchOutSetupRequest` の `BrowserFormPost` 要素で指定した URL の示す場所にその情報を書き込みます。ユーザーが買物の終了後に Web サイトの [チェックアウト] ボタンをクリックすると、Web サイトはそのデータを HTML Form で調達アプリケーションに送信します。下の図に、その処理を示します。



パッキングとアンパッキングについて以下で説明します。

Form パッキング

リモート Web サイトで、各 `PunchOutOrderMessage` ドキュメントが `cXML-urlencoded` または `cXML-base64` という Form の隠しフィールドに割り当てられます。また、HTML Form 要素で METHOD に POST を、ACTION に `PunchOutSetupRequest` の `BrowserFormPost` 要素の中で示された URL を割り当てます。例:

```
<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>
```

ページ上に追加された HTML タグには、ショッピングカートの内容を詳しく説明するために、上記のフラグメントが含まれることがあります。

> **i 注記**
>
> Web サーバーが `cXML-urlencoded` フィールドを送信したときには、URL はまだエンコードされていません。このエンコードは、フォームが Web ブラウザによって送信されるとき (上記の例で、ユーザーが [チェックアウト] をクリックしたとき) にのみ必要です。Web ブラウザ自体はこの必要条件を満たしています。Web サーバーが HTML エンコードの処理をする必要があるのは、フィールド値、エスケープのための引用符、およびその他の特殊文字のみであるため、フォームはユーザーにとって適切な形で表示されます。

名前 `cXML-urlencoded and cXML-base64` では、大文字/小文字が区別されます。

cXML-urlencoded

`cXML-urlencoded` フィールドは、Web サーバーやサプライヤではなく、Web ブラウザによって (HTTP 仕様に従って) エンコードされた URL です。これは、たとえば前の例で、ユーザーが [チェックアウト] をクリックした場合のように、Web ブラウザによってフォームが送信された場合にのみエンコードが必要となるためです。ただし、フォームを適切な形で表示するために、Web サーバーはフィールド値、エスケープのための引用符、およびその他の特殊文字を HTML エンコード処理する必要があります。

> **i 注記**
>
> サプライヤは、cXML-urlencoded フィールドを URL エンコード処理しないでください。このフィールドは、Web ブラウザによって自動的に URL エンコードされます。

受信パーサーは、`cXML-urlencoded` データに関して、メディアタイプ `text/xml` の通常の値を超えた `charset` パラメータを解釈することができません。送信されたデータの文字エンコード情報は、HTTP POST では保持されません。受信 Web サーバーは、隠しフィールドを含む HTML ページのエンコードを解釈できません。したがって、この方法で転送される cXML ドキュメントでは、`us-ascii` 文字エンコードを使用する必要があります。XML ソースドキュメントのすべての文字 (「%XX」として「URI エンコード処理」されたものを含みます) が、「US-ASCII」文字になっている必要があります。その他の Unicode 記号は、そのソースドキュメント内の文字エンティティを使用してエンコードします。

cXML-Base64

cXML-base64 の隠しフィールドは、多言語のドキュメントをサポートしています。「US-ASCII」以外の記号を含む cXML ドキュメントは、cXML-urlencoded 隠しフィールドではなく、このフィールドを使う必要があります。この方法は意味的にはほぼ同じですが、ブラウザに対する HTML エンコードや受信 Web サーバーに対する URL エンコードを使用せず、転送を通じてドキュメント全体を Base64 エンコードします。Base64 エンコードは、RFC 2045 の『Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies』で説明されています。

ブラウザを介したリモート Web サイトからクライアントの受信 Web サーバーまでの Base64 エンコードでは、cXML ドキュメントの元の文字のエンコードを保持します。charset パラメータは通知される情報と一緒には届きませんが、(転送エンコードが削除された後の) デコード済みドキュメントは、メディアタイプ application/xml として処理できます。この処理によって、受信パーサーは、XML 宣言で指定されたすべての encoding 属性を受け付けることができます。このフィールド (すべての application/xml ドキュメント) に対する通常の文字エンコードは、UTF-8 です。

これらの隠しフィールド (cXML-urlencoded または cXML-base64) のいずれか一方は、調達アプリケーションに送信されるデータに含まれていなければなりません。最初に検索されるのは cXML-base64 ですが、両方のフィールドを送信する必要はありません。

Form のアンパッキングと処理

事前に適切な URL を通知している調達アプリケーションは、上述の Form データを含む HTML Form POST を受信します。Form POST プロセッサは、最初に cXML-base64 変数を検索し、その値を抽出して、内容を Base64 でデコードします。こ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 変数を検索し、URL エンコードによる cXML メッセージを抽出し、それを URL デコードします。デコードされたフィールドの内容は、それが通常の HTTP Request/Response サイクルによって受信されたデータの場合と同様に処理されます。

デコード後に、ドキュメントの暗黙のメディアタイプが変化して、異なる文字エンコードが指定されている可能性があります。

- cXML-urlencoded 変数の場合は、メディアタイプ text/xml を意味し、charset 属性はありません。そのため、この文字エンコードは US-ASCII に限定されます。ブラウザがエンコードを変更している可能性があるため、受信パーサーは、cXML ドキュメントの XML 宣言の encoding 属性を無視する必要があります。
- cXML-base64 変数の場合は、メディアタイプ application/xml を意味し、どのような文字エンコードも含まれる可能性があります (XML 宣言に encoding 属性がある場合は、それによって示されています)。application/xml ドキュメントの通常の文字エンコードは UTF-8, です。

このトランザクションと標準の Request/Response トランザクションとの主な相違点は、Response 送信用の HTTP 接続がないため Response を生成しないことです。

3.1.13 サービス状況応答

このトランザクションは、特定のサービスが現在利用可能かどうかの問い合わせを解決します。HTTP GET がサービスサイトに送られると、サービスは動的に作成された有効な cXML Response ドキュメントで応答します。サービスの HTTP URL は、cXML Request ドキュメントを受信できればどこでも可能です。

たとえば、`https://service.ariba.com/service/transaction/cxml.asp` に送信された HTTP `GET` は、次のような Response を受け取ります。

```
<?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--488292003110001493 6@206.251.25.169">
    <Response>
        <Status code="200" text="OK">Ping Response Message</Status>
    </Response>
</cXML>
```

> **i 注記**
>
> このトランスポート (HTTP) とプロトコル (cXML) レベルの組み合せは、上記の場合に限定してください。

3.2 基本要素

以下のエンティティと要素は、cXML 仕様全体に使用できます。ここで一覧表示している定義のほとんどは、高次元のビジネスドキュメントを記述するための基本的な用語です。ここでは、共通の type エンティティと、低レベルのオブジェクトを表す共通の要素について定義します。

3.2.1 Type エンティティ

これらの定義の大部分は、W3C (World Wide Web Consortium) に Note として提出された XML-Data で定義されています。ここで定義されるいくつかの高レベル type エンティティは、XML-Data では定義されていません。

isoLangCode

ISO 639 規格による ISO 言語コードです。

isoStateCode

都道府県/州を識別する ISO 3166-2:2013 国細分コードです。ISO 3166-1 に示されている国コードとともに使用されます。

isoCountryCode

ISO 3166 規格による ISO 国コードです。

xmlLangCode

XML 1.0 仕様書によって定義される言語コード (www.w3.org/TR/1998/REC-xml-19980210.html を参照) です。一般的には、この言語コードには、ISO 639 言語コードと、ハイフンによって区切られる ISO 3166 国コード (任意設定) が含まれます。完全な XML 勧告とは異なる IANA や私的な言語コードを cXML で使用すべきではありません。IANA と私的なサブコードを使用する場合は、それらの前に正しい ISO 3166 国コードを指定してください。

推奨される cXML 言語コードフォーマットは、xx[-YY[-zzz]*]? です。ここで、xx は ISO 639 言語コード、YY は ISO 3166 国コード、そして zzz は IANA または該当する言語についての私的なサブコードです。国コードは常に指定するようにしてください。言語コードは小文字、国コードは大文字で表記する規約がありますが、コードが正しく一致するための必要条件ではありません。

UnitOfMeasure

`UnitOfMeasure` には、製品を梱包して出荷する方法を記述します。数量単位は、数量の共通コードである UN/CEFACT 単位に準拠していなければなりません。参照資料 www.unece.org/cefact/codesfortrade/codes_index.html を参照してください。

URL

HTTP/1.1 規格によって定義された URL (Uniform Resource Locator) です。

関連情報

cXML エンベロープ [25 ページ]

3.2.2 ベース要素

この項で説明する要素は、`Name` や `Extrinsic` などの一般的な要素から `Money` などの特定の要素まで、仕様全体に使用されます。

Money

Money 要素には、currency、alternateAmount、および alternateCurrency． の 3 つの属性があります。currency と alternateCurrecy 属性は、必ず ISO 4217 で規定された 3 文字の通貨コードにしてください。Money 要素と aternateAmount (代替通貨) 属性の値は、数値でなければなりません。例:

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

任意設定の alternateCurrency と alternateAmount 属性は同時に使用し、代替通貨の金額を指定します。これらは、ユーロのような二重通貨をサポートするためにも使用できます。例:

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

> **i 注記**
>
> 任意で 3 桁ごとに区切りのカンマを使用することができます。小数点の代わりにカンマを使用しないでください。

State

都道府県/州または国細分識別子が含まれます。PostalAddress 要素の中で定義します。これには、任意設定の isoStateCode [43 ページ] 属性があります。

```
<State isoStateCode="US-CA">CA</State>
```

国

所在地の項で指定する国名です。PostalAddress 要素の中で定義します。これには、任意設定の isoCountryCode [44 ページ] 属性があります。

```
<Country isoCountryCode="US">United States</Country>
```

CountryCode

国コードに対応した、ITU 電話コードです。エスケープコードの後に電話キーパッドから入力して、該当の国に接続します。TelephoneNumber 要素で使用されます。

```
<TelephoneNumber>
  <CountryCode isoCountryCode="US">1</CountryCode>
  <AreaOrCityCode>800</AreaOrCityCode>
  <Number>5551212</Number>
</TelephoneNumber>
```

連絡先

Contact 要素には、現在のトランザクションにとって重要などのような連絡先に関する情報も定義できます。例:

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

4 その他の認証方法

cXML では、共有シークレット以外の認証方法でも、cXML ドキュメントの送信者を検証できます。

4.1 メッセージ認証コード (MAC)

メッセージ認証コード (MAC) を使用すると、(ネットワークハブなどの) 信頼できるサードパーティを経由せずに、クライアントからサーバーに直接送信されたドキュメントを認証できます。このようなドキュメントは、信頼できるサードパーティと受信者によってのみ解読可能な認証コードを持つ認証情報を含んでおり、送信者は解読できません。

MAC を含む `Credential` 要素 (element) のフォーマットについては、Credential [30 ページ] で説明しています。

4.1.1 MAC の概要

MAC の主な目的は、受信者の共有シークレットを、送信者に対して明らかにせずに伝達することです。MAC では、ハッシュを使用してエンコードすることで、共有シークレットの安全性を確保しています。

MAC は、共有シークレットと同様に安全です。送信者は、共有シークレットと同様に注意深く MAC を取り扱う必要があります。たとえ断片的であっても、情報の漏洩について妥協することは、取引先を危険にさらす恐れがあります。

MAC 認証を使用するには、信頼できるサードパーティと受信者の両方で MAC を計算できるようにする必要があります。

4.1.2 計算アルゴリズム

MAC は、信頼できるサードパーティと受信者の両者にとって既知のデータを組み合わせるアルゴリズムによって生成されます。

cXML では、IETF RFC 2104 の「HMAC: Keyed-Hashing for Message Authentication」に記述されている HMAC-SHA1 アルゴリズムを使用するように指定されています。

HMAC-SHA1 アルゴリズムは、cXML に必要な安全性を提供します。このアルゴリズムは、基になるハッシュアルゴリズムと同様に安全であることが正式に証明されています。

IETF RFC 2104 の詳細については、www.ietf.org/rfc/rfc2104.txt を参照してください。

4.1.3 作成日と有効期限

作成日と有効期限を指定することで、MAC の安全性が向上します。

MAC が盗まれた場合、送信者の共有シークレットを変更しても効果はありません。MAC を無効にするため、送信者が受信者に帯域外で連絡することを期待することは現実的ではありません。これは、両者の間に取引関係が確立されていない場合もあるためです。この問題に対処するため、MAC には、作成日 (`creationDate`) と有効期限 (`expirationDate`) が組み込まれています。有効期限を指定し、MAC が期限切れになることで、MAC の盗難による被害を最小限に抑えます。有効期限が短いほど、安全性は向上します。受信者は、expirationDate 後に受け取った MAC を却下する必要があります。

受信者は、作成日からの経過日数に基づいて、期限切れになっていない MAC を拒否することもできます。たとえば、数年前に作成され、受信の翌日に期限切れになる MAC を受信した場合、受信者はその MAC の受け入れを望まないかもしれません。この判断は、受信システムの実装者に委ねられています。

受信者は、作成日が過去の日付で有効期限が未来の日付であることを確認し、どちらか一方でも条件が満たされていない場合は、その MAC を拒否する必要があります。しかし、作成日が古すぎる場合に、拒否するかどうかは、受信者側の任意です。

受信者は、MAC が有効なことを確認するのみでなく、MAC で認証されたデータが受け入れ可能かどうかも確認する必要があります。特に、From と Sender の認証情報で認識されるエンティティからのメッセージを受け入れることが妥当かどうかを検証する作業が必要です。

4.1.4 計算プロセス

この項では、`type="FromSenderCredentials"` の MAC を計算する方法について説明します。この方法で MAC に入力するデータは、信頼できるサードパーティと受信者のみに既知です。

信頼できるサードパーティは、この計算方法で `ProfileResponse Option` 要素を作成し、受信側のサーバーは同じ計算方法で `CredentialMac` 要素を検証します。

4.1.4.1 ハッシュによる入力の生成

MAC 関数の入力には、データ入力と秘密鍵入力の 2 つが必要です。

- 入力するデータは、次の各値を、正規化した上で値の末尾に 1 つのヌルバイト (0x00) を付加し、下に示す順番どおりに並べて UTF-8 でエンコードしたバイト表現です。

```
From/Credential@domain
From/Credential/Identity
Sender/Credential@domain
Sender/Credential/Identity
Sender/Credential/CredentialMac@creationDate
Sender/Credential/CredentialMac@expirationDate
```

- 入力する秘密鍵は、受信者とサードパーティとの間で使用される cXML 共有シークレットです。

4.1.4.2 入力値の正規化

ハッシュ入力値を計算の前に正規化して、大文字と小文字の違いやフォーマットの違いを取り除きます。

値	正規化の内容	正規化された例
`domain`	たとえば、「AribaNetworkUserId」のように大文字と小文字を区別する場合を除き、小文字の文字列を使用します。「NetworkId」と「DUNS」は、大文字と小文字が区別されないことに注意してください。	`networkid`
`Identity`	先頭と末尾の空白文字は削除し、小文字の文字列を使用します。	`an9900000100`
`creationDate` `expirationDate`	正規化の必要はありません。これらは、日付、時刻およびその他のデータタイプ [26 ページ]で説明している ISO8601 フォーマットで記述されているためです。	`2003-01-15T11:42:46-08:00`

共有シークレットは、正規化しないでください。

4.1.4.3 MAC アルゴリズム

サポートされている MAC アルゴリズムの値は "`HMAC-SHA1-96`" のみです。これは、HMAC-SHA1 アルゴリズムに対応し、160 ビット (20 バイト) の出力を生成し、左側の 96 ビット (12 バイト) のみを保持します。この 12 バイトはさらに Base-64 でエンコードされ、[A-Z a-z 0-9 +/] の文字セットのみで構成される 16 バイトの文字列を生成します。

MAC を計算する手順は、次のとおりです。

1. 次に示す各文字列の末尾にヌルバイト (0x00) を付加し、それを UTF-8 でエンコードしたバイト表現にして連結します。(各文字列は、前述の方法で正規化されています。)
   “networkid”、“an9900000100”、“networkid”、“an9900000100”、“2003-01-15T08:42:46-08:00”、“2003-01-15T11:42:46-08:00”
   文字列を連結すると、次のバイトシーケンスが得られます。

   ```
   6e 65 74 77 6f 72 6b 69 64 00 61 6e 39 39 30 30
   30 30 30 31 30 30 00 6e 65 74 77 6f 72 6b 69 64
   00 61 6e 39 39 30 30 30 30 30 31 30 30 00 32 30
   30 33 2d 30 31 2d 31 35 54 30 38 3a 34 32 3a 34
   36 2d 30 38 3a 30 30 00 32 30 30 33 2d 30 31 2d
   31 35 54 31 31 3a 34 32 3a 34 36 2d 30 38 3a 30
   30 00
   ```

2. 上記のシーケンスを、HMAC-SHA1 を使用して受信者の共有シークレットでハッシュ化します。たとえば、共有シークレットが「abracadabra」(61 62 72 61 63 61 64 61 62 72 61) の場合は、次のようになります。

   ```
   71 1e 89 a7 3e 7c 9e b8 97 11 10 cd 78 57 fd a0 94 da fd
   ```

   共有シークレットは、正規化しないでください。また、終端処理も行わないでください。
3. 上記の結果から 96 ビット (12 バイト) を残して切り捨てます。

   ```
   71 1e 89 a7 3e 7c 9e b8 97 11 10 cd
   ```

   切り捨て処理によって、ハッシュのセキュリティが向上します。

4. 上記の結果を Base-64 でエンコードして、次の最終結果を得ます。

```
    cR6Jpz58nriXERDN
```

信頼できるサードパーティは、ProfileResponse ドキュメントに最終結果を挿入して、クライアントとなるエンティティ (ドキュメントの送信者) に送信します。クライアントは、サーバー (ドキュメントの受信者) と直接行うすべての通信で、CredentialMac 要素にその最終結果を挿入します。

4.1.5 ProfileResponse

次の cXML の例は、信頼できるサードパーティ (ネットワークハブなど) からクライアント (購買アプリケーションなど) に送信される ProfileResponse の例です。クライアントは、これにより受信サーバーに要求を直接送信できるようになります。

```
<cXML payloadID="1234567890@bighub.com"
  timestamp="2003-01-15T09:39:09-08:00" xml:lang="en-US">
  <Response>
    <Status code="200" text="OK"/>
    <ProfileResponse>
      <Option name="CredentialMac.type">FromSenderCredentials</Option>
      <Option name="CredentialMac.algorithm">HMAC-SHA1-96</Option>
      <Option name="CredentialMac.creationDate">2003-01-15T08:42:46
      -0800</Option>
      <Option name="CredentialMac.expirationDate">2003-01-15T11:42:46
      -0800</Option>
      <Option name="CredentialMac.value">cR6Jpz58nriXERDN</Option>
      <Transaction requestName="OrderRequest">
        <URL>https://service.hub.com/ANCXMLDispatcher.aw/ad/cxml</URL>
      </Transaction>
      <Transaction requestName="PunchOutSetupRequest">
        <URL>https://service.hub.com/AN/cxml</URL>
        <Option name="Direct.URL">https://bigsupplier.com/punchout</Option>
        <Option name="Direct.AuthenticationMethod.CredentialMac">Yes
        </Option>
        <Option name="Direct.AuthenticationMethod.Certificate">Yes</Option>
      </Transaction>
    </ProfileResponse>
  </Response>
</cXML>
```

4.1.6 CredentialMac

次に示す cXML ドキュメントの一部は、CredentialMac 要素の例です。クライアントは、このような内容を、サーバーに直接送信するドキュメントに挿入します。

```
<cXML>
  <Header>
    <To>
      <Credential domain="DUNS">
        <Identity>049329048</Identity>
      </Credential>
    </To>
    <From>
      <Credential domain="NetworkId">
```

```
        <Identity>AN9900000100</Identity>
      </Credential>
    </From>
    <Sender>
      <Credential domain="NetworkId">
        <Identity>AN9900000100</Identity>
        <CredentialMac type="FromSenderCredentials"
            algorithm="HMAC-SHA1-96"
            creationDate="2016-01-15T08:42:46-0800">
            expirationDate="2016-01-15T11:42:46-0800">
          cR6Jpz58nriXERDN
        </CredentialMac>
        <UserAgent>Procure System 3.0</UserAgent>
      </Credential>
    </Sender>
   </Header>
  [. . .]
</cXML>
```

関連情報

Credential [30 ページ]

4.2 Auth トランザクション

Auth トランザクションを利用すると、受信者は、相互に信頼しているサードパーティを通じて、組織の認証情報を検証できます。このトランザクションは、共有シークレットや MAC が含まれていないドキュメントを受け取ったときの認証に使用します。

受信者は、送信者 (プリンシパル) の認証情報を `AuthRequest` ドキュメントに入れ、信頼できるサードパーティに送信して検証を依頼します。

プリンシパルがデジタル証明書を使用してクライアント認証を行おうとしている場合、受信者が、プリンシパルの認証情報と、デジタル証明書に関する情報の両方を `AuthRequest` ドキュメントに含めます。(受信者は、この証明書情報を、その Web サーバーまたは TLS 実装から取得します。)

信頼できるサードパーティは、`AuthRequest` を受け取り、プリンシパルの認証情報を照会して、プリンシパルが認識されている組織かどうかを確認します。プリンシパルの証明書情報が含まれていた場合、信頼できるサードパーティは、その証明書が有効であることと、その証明書が認証情報に関連付けられている組織のものであることを確認します。

認証情報 (および、指定されていれば証明書) による認証が完了すると、信頼できるサードパーティは、検証済みの認証情報が含まれる肯定の `AuthResponse` で応答します。認証情報が無効な場合、信頼できるサードパーティは、空の cXML Response である状況 403 (Forbidden) を返します。

受信者は、`AuthResponse` で示される有効期限まで、Auth トランザクションの結果をキャッシュすることができます。この期間にプリンシパルから同じ認証情報と証明書が提示された場合、同じ `AuthRequest` を受信者が再度送信する必要はありません。

4.2.1 AuthRequest

エンティティを認証するために、相互に信頼しているサードパーティに送信する要求です。

次の例には、X509 証明書情報が含まれています。このデジタル証明書で、要求元エンティティのクライアント認証が行われます。

```
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML timestamp="2000-12-28T16:56:03-08:00" payloadID="foo123@bigsupplier.com">
  <Header>
    <From>
      <Credential domain="NetworkId">
        <Identity>AN99000000092</Identity>
      </Credential>
     </From>
    <To>
      <Credential domain="NetworkId">
        <Identity>AN99000000092</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkId">
        <Identity>AN99000000092</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>cXML application 2.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <AuthRequest>
      <Credential domain="DUNS">
        <Identity>12345</Identity>
      </Credential>
      <X509Data>
        <X509IssuerSerial>
          <X509IssuerName>Verisign</X509IssuerName>
          <X509SerialNumber>12345</X509SerialNumber>
        </X509IssuerSerial>
      </X509Data>
    </AuthRequest>
  </Request>
</cXML>
```

4.2.1.1 Credential

cXML の認証情報です。Credential [30 ページ] を参照してください。

4.2.1.2 X509Data

認証に使用する X.509 クライアント証明書を示します。

X509IssuerSerial

X.509 証明書のシリアル番号と発行者名を定義します。

`X509IssuerSerialChild` には以下の要素が含まれます。

- `X509IssuerName`
  X.509 証明書の発行者の識別名です。この識別名は、RFC 2253 に従って LDAP 識別名を文字列表現したものです。たとえば、次のようになります。
  `C=US, O="Mega Data Security, Inc.", OU=Secure Server CA`
- `X509SerialNumber`
  X.509 証明書のシリアル番号です。

X509SKI

X.509 証明書のサブジェクトキー識別子です。

X509 SubjectName

X.509 証明書のサブジェクトの識別名です。この識別名は、RFC 2253 に従って LDAP 識別名を文字列表現したものです。

X509Certificate

Base-64 でエンコードした X.509v3 証明書です。

X509CRL

Base-64 でエンコードした X.509v3 証明書の失効リストです。

4.2.2 AuthResponse

`AuthRequest` ドキュメントの個人エンティティの有効な認証情報のリストを返します。なお、この Response は認証に成功した場合のみ送信します。

`AuthResponse` には次の属性があります。

属性	説明
expirationDate	ここに示された日付以降は、AuthResponse に含まれている情報を破棄すべきであることを意味します。この属性が記述されている場合、受信者が expirationDate まで AuthResponse 情報をキャッシュできることを意味します。

expirationDate が記述されていない場合、キャッシュ処理は禁止されていると解釈する必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="234234@hub.com" timestamp="2001-01-25T15:19:07-08:00">
    <Response>
        <Status code="200" text="OK"/>
        <AuthResponse expirationDate="2002-12-31T09:00:00-08:00">
            <Credential domain="DUNS">
                <Identity>12345</Identity>
            </Credential>
        </AuthResponse>
    </Response>
</cXML>
```

5 cXML 電子署名

すべての cXML の Request、Response、または Message は、W3C (World Wide Web Consortium) XML Digital Signatures を使用して署名できます。XML Advanced Electronic Signature (XAdES) 標準もサポートしています。

このセクションの読者は、非対称鍵のペア、証明書、およびスマートカードなどの、電子署名の用語および概念について知識を有している必要があります。

5.1 電子署名の概要

電子署名は、電子ドキュメント送信者を識別し、署名した本人が作成した後で、そのドキュメントが変更されていないことを保証します。この署名は、暗号化情報を含むバイト列です。この情報には、署名されたドキュメントのコンテンツに関する詳細情報および送信者の公開鍵が含まれています。

XML 電子署名は、電子署名の 1 つの形態で、暗号化された署名に加え、署名の対象の一覧、署名者の公開鍵、その他の属性といった、さまざまな情報を含む要素です。この章で後述するように、cXML 署名は XML 電子署名の一形態です。

XML Advanced Electronic Signature (XAdES) では、基本認証および完全性の保護が提供されます。

W3C XML 署名および XAdES には、柔軟性を考慮して設計された多くの任意設定項目が用意されています。

W3C XML 電子署名については、次のリソースを参照してください。

- www.w3.org
- XML 署名構文と処理バージョン 1.1

XAdES の詳細については、次のリソースを参照してください。

- XML Advanced Electronic Signatures (XAdES)
- uri.etsi.org/01903/v1.3.2

5.1.1 電子署名の利用方法

ドキュメントに署名を行う代行サービスを利用することも、必要なハードウェアシステムやソフトウェアシステムを実装して、自分でドキュメントに署名を行うことも可能です。自身で署名システムを実装する場合は、受信者から信頼される認証局 (CA) による署名済み証明書を取得する必要があります。受信者の必要条件を満たすには、スマートカードやハードウェアセキュリティモジュールなどのような、秘密鍵を秘匿しておくためのハードウェアが必要になる場合があります。

署名および証明書の必要条件は、地域ごとの法律および規則によって異なることに注意してください。署名システムの実装に先立ち、該当する地域の必要条件について十分に理解しておく必要があります。

5.2 cXML ドキュメントの署名

有効な cXML 電子署名は、単なる XML 署名ではなく、特定の任意設定項目が使用され、特定の要素が存在し、ドキュメントの特定の部分に対して署名された (または、特定の部分にのみ署名されていない) XML 署名です。

5.2.1 cXML 電子署名

ほかの仕様で定義された要素を参照する場合、名前空間のプリフィックス規則が使用されることに注意してください。W3C XML のすべての電子署名の要素では `ds` プリフィックスが使用されます。また、すべての XAdES 要素では `xades` プリフィックスが使用されます。

5.2.1.1 ds:Signature 要素

`cXML` 要素では、`Request`、`Response`、または `Message` 要素の後に `ds:Signature` 要素を定義します。`ds:Signature` 要素では、署名の対象、1 つ以上の署名、および署名を作成するために使用した鍵の情報を定義します。XAdES 拡張または添付ファイルの送り状などの追加情報を挿入することもできます。

`cXML` 要素では `signatureVersion` 属性を指定することもできます。

属性	説明
`signatureVersion`	`signatureVersion` が存在する場合には、ドキュメントが電子署名付きであることを示します。このドキュメントには、`Request`、`Response`、または `Message` 要素の直後に有効な `ds:Signature` 要素が定義されています。ドキュメントが署名される場合、この属性は必須です。この属性の有効な値は `1.0` のみです。その他の値は将来の使用に備えて予約されています。
`Id`	この属性は、要素およびそのすべての子要素を署名の対象として指定するために使用します。たとえば、ドキュメントに `<Request Id="foo">` が含まれる場合、電子署名では `<Reference URI="#foo">` によって `Request` 要素およびそのすべての子要素が参照されます。ドキュメントが署名される場合、この属性は必須です。

`Message`、`Request`、および `Response` 要素には、`Id` 属性が含まれます。

関連情報

cXML エンベロープ [25 ページ]
cXML の基本 [18 ページ]

5.2.1.2 cXMLSignedInfo

cXMLSignedInfo 要素には、署名に関する cXML 固有の詳細情報を含み、次の属性があります。

属性	説明
signatureVersion (必須)	ドキュメントが電子署名付きであることを示します。このドキュメントには、Request、Response、または Message 要素の直後に有効な ds:Signature 要素が定義されています。この属性の有効な値は 1.0 のみです。その他の値は将来の使用に備えて予約されています。
payloadID (必須)	ドキュメント間のリンクを確立するために使用されます。cXMLSignedInfo 要素内の payloadID は、ドキュメント本体の cXML 要素内の payloadID と同じである必要があります。
Id (必須)	この cXMLSignedInfo 要素を署名のために識別します。この属性は常に存在し、値は必ず "cXMLSignedInfo" である必要があります。

5.2.1.3 署名の基本

cXML ヘッダーの一部の情報は重要なため、署名が必要です。これらの属性に対してヘッダーで署名するには、ds:Object 要素の中の cXMLSignedInfo 要素の中で同じ情報を再度記述します。ds:Object は、その署名の中の最初の ds:Object である必要があります。例:

```
<ds:Object>
<cXMLSignedInfo Id="cXMLSignedInfo"
             signatureVersion="1.0"
             payloadID="xxx"/>
</ds:Object>
```

Id 属性の値は "cXMLSignedInfo" である必要があります。signatureVersion および payloadID 属性の値は cXML 要素で指定された値と一致する必要があり、ドキュメントの受信者はこれらが一致していることを検証する必要があります。この ds:Reference の中では、変換することはできません。この要素は、次のように ds:SignedInfo, の最初の ds:Reference オブジェクトで署名する必要があります。

```
<ds:Reference URI="#cXMLSignedInfo">
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>xxxxxxxxxxxxxxxxxxxxxxxxxxxx</ds:DigestValue>
</ds:Reference>
```

Request、Response、または Message 要素は、その全体に対して署名される必要があります。そのためには、文字列 "cXMLData" を Request、Response、または Message 要素の Id 属性値に指定し、URI "#cXMLData" が指定された ds:Reference 要素を ds:SignedInfo に含めます。この参照に対して変換は適用できません。この ds:Reference は、ds:SignedInfo の中で 2 番目の ds:Reference である必要があります。

ds:KeyInfo 要素には、ds:X509Certificate 要素を 1 つ定義します。この要素には、ドキュメントの署名に使用された秘密鍵に対応する公開鍵が含まれる X.509 証明書の DER 表現の Base64 エンコードを定義します。

5.2.1.4 XAdES の使用

電子署名では、XAdES を使用する必要があります。署名では、xades:QualifyingProperties が 2 番目の ds:Object である必要があります。xades:SignedProperties 要素およびそのすべての子要素に署名するには、xades:SignedProperties の Id 属性値に "XAdESSignedProps" を指定し、URI に「#XAdESSignedProps」を指定した ds:Reference を ds:SignedInfo に含める必要があります。ds:SignedInfo では変換はできません。XAdES が使用される場合、xades:Cert 要素で参照される証明書は ds:KeyInfo 要素に定義するものと同一である必要があり、ds:Signature 要素の Id 属性は cXMLSignature に、xades:QualifyingProperties の Target 属性は #cXMLSignature に設定する必要があります。

5.2.1.5 添付ファイルの署名

該当するドキュメントに添付ファイルを含める場合、ドキュメントのみの署名にも、またドキュメントおよびその添付ファイルの両方の署名にも、電子署名を使用することができます。署名は、たとえ添付ファイルが破棄された場合でもドキュメント自身の署名は検証可能となるように構成されています。

添付ファイルは、その署名の中の ds:Object に含まれる ds:Manifest 要素の ds:Reference 要素を使用して署名する必要があります。ds:Manifest 要素の Id 属性は、常に「AttachmentManifest」です。ds:Object は xades:QualifyingProperties 要素を含む ds:Object が存在する場合、その直後に定義します。それ以外の場合は、cXMLSignedInfo element を含む ds:Object の直後に定義します。

送り状の中の各 ds:Reference では、スキーム "cid:" を使用した URI を使用して、MIME の Content-Id を介して添付ファイルを参照します。ds:Manifest 要素自身は、ds:SignedInfo. に含まれるフラグメント URI 参照を使用して署名する必要があります。この必要条件が存在する理由は、規格に準拠した XML 署名の実装においては、ds:SignedInfo 内のすべての ds:Reference 要素が検証される必要があるためです。基本的な検証によって、送り状自体が有効であることが保証されます。ただし、送り状の中で参照されているオブジェクトは検証されません。この方法では、添付ファイルが破棄された場合でも、ドキュメント自身は検証されることになります。例:

```
<ds:Object>
    <ds:Manifest Id="AttachmentManifest">
        <ds:Reference URI="cid:23482390498.34284203.part1@some.host.com">
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>P6ua59kKBLLtMBFE+IwPUgp2xqc=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="cid:23482390498.34284203.part2@some.host.com">
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>P6ua59kKBLLtMBFE+IwPUgp2xqc=</ds:DigestValue>
        </ds:Reference>
    </ds:Manifest>
</ds:Object>
```

5.2.2 電子署名のエラー状況コード

次の表は、cXML 電子署名の状況コードの一覧です。

状況	テキスト	意味
475	Signature Required	ドキュメントに電子署名がないため、受信者はドキュメントを受け付けようとしません。
476	Signature Verification Failed	転送中にドキュメントが変更、または署名で使用されたアルゴリズムの 1 つまたは複数を受信者がサポートしないなどの原因により、受信者は署名を検証することができません。
477	Signature Unacceptable	署名は技術的に有効ですが、その他の理由により受信者に承認されません。署名規定または証明書規定が承認されないか、使用された証明書の種類が承認されないか、またはその他の問題が存在する可能性があります。

5.2.3 電子署名の例

次に示すのは、署名済みインボイスの例です。この例では、インボイスのいくつかの部分が省略されているため、ダイジェスト値および署名の値は正しくないことに注意してください。

```
<?xml version="1.0" ?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.0.11/
InvoiceDetail.dtd">
<cXML payloadID="20030912.jdoe004@live.company.com" signatureVersion="1.0"
timestamp="200104-20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>jdoe@company.com</Identity>
      </Credential>
      </From>
      <To>
        <Credential domain="AribaNetworkUserId">
          <Identity>smistry@company.com</Identity>
        </Credential>
      </To>
      <Sender>
        <Credential domain="AribaNetworkUserId">
          <Identity>jdoe@company.com</Identity>
          <SharedSecret>abracadabra</SharedSecret>
        </Credential>
        <UserAgent>Our Invoice Application 4.0</UserAgent>
      </Sender>
  </Header>
  <Request Id="cXMLData" deploymentMode="production">
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader invoiceDate="2001-04-20T23:59:20-07:00"
        invoiceID="123456-004" operation="new"
        purpose="standard">
        ...
      </InvoiceDetailRequestHeader>
      <InvoiceDetailOrder>
        ...
      </InvoiceDetailOrder>
      <InvoiceDetailSummary>
        ...
      </InvoiceDetailSummary>
    </InvoiceDetailRequest>
  </Request>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="cXMLSignature">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/
    REC-xml-c14n20010315"></ds:CanonicalizationMethod>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
    </ds:SignatureMethod>
```

```
    <ds:Reference URI="#cXMLSignedInfo">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
      </ds:DigestMethod>
      <ds:DigestValue>mxtVp6Rg9K5wo/c5BO88g7sZYEg=</ds:DigestValue>
      </ds:Reference>
      <ds:Reference URI="#cXMLData">
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
      </ds:DigestMethod>
        <ds:DigestValue>1uBJgSa3BXewh/1wsPDWCzn8Sgk=</ds:DigestValue>
      </ds:Reference>
      <ds:Reference URI="#XAdESSignedProps">
        <ds:DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
        </ds:DigestMethod>
        <ds:DigestValue>XIasOHckorH8fz/thdyZIZvV2yI=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
nNfsBpc22u9aypYLvgE5cuiHVO077vnaolS76LoAuks9bAwLO0kz/nkTQfb2zKSQTy8jj6W/
TJGCQj691PlKBnIqaMPPN3k+hbi6A5cJHPRd3HNPexU5sSi4StTuxlWAiHe/
XEeBEeclu7K6sR4Rh1gzzELg05v21aRX4oVGbjk=</ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>
MIICgDCCAekCAw7cUTANBgkqhkiG9w0BAQQFADCBijELMAkGA1UEBhMCVV
w7cUTANBgkqhkiG9w0BAQQFADCBijELMAkGA1UEBhMCVVMxEzARBgNV
MIICgDCCAekCAw7cUTANBgkqhkiG9w0BAQQFADCBijELMAkGA1UEBhMCVV
w7cUTANBgkqhkiG9w0BAQQFADCBijELMAkGA1UEBhMCVVMxEzARBgNVBA
MIICgDCCAekCAw7cUTANBgkqhkiG9w0BAQQFADCBijELMAkGA1UEBhMCVV
zuRel/9tb8M95FuN5yR9GUGl5PgkzwuCQYobJqIcAs=</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
      <ds:Object>
        <cXMLSignedInfo Id="cXMLSignedInfo"
          payloadID="20030912.rsmith004@live.hub.com" signatureVersion="1.0">
        </cXMLSignedInfo>
      </ds:Object>
      <ds:Object>
        <xades:QualifyingProperties xmlns:xades=
        "http://uri.etsi.org/01903/v1.1.1#"
        Target="#cXMLSignature">
        <xades:SignedProperties Id="XAdESSignedProps">
           <xades:SignedSignatureProperties>
             <xades:SigningTime>2003-09-30T18:32:27Z</xades:SigningTime>
               <xades:SigningCertificate>
                 <xades:Cert>
                   <xades:CertDigest>
                     <ds:DigestMethod
                         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
                     </ds:DigestMethod>
                     <ds:DigestValue>LETnT8c7gvZqp3oVt8/BLOJpeeA=
                     </ds:DigestValue>
                   </xades:CertDigest>
                   <xades:IssuerSerial>
                     <ds:X509IssuerName>EMAILADDRESS=an_ops@company.com,
                       CN=anrc.hub.com, O="Hub, Inc.", L=Mountain View,
                       ST=California, C=US</ds:X509IssuerName>
                     <ds:X509SerialNumber>973905</ds:X509SerialNumber>
                   </xades:IssuerSerial>
                 </xades:Cert>
               </xades:SigningCertificate>
             <xades:SignaturePolicyIdentifier>
               <xades:SignaturePolicyImplied>
               </xades:SignaturePolicyImplied>
             </xades:SignaturePolicyIdentifier>
           </xades:SignedSignatureProperties>
        </xades:SignedProperties>
      </xades:QualifyingProperties>
```

```
    </ds:Object>
  </ds:Signature>
</cXML>
```

6 改訂履歴

次の表は、このガイドの改訂履歴の概要です。

更新時期	更新されたトピック	更新内容
2021 年 1 月	cXML の基本	トピック「Correspondent」および「Routing」を追加/更新。
2020 年 10 月	cXML の基本	トピック「ドキュメントにおける特殊文字の処理」を更新。
2020 年 1 月	cXML の基本	トピック「ベース要素」を更新 (`State` 要素を追加)。
2019 年 10 月	cXML の基本	トピック「Type エンティティ」を更新 (`isoStateCode` を追加)。
	複数のページ	整合性を確保するため、ネットワークハブの表記を更新。
2018 年 7 月	タイトルページ	誤った cXML バージョンを記載していたサブタイトルを削除。本書は、すべての cXML バージョンに適用されます。
2018 年 4 月	該当なし	最初のバージョン