



cXML User's Guide

VERSION 1.2.011

APRIL , 2004

cXML License Agreement

IMPORTANT: PLEASE CAREFULLY READ THIS cXML LICENSE AGREEMENT ("LICENSE") BEFORE USING THE cXML SPECIFICATION ("SPECIFICATION"). BY USING THE SPECIFICATION, YOU AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, DO NOT USE OR ACCESS THE SPECIFICATION. This License may be modified at any time upon posting of the modified agreement on the cXML site (www.cXML.org) and any such modifications shall be effective immediately. You agree to review the cXML site periodically, and each use by you shall constitute and be deemed your unconditional acceptance of the then-current License.

1. Openness. cXML is designed and intended to be an open standard to facilitate electronic commerce. You are welcome to use and adopt this standard, and to submit comments, recommendations, and suggestions to cXML.org. Once submitted, your comments go through an approval process – and your comments may ultimately be incorporated into cXML.

2. License. Subject to the terms and conditions herein, Licensor hereby grants to you a perpetual, nonexclusive, royalty-free, worldwide right and license to use the Specification under any Licensor intellectual property rights in the Specification to (a) use, copy, publish, and distribute the unmodified Specification, and (b) to implement and use the cXML tags and schema guidelines included in the Specification for the purpose of creating computer programs that adhere to such guidelines. If you use, publish, or distribute the unmodified Specification, you may call it "cXML".

3. Restrictions. Your rights under this License will terminate automatically without notice from Licensor if you fail to comply with any terms of this License. Because the Specification is intended to be an open standard, you agree to not assert any intellectual property rights against Licensor or any other entity for its use of the Specification.

Licensor expressly reserves all other rights it may have in the material and subject matter of this Specification, and you acknowledge and agree that Licensor owns all right, title, and interest in and to all Specification, and all derivations thereof. If you publish, copy or distribute this Specification, then this Agreement notice must be attached. If you modify this Specification, the name of the modified specification shall NOT include the term "cXML" in the new name. If you submit any comments or suggestions to Licensor, and Licensor modifies cXML based on your input, Licensor shall own the modified cXML version.

4. No Warranty. YOU ACKNOWLEDGE AND AGREE THAT ANY USE OF THE SPECIFICATION BY YOU IS AT YOUR OWN RISK. THE SPECIFICATION IS PROVIDED FOR USE "AS IS" WITHOUT WARRANTY OF ANY KIND. LICENSOR AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES OF ANY KIND, INCLUDING BUT NOT LIMITED TO ANY EXPRESS WARRANTIES, STATUTORY WARRANTIES, AND ANY IMPLIED WARRANTIES OF: MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. YOUR SOLE AND EXCLUSIVE REMEDY RELATING TO YOUR USE OF THE SPECIFICATION SHALL BE TO DISCONTINUE USING THE SPECIFICATION.

5. Limitation of Liability. TO THE MAXIMUM EXTENT PERMITTED BY LAW, UNDER NO CIRCUMSTANCES SHALL LICENSOR BE LIABLE FOR ANY DAMAGES WHATSOEVER RELATING TO THIS LICENSE OR YOUR USE OF THE SPECIFICATION (INCLUDING BUT NOT LIMITED TO INCIDENTAL, SPECIAL, PUNITIVE, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES), REGARDLESS OF WHETHER A CLAIM IS BASED ON TORT, CONTRACT, OR OTHER THEORY OF LIABILITY, AND EVEN IF LICENSOR IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. To the extent your jurisdiction does not allow any of the above exclusions of damages, in such case you agree that Licensor's total liability to you for all damages under this License shall not exceed the amount of ten dollars (\$10.00).

6 Government End Users. If the Specification is supplied to the United States Government, the Specification is classified as "restricted computer software" as defined in clause 52.227-19 of the FAR. The United States Government's rights to the Specification are as provided in clause 52.227-19 of the FAR.

7. This License shall be deemed to have been made in, and shall be construed pursuant to the laws of the State of California and the federal U.S. laws applicable therein, excluding its conflict of laws provisions. Any legal action or proceeding relating to this License shall be instituted in a state or federal court in San Francisco, Santa Clara or San Mateo County, California, and each party hereby consents to personal jurisdiction in such counties. If for any reason a court of competent jurisdiction finds any provision, or portion thereof, to be unenforceable, the remainder of this License shall continue in full force and effect.

8. You assume the entire risk resulting from your use of the Specification and agree to hold harmless, indemnify, and defend Licensor, its officers, directors, employees and agents from and against any losses, damages, fines and expenses (including attorneys' fees and costs) arising out of or relating to any claims alleging that you have violated a third party's rights.

9. Complete Agreement. This License is the complete and exclusive statement, and an absolute integration of the mutual understanding of the parties and supersedes and cancels all previous written and oral agreements and communications relating to the subject matter of this License. You acknowledge that any breach by you of the provisions of the License will cause irreparable damage to Licensor and that a remedy at law will be inadequate. Therefore, in addition to any and all other legal or equitable remedies, Licensor will be entitled to injunctive relief for any breach of this License. Ariba, Inc. shall be deemed the Licensor.

10. Notices. Any notice directed to Licensor must be sent in writing to comments@cXML.org.

Information in this document is subject to change without notice.

3-07-02

Table of Contents

Preface	xv
Audience and Prerequisites	xv
Which Chapters to Read	xv
Typography	xvi
 Chapter 1	
Introduction to cXML	15
cXML, an XML Implementation	15
cXML Capabilities	16
Catalogs	16
PunchOut	17
Purchase Orders	19
Types of Applications that Use cXML	20
Procurement Applications	20
Commerce Network Hubs	20
PunchOut Catalogs	20
Order-Receiving Systems	21
Content Delivery Strategy	21
cXML DTDs	22
Getting cXML DTDs	23
Caching DTDs	23
Profile Transaction	24
ProfileRequest	24
ProfileResponse	24
Service Status Response	24
XML Utilities	24

Chapter 2	
cXML Basics	27
Protocol Specification	27
Request-Response Model	27
cXML Conventions	29
cXML Document	29
Wrapping Layers	30
Attachments	31
cXML Envelope	36
Special Characters	37
Header	40
Request	44
Response	44
One-Way (Asynchronous) Model	48
Message	49
Transport Options	50
Service Status Response	53
Basic Elements	54
Type Entities	54
Base Elements	55
Chapter 3	
Profile Transaction	57
Introduction to the Profile Transaction	57
ProfileRequest	58
ProfileResponse	58
Option Element	59
Transaction	62
Scenarios	63
From Buyer to Supplier	63
From Buyer to the Network	64
From a Network to Supplier	66
From the Network to Service Provider	67
From a Network to Buyer	68
From Service Provider to Buyer	68

Chapter 4	
PunchOut Transaction	69
PunchOut Requirements	69
Buying Organizations	69
Suppliers	71
PunchOut Event Sequence	73
Steps 1 & 2: PunchOut Request	73
Step 3: Product Selection	74
Step 4: Check Out	75
Step 5: Transmittal of Purchase Order	76
PunchOut Documents	78
PunchOut Index Catalog	78
PunchOutSetupRequest	79
PunchOutSetupResponse	84
PunchOutOrderMessage	85
Modifications to the Supplier's Web Pages	87
Launch Page	87
Start Page	91
Sender Page	91
Order Receiver Page	95
PunchOut Website Suggestions	95
Implementation Guidelines	95
Buyer and Supplier Cookies	96
Personalization	96
PunchOut Transaction	97
Sourcing	98
PunchOutSetupRequest	98
PunchOutSetupResponse	103
PunchOutOrderMessage	104
Direct PunchOut	110
Authentication Methods	110
ProfileResponse	110

Chapter 5	
Purchase Orders	111
Purchase Order Process	111
OrderRequest Documents	112
OrderRequestHeader Element	116
ItemOut	125
Distribution	152
Response to an OrderRequest	154
Accepting Order Attachments	154
Chapter 6	
Path Routing	157
Overview of Path Routing	157
Nodes	158
Path Element	158
Router Nodes	159
Copy Nodes	160
Adding Nodes to PunchOutOrderMessage	160
Path Element	161
Credentials	161
Creating OrderRequests	162
Path Element	162
Credentials	162
Other Routable Documents	164
PunchOutSetupRequest	164
ConfirmationRequest and ShipNoticeRequest	164
CopyRequest	165

Chapter 7	
Payment	167
Overview of Payment	167
PaymentRemittance DTD	168
Working with Payment Documents	168
PaymentRemittanceRequest	168
PaymentRemittanceRequestHeader	170
PaymentRemittanceSummary	177
RemittanceDetail	177
PaymentRemittanceStatusUpdateRequest	180
DocumentReference	181
PaymentRemittanceStatus	181
Payment Document Examples	182
PaymentRemittanceRequest Example	182
PaymentRemittanceStatusUpdateRequest Example	185
Chapter 8	
TimeCard Transaction	187
TimeCard Requests	187
Supplier to Buyer Request	188
Buyer to Supplier Request	188
TimeCard Element	188
OrderInfo	189
Contractor	189
ReportedTime	189
SubmitterInfo	191
ApprovalInfo	191
DocumentReference	191
TimeCard Examples	192
Chapter 9	
Master Agreements	197
Overview of Master Agreements	197
MasterAgreementRequest	197
MasterAgreementRequestHeader Element	199
AgreementItemOut Element	200

Chapter 10	
Later Status Changes	201
Overview of Status	201
StatusUpdateRequest	202
DocumentReference Element	203
PaymentStatus Element	204
SourcingStatus Element	205
InvoiceStatus Element	206
ConfirmationRequest	206
OrderReference Element	209
ConfirmationHeader Element	209
ConfirmationItem Element	216
ShipNoticeRequest	220
ShipNoticeHeader Element	222
ServiceLevel Element	224
Route Element	226
CarrierIdentifier Element	227
ShipmentIdentifier Element	228
PackageIdentification Element	228
ShipNoticePortion Element	229
ShipNoticeItem Element	230
OrderReference Element	232

Chapter 11

Invoices	233
Overview of Invoices	233
Early InvoiceRequest Document	234
Debit and Credit Amounts	234
Shipping Information	234
Types of Invoices	234
Invoice DTD	236
InvoiceDetailRequest	236
InvoiceDetailRequestHeader	237
InvoiceDetailOrder	242
InvoiceDetailHeaderOrder	252
InvoiceDetailSummary	254
Response	255
Invoice Status Update	255
Example Invoices	257
Standard Header Invoice	257
Standard Detail Invoice	261
Service Invoice	265
Marketplace Invoice	270

Chapter 12

Catalogs	271
Catalog Definitions	271
Supplier	272
Index	274
Type Definitions	277
TypeProvider	277
Type	278
TypeAttribute	279
PrimitiveType	281
Subscription Management Definitions	281
Supplier Data	282
Catalog Subscriptions	285
Catalog Upload Transaction	288
CatalogUploadRequest	289
Response	293

Chapter 13**GetPending Transaction 297**

GetPendingRequest. 297

GetPendingResponse 298

Chapter 14**Provider PunchOut Transaction 301**

Message Flow 302

ProviderSetupRequest Document 302

Header 302

Request 303

Sample 305

ProviderSetupResponse Document 306

Sample 308

ProviderDoneMessage Document 308

Header 308

Message 309

OriginatorCookie 310

ReturnData 310

ReturnValue 310

Sample 311

Chapter 15**Alternative Authentication Methods 313**

Message Authentication Code (MAC) 313

Overview of MACs 313

Computation Algorithm 314

Creation and Expiration Dates 314

Computation Process 315

ProfileResponse 317

CredentialMac 317

Auth Transaction 318

AuthRequest 319

AuthResponse 321

Chapter 16

cXML Digital Signatures	323
Digital Signature Overview	323
Options for Signing	324
Signing cXML Documents	324
cXML Digital Signatures	324
Error Status Codes for Digital Signatures	327
Digital Signature Example	327

Appendix A

New Features in cXML 1.2.011	331
CopyRequest Enhancement	331
InvoiceDetailPaymentTerm Deprecated	332
New PaymentTerm Element	332
Digital Signature Support	332
New SpendDetail Element	332
TravelDetail Element	332
FeeDetail Element	333
LaborDetail Element	333
Extrinsic	333
New TimeCard Element	333
New LeadTime Element	334
Limited Changes in Use of UnitOfMeasure and UnitRate	334
New preferredLang Attribute	334
SearchDataElement Change	335
New inspectionDate Attribute	335
Followup Element Deprecated	335

Index	337
--------------	------------

Preface

This document describes how to use cXML (commerce eXtensible Markup Language) for communication of data related to electronic commerce.

Audience and Prerequisites

This document is intended for application developers who design cXML-enabled applications.

cXML is an open versatile language for the transaction requirements of:

- Network e-commerce hubs
- Electronic product catalogs
- PunchOut catalogs
- Procurement applications
- Buying communities
- E-commerce service providers

Readers should have a working knowledge of e-commerce concepts, the HTTP Internet communication standard, and XML format.

This document does not describe how to use specific procurement applications or commerce network hubs.

Which Chapters to Read

- **E-commerce Business Managers**—For an overview of cXML capabilities, read Chapter 1, “[Introduction to cXML](#).”
- **Web Programmers**—Web programmers who implement e-commerce sites should read all chapters.

- **PunchOut Site Administrators**—Web engineers experienced with PunchOut websites should read Appendix A, [“New Features in cXML 1.2.011.”](#)

Typography

cXML elements and attributes are denoted with a monotype font. cXML element and attribute names are case-sensitive. Both are a combination of lower and uppercase, with elements beginning with an uppercase letter, and attributes beginning with a lowercase letter. For example, MyElement is a cXML element, and myAttribute is a cXML attribute.

The following table describes the typographic conventions used in this book:

Typeface or Symbol	Meaning	Example
<AaBbCc123>	Text you need to change is italicized, and appears between angle brackets.	<i>http://<server>:<port>/inspector</i>
AaBbCc123	The names of user interface controls, menus, and menu items.	Choose Edit from the File menu.
AaBbCc123	Files and directory names, parameters, fields in CSV files, command lines, and code examples.	There is one line in ReportMeta.csv for each report in the system.
<i>AaBbCc123</i>	The names of books.	For more information, see <i>Acme Configuration Overview</i> .

Chapter 1

Introduction to cXML

This chapter introduces cXML (commerce eXtensible Markup Language) for electronic-commerce transactions.

This chapter describes:

- [cXML, an XML Implementation](#)
- [cXML Capabilities](#)
- [Types of Applications that Use cXML](#)
- [Content Delivery Strategy](#)
- [cXML DTDs](#)
- [Profile Transaction](#)
- [XML Utilities](#)

cXML, an XML Implementation

XML (eXtensible Markup Language) is a meta-markup language used to create syntaxes for languages. It is also a standard for passing data between applications, particularly those that communicate across the Internet.

XML documents contain data in the form of tag/value pairs, for example:

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

XML has a structure similar to HTML (HyperText Markup Language), which is an implementation of SGML, XML's parent meta language. But, applications can extract and use data from XML documents more easily than from HTML ones, because in XML, all data is tagged according to its purpose. XML contains only data, while HTML contains both data and presentation information.

Each cXML document is constructed based XML Document Type Definitions (DTDs). Acting as templates, DTDs define the content model of a cXML document, for example, the valid order and nesting of elements, and the data types of attributes.

The DTDs for cXML are files available on the www.cXML.org website. For more information, see “[Getting cXML DTDs](#)” on page 23.

cXML Capabilities

cXML allows buying organizations, suppliers, service providers, and intermediaries to communicate using a single, standard, open language.

Successful business-to-business electronic commerce (B2B e-commerce) portals depend upon a flexible, widely adopted protocol. cXML is a well-defined, robust language designed specifically for B2B e-commerce, and it is the choice of high volume buying organizations and suppliers.

cXML transactions consist of *documents*, which are simple text files containing values enclosed by predefined tags. Most types of cXML documents are analogous to hardcopy documents traditionally used in business.

The most commonly used types of cXML documents are:

- [Catalogs](#)
- [PunchOut](#)
- [Purchase Orders](#)

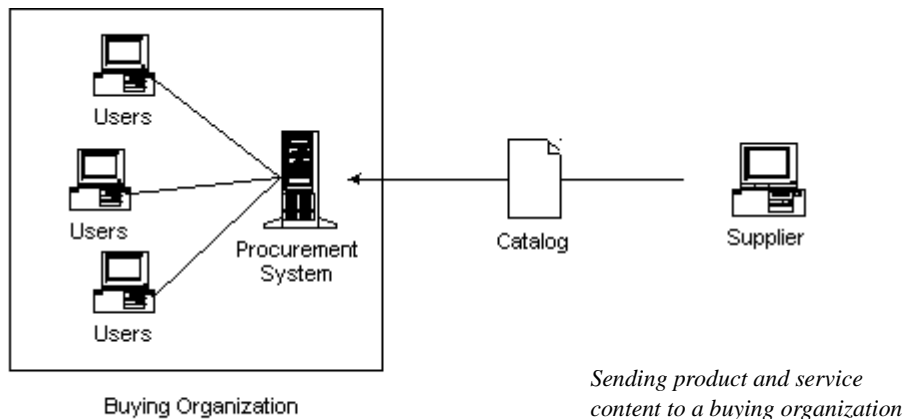
The following subsections describe these cXML documents.

Catalogs

Catalogs are files that convey product and service content to buying organizations. They describe the products and services offered by a supplier and their prices, and they are the main communication channel from suppliers to their customers.

Suppliers create catalogs so that organizations that use procurement applications can see their product and service offerings and buy from them. Procurement applications read catalogs and store them internally in their databases. After a buying organization

approves a catalog, that content is visible to users, who can choose items and add them to purchase requisitions.



Suppliers can create catalogs for any product or service, regardless of how it is measured, priced, or delivered.

For each item in a catalog, basic information is required, and optional information enables advanced catalog features, such as multi-language descriptions.

PunchOut

PunchOut is an easy-to-implement protocol for interactive sessions managed across the Internet. Using real-time, synchronous cXML messages, PunchOut enables communication between applications, providing seamless user interaction at remote sites.

There are three types of PunchOut:

- [Procurement PunchOut](#)
- [PunchOut Chaining](#)
- [Provider PunchOut](#)

Procurement PunchOut

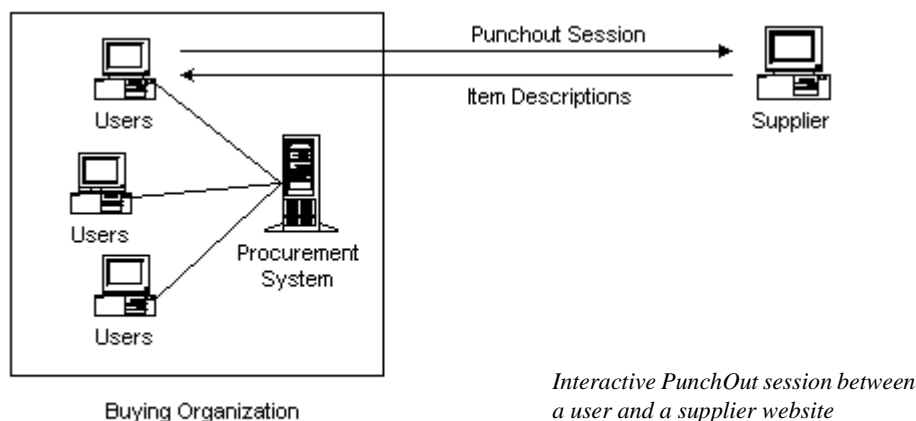
Procurement PunchOut gives suppliers an alternative to static catalog files. PunchOut sites are live, interactive catalogs running on a website.

Suppliers that have e-commerce websites can modify them to support PunchOut. PunchOut sites communicate with procurement systems over the Internet by using cXML.

For more information:

Chapter 4, “PunchOut Transaction.”

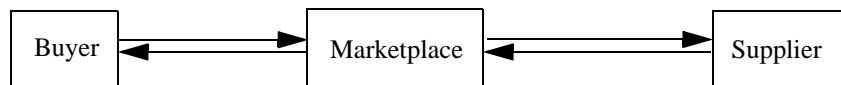
For PunchOut sites, procurement applications display a button instead of product or pricing details. When users click this button, their Web browsers display pages from the supplier’s local website. Depending on how the supplier implements these pages, users can browse product options, specify configurations, and select delivery methods. When users are done selecting items, they click a button that returns the order information to the procurement application. The fully configured products and their prices appear within users’ purchase requisitions.



Suppliers’ websites can offer previously agreed-upon contract products and prices.

PunchOut Chaining

PunchOut chaining is Procurement PunchOut that involves more than one PunchOut. cXML Path Routing enables this functionality.



cXML Path Routing allows the order and other subsequent messages to return to the marketplaces and suppliers involved in producing the quote. Path Routing notifies all parties about the final order, and any subsequent PunchOut specifies to the procurement application how to split orders on behalf of the marketplace.

Provider PunchOut

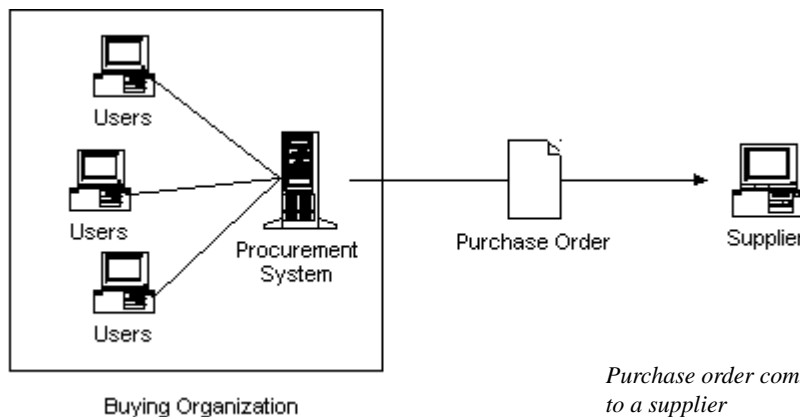
Provider PunchOut enables applications to punch out to a remote applications that supply services to the originating application, such as credit card validation, user authentication, or self-registration.

Purchase Orders

Buying organizations send purchase orders to suppliers to request fulfillment of a contract.

For more information:

Chapter 5, "Purchase Orders."



cXML is better for communicating purchase orders than other formats (such as ANSI X12 EDI 850), because it is flexible, inexpensive to implement, and it supports the widest array of data and attachments.

Types of Applications that Use cXML

cXML can be used by any e-commerce application. It is currently used by buying organizations, vertical and horizontal buying communities, suppliers, and application vendors. The following subsections describe the main types of applications that currently use cXML.

Procurement Applications

Procurement applications, such as Ariba Buyer and Ariba Marketplace, Network Edition, use cXML for external transactions.

Ariba Buyer is an enterprise application hosted by large organizations for use by their employees over an intranet.

Ariba Marketplace, Network Edition, is an Internet-based service that allows the creation of buying communities composed of many small- to medium-sized businesses.

These applications allow communities of users to buy contract products and services from vendors approved by their purchasing managers. Managers in the communities first approve requested purchases, and approved purchase orders are transmitted to suppliers through several possible channels, including cXML over the Internet.

Commerce Network Hubs

Commerce network hubs, such as Ariba Supplier Network, are Web-based services for connecting buyers and suppliers. These Web services provide features such as catalog validation and versioning, catalog publishing and subscription, automated purchase order routing, and purchase order history.

Commerce network hubs can act as intermediaries that authenticate and route requests and responses to and from diverse organizations. Communication between these organizations can occur entirely through cXML over the Internet.

PunchOut Catalogs

As described in the previous section, PunchOut catalogs are interactive catalogs, available at supplier websites. PunchOut catalogs are made possible by Web server applications, written in a programming language such as ASP (Active Server Pages), JavaScript, or CGI (Common Gateway Interface), that manage buyers' PunchOut sessions.

For more information:

Chapter 4, “PunchOut Transaction.”

PunchOut catalogs accept PunchOut requests from procurement applications, identify the buying organization, and display the appropriate products and prices in HTML format. Users then select items, configure them, and select options if appropriate.

At the end of the PunchOut session, the PunchOut site sends descriptions of the users’ selections, in cXML format, to the procurement applications.

Order-Receiving Systems

For more information:

Chapter 5, “Purchase Orders.”

Order-receiving systems are applications at supplier sites that accept and process purchase orders sent by buying organizations. Order-receiving systems can be any automated system, such as inventory management systems, order-fulfillment systems, or order-processing systems.

Because it is simple to extract information from cXML purchase orders, it is relatively easy to create the adapters that enable existing order-receiving systems to accept them.

Content Delivery Strategy

Procurement applications present product and service content to users. Suppliers want to control the way their customers view their products or services, because presentation is critical to their sales process. Buying organizations want to make content easily accessible and searchable to ensure high contract compliance.

Buying organizations and suppliers can choose from multiple methods for delivering product and service content. The particular method to use is determined by agreement between a buying organization and a supplier, and the nature of the products or services traded.

The following table lists example categories of commonly procured products and services, and their preferred content delivery methods.

Commodities	Properties	Content Delivery Method
Office Supplies, Internal Supplies	Static content, stable pricing	Static catalogs
Lab Supplies, MRO (Maintenance, Repair, and Operations), Electronic Parts	Requires normalization to be useful	PunchOut to a vertical commodity portal
Books, Chemicals	Large number of line items	PunchOut to a supplier hosted site
Computers, Network Equipment, Peripherals	Many possible configurations	PunchOut to a supplier hosted configuration tool
Services, Printed Materials	Content has highly variable attributes	PunchOut to an electronic form at a supplier site

Buying organizations can either store content locally within the organization, or they can access it remotely on the Internet, through PunchOut. cXML catalogs support both storage strategies.

As this table indicates, PunchOut offers a flexible framework upon which suppliers, depending on their commodity or customer, can provide customized content. The objective of this content strategy is to allow buyers and suppliers to exchange catalog data by the method that makes the most sense.

cXML DTDs

Because cXML is an XML language, a set of Document Type Definitions (DTDs) thoroughly define it. These DTDs are text files that describes the precise syntax and order of cXML elements. DTDs enable applications to validate the cXML they read or write.

The header of each cXML document contains the URL to the DTD that defines the document. cXML applications can retrieve the DTD and use it to validate the document.

For the most robust transaction handling, validate all cXML documents received. If you detect errors, issue the appropriate error code so the sender can retransmit. cXML applications are not required to validate cXML documents received, although it is recommended. However, all cXML documents must be valid and must refer to the cXML DTDs described in the following section.

Getting cXML DTDs

DTDs for all versions of cXML are available on cXML.org. The various kinds of cXML documents are defined in multiple DTDs to reduce DTD size, which enables faster validation in some parsers.

Document	DTD
Basic cXML documents	<a href="http://xml.cXML.org/schemas/cXML/<version>/cXML.dtd">http://xml.cXML.org/schemas/cXML/<version>/cXML.dtd
Confirmation and Ship Notice	<a href="http://xml.cXML.org/schemas/cXML/<version>/Fulfill.dtd">http://xml.cXML.org/schemas/cXML/<version>/Fulfill.dtd
Invoice	<a href="http://xml.cXML.org/schemas/cXML/<version>/InvoiceDetail.dtd">http://xml.cXML.org/schemas/cXML/<version>/InvoiceDetail.dtd
Type Definition	<a href="http://xml.cXML.org/schemas/cXML/<version>/Catalog.dtd">http://xml.cXML.org/schemas/cXML/<version>/Catalog.dtd
Payment Remittance	<a href="http://xml.cXML.org/schemas/cXML/<version>/PaymentRemittance.dtd">http://xml.cXML.org/schemas/cXML/<version>/PaymentRemittance.dtd

where *<version>* is the full cXML version number, such as 1.2.011.

cXML applications use these DTDs to validate all incoming and outgoing documents.

Caching DTDs

For best performance, cXML applications should cache DTDs locally. After cXML DTD files are published, they never change, so you can cache them indefinitely. (Each new version of the DTDs has a new URL). When cXML applications parse a cXML document, they should look at the SYSTEM identifier in the document header and retrieve that DTD if it has not already been stored locally.

Caching DTDs locally offers the advantages of faster document validation and less dependence on the cXML.org site.

In some environments, cXML applications might not be allowed to automatically retrieve DTDs as they receive new documents. In these environments, you must manually retrieve the DTDs, store them locally, and instruct your applications to look for them locally, not at cXML.org. However, generated cXML documents must point to the DTDs at cXML.org, not the local DTDs.

Profile Transaction

The Profile transaction communicates basic information about what transactions a particular cXML server can receive. All cXML servers must support this transaction. It is intended for back-end integrations between applications, making the capabilities of cXML servers available to client systems.

This transaction consists of two documents, ProfileRequest and ProfileResponse. Together, they retrieve server capabilities, including supported cXML version, supported transactions, and options to those transactions.

Note: All cXML 1.1 and higher servers **must** support the Profile transaction.

ProfileRequest

The ProfileRequest document has no content. It simply routes to the specified cXML server.

ProfileResponse

The server responds with a ProfileResponse document, which lists the cXML transactions it supports, their locations, and any named options with a string value.

Service Status Response

A response with a status code of 200 from an URL that accepts POSTed cXML is up and running. When an HTTP GET is sent to a service location, the service responds with a valid, dynamically generated cXML Response document. A service can be any HTTP URL at which cXML Request documents are received.

XML Utilities

Utilities for editing and validating XML files are available for free and for purchase on the Web. The following listing describes a few of these utilities:

- **Internet Explorer** from Microsoft. An XML-aware Web browser that can validate XML files against DTDs.

www.microsoft.com/windows/ie/default.htm

- **Extensibility** from TIBCO Software. A Java-based XML Schema and DTD editor, with an intuitive tree-based graphical user interface.

www.tibco.com/solutions/products/extensibility

- **XML Spy** from Altova. A tool for maintaining DTDs and XML files with a grid, source and browser view.

www.icon-is.com

- **XMetaL** from Softquad Software. A customizable XML authoring tool.

www.softquad.com

- **XMLwriter** from Wattle Software. A graphical XML authoring tool designed to manage XML projects.

www.xmlwriter.net

In addition, the following websites list more XML tools:

www.xmlsoftware.com

www.xml.com

Chapter 2

cXML Basics

This chapter describes the basic protocol and data formats of cXML. It contains information needed to implement all transactions.

This chapter describes:

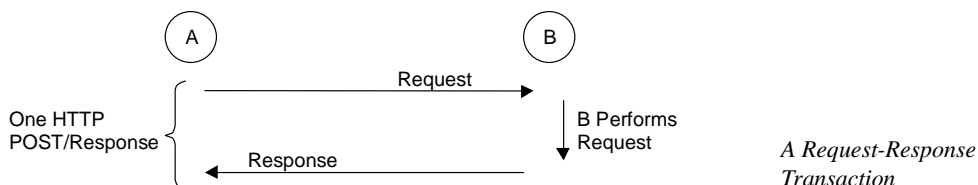
- [Protocol Specification](#)
- [Basic Elements](#)

Protocol Specification

There are two communication models for cXML transactions: Request-Response and One-Way. Because these two models strictly specify the operations, they enable simple implementation. Both models are required, because there are situations when one model would not be appropriate.

Request-Response Model

Request-Response transactions can be performed only over an HTTP or HTTPS connection. The following figure illustrates the steps in a Request-Response interaction between parties A and B:



This transaction contains the following steps:

1. *Site A* initiates an HTTP/1.x connection with *Site B* on a predetermined URL that represents *Site B*'s address.
2. *Site A* uses a POST operation to send the cXML document through the HTTP connection. *Site A* then waits for a response.
3. *Site B* has an HTTP/1.x-compliant server that dispatches the HTTP Request to the resource specified by the URL used in step 1. This resource can be any valid location known to *Site B*'s HTTP server, for example, a CGI program or an ASP page.
4. *Site B*'s resource identified in step 3 reads the cXML document contents and maps the Request to the appropriate handler for that request.
5. *Site B*'s handler for the cXML Request performs the work that the Request specifies and generates a cXML Response document.
6. *Site B* sends the cXML Response to *Site A* through the HTTP connection established in step 1.
7. *Site A* reads the cXML Response and returns it to the process that initiated the Request.
8. *Site A* closes the HTTP connection established in step 1.

This process is then repeated for further Request/Response cycles.

To simplify the work in the above steps, cXML documents are divided into two distinct parts:

- Header—Contains authentication information and addressing.
- Request or Response data—Contains a specific request or response and the information to be passed.

Both of these elements are carried in a parent envelope element. The following example shows the structure of a cXML Request document:

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

The following example shows the structure of a cXML Response document:

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

The Response structure does not use a Header element. It is not necessary, because the Response always travels in the same HTTP connection as the Request.

cXML Conventions

cXML uses elements to describe discrete items, which are properties in traditional business documents. Elements also describe information with obvious subdivisions and relationships between those subdivisions, such as an addresses, which are composed of street, city, and country.

cXML also uses attributes, which modify elements or provide context.

Element and attribute names are case-sensitive and use whole words with capitals (not hyphens) separating the words. Element names begin with an uppercase letter; attribute names begin with a lowercase letter, for example:

```
Elements:  Sender, Credential, Payment, ItemDetail
Attributes: payloadID, lineNumber, domain
```

If optional elements have no content (they are null), leave them out entirely. Avoid empty or whitespace elements, because missing values can affect some parsers.

In DTD files and in this document, symbols are used to indicate how many times an element can occur in a transaction. A '+' means the element can occur one or more times, a '?' means the element can occur 0 or once, and a '*' means the element can occur 0 or more times.

cXML Document

The cXML element is the body of a cXML document. A document might begin as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE cXML SYSTEM "http://xml.cxm1.org/schemas/cXML/1.2.0111.2.011/
cXML.dtd">
<cXML xml:lang="en-US"
  payloadID="1234567.4567.5678@test.ariba.com"
```

```
timestamp="2002-01-09T01:36:05-08:00">
...
```

The first characters in cXML documents must be `<?` or `<!`. Documents must not start with white space or tabs. For example, the HTML form that contains a `PunchOutOrderMessage` document must not insert any character between the opening quote and the left angle bracket.

The second line in cXML documents must contain the DOCTYPE document type declaration. This is the only external entity that can appear in cXML documents. This line references the cXML DTD. See [“cXML DTDs”](#) on page 22 for more information about cXML DTDs.

cXML documents can have any one of the following top-level elements: `cXML`, `Supplier`, `Contract`, and `Index`. The `cXML` element is for “transactional” data. The other elements describe static content.

Wrapping Layers

cXML documents are usually transmitted through HTTP with the HTTP header specifying a MIME (Multipurpose Internet Mail Extensions) media type of `text/xml` and a `charset` parameter matching the encoding in the cXML document.

Because HTTP is eight-bit clean, any character encoding supported by the receiving parser can be used without a content-transfer encoding such as `base64` or `quoted-printable`. All XML parsers support the UTF-8 (Universal Transformation Format) encoding, which includes all Unicode characters, including all of US-ASCII. Therefore, applications should use UTF-8 when transmitting cXML documents.

Note: According to IETF RFC 2376 “XML Media Types,” the MIME `charset` parameter overrides any encoding specified in the XML declaration. Further, the default encoding for the `text/xml` media type is `us-ascii`, not UTF-8 as mentioned in Section 4.3.3 of the XML Specification. For clarity, cXML documents should include an explicit encoding in the XML declaration. MIME envelopes should use a matching `charset` parameter for the `text/xml`. You can also use the `application/xml` media type, which does not override the XML declaration or affect the recipient's decoding notes, and which does not require the `charset` parameter.

An HTTP transmission of a cXML document might include the following MIME and HTTP headers:

```
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"?>
...
```

Attachments

The cXML protocol supports the attachment of external files of any type to cXML documents. For example, buyers sometimes need to clarify purchase orders with supporting memos, drawings, or faxes. Another example is the `CatalogUploadRequest` document, which includes catalog files as attachments.

Files referenced by cXML documents can reside either on a server accessible by the receiver or within an envelope that also includes the cXML documents themselves. To attach external files to a cXML document in a single envelope, use Multipurpose Internet Mail Extensions (MIME). The cXML document contains references to external parts sent within a multipart MIME envelope.

Including Attachments

A cXML requirement for this envelope (over the requirements described in IETF RFC 2046 “Multipurpose Internet Mail Extensions Part Two: Media Types”) is the inclusion of Content-ID headers with each attached file.

The contained URL must begin with `cid:`, which is the identifier for the referenced attachment within the larger transmission. The `cid:` identifier must match the Content-ID header of one (and only one) part of the MIME transmission containing the document being forwarded.

The following example shows the required skeleton of a cXML document with an attached JPEG image (without the HTTP headers shown above):

```
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--

```

This skeleton is also all that a receiving MIME parser must be able to process. Applications that make use of the media type described in RFC 2387 “The MIME Multipart/Related Content-type” will get much more information if the skeleton is enhanced:

```

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--

```

Receiving MIME parsers that do not understand the multipart/related media type must treat the two examples above identically. Each part of the MIME transmission can additionally have a Content-transfer-encoding and use that encoding. This addition is not necessary for HTTP transmission. Content-description and Content-disposition headers are optional within the cXML protocol, although they provide useful documentation.

Attachment Examples

The following example shows a CatalogUploadRequest with an attached catalog:

```

POST /cXML HTTP/1.0
Content-type: multipart/related; boundary=kdf1kajfdksadjfk;
    type="text/xml"; start="<part1.PCO28.975@saturn.workchairs.com>"
First MIME body part } --kdf1kajfdksadjfk
header               } Content-type: text/xml; charset=UTF-8

```

Content-ID: <part1.PCO28.975@saturn.workchairs.com>

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/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.PCO28.975@saturn.workchairs.com</URL>
      </Attachment>
    </CatalogUploadRequest>
  </Request>
</cXML>
--kdfkajfdksadjfk
Content-type: text/plain; charset=US-ASCII
Content-Disposition: attachment; filename=PremiereCatalog.cif
Content-ID: <part2.PCO28.975@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
```

ID of MIME attachment ————— **<URL>cid:part2.PCO28.975@saturn.workchairs.com</URL>**

Second MIME body part header { **Content-type: text/plain; charset=US-ASCII**
Content-Disposition: attachment; filename=PremiereCatalog.cif
Content-ID: <part2.PCO28.975@saturn.workchairs.com>
Content-length: 364

MIME trailer ——— ENDOFDATA
 --kdfkajfdksadjfk--

Surround IDs in Content-ID or Content-Type headers with angle brackets (< >), but omit these brackets when referring to IDs in URL elements. Similarly, prepend message IDs with cid: in URL elements, but not in MIME headers.

Special characters in cid URLs must be hex encoded (in %hh format).

Use the Attachment element when attaching text files, PDFs, images, or other such documents to a cXML document. When attaching another cXML document, use cXMLAttachment, regardless of whether that cXML document contains attachments itself. The cXMLAttachment element serves to alert the receiving system that additional cXML processing might be required to handle the attachment.

The following example shows a CopyRequest forwarding a cXML document with attachments using cXMLAttachment. For more information about CopyRequest, see [“CopyRequest”](#) on page 165.

```
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/1.2.011/cXML.dtd">
<cXML version="1.0" 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>password</SharedSecret>
      </Credential>
    <UserAgent>Sender Application</UserAgent>
  </Sender>
</Header>
<Request deploymentMode="test">
  <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--

```

More Information About MIME

For more information about the MIME standard, see the following websites:

www.hunnysoft.com/mime
www.rad.com/networks/1995/mime/mime.htm
www.ietf.org/rfc/rfc1341.txt
www.ietf.org/rfc/rfc2046.txt
www.ietf.org/rfc/rfc2387.txt

For more information about attaching external files to purchase orders, see [“Attachment”](#) on page 123.

cXML Envelope

The cXML element is the root of cXML documents, and it contains all other elements. The cXML element is present in every cXML transaction. The following example shows a fully specified cXML element:

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

cXML has the following attributes:

version (deprecated)	<p>This attribute was deprecated in cXML 1.2.007; do not use it in new cXML documents.</p> <p>Specifies the version of the cXML protocol. A validating XML parser could also determine the version attribute from the referenced DTD. Because this version number also appears in the SYSTEM identifier in the cXML document, you should omit this attribute.</p>
xml:lang (optional)	<p>The locale used for all free text sent within this document. The receiver should reply or display information in the same or a similar locale. For example, a client specifying xml:lang="en-UK" in a request might receive "en" data in return. Specify the most descriptive and specific locale possible.</p>
payloadID	<p>A unique number with respect to space and time, used for logging purposes to identify documents that might have been lost or had problems. This value should not change for retry attempts.</p> <p>The recommended implementation is: datetime.process id.random number@hostname</p>
timestamp	<p>The date and time the message was sent, in ISO 8601 format. This value should not change for retry attempts.</p> <p>The format is YYYY-MM-DDThh:mm:ss-hh:mm (for example, 1997-07-16T19:20:30+01:00).</p>
signatureVersion	<p>If present, implies that the document is digitally signed, that is, that the document contains a valid <i>ds:Signature</i> element immediately following the <i>Request</i>, <i>Response</i>, or <i>Message</i> element. The only valid value for the attribute is 1.0; other values are reserved for future use. For more information, see Chapter 16, "cXML Digital Signatures."</p>

Locale Specified by xml:lang

The `xml:lang` attribute also appears with most free text elements (such as `Description` and `Comments`). While the XML specification allows the locale for an element to default to that specified for any parent element, such defaults result in inefficient queries of the document tree. cXML attempts to keep the locale identifiers together with the affected strings. The most descriptive and specific locale known should be specified in this attribute.

The `xml:lang` attributes appearing throughout the cXML protocol have no effect on formatted data such as numbers, dates, and times. As described for the timestamp attribute in the following section, for the timestamp attribute, such discrete values are formatted according to their data types. Longer strings (and referenced Web pages) not intended for machine processing might contain a locale-specific numeric or date format that matches a nearby `xml:lang` attribute.

Date, Time, and other Data Types

The timestamp attribute, and all other dates and times in cXML, must be formatted in the restricted subset of ISO 8601. This is described in the World Wide Web Consortium (W3C) Note entitled “Date and Time Formats” available at www.w3.org/TR/NOTE-datetime-970915.html.

Timestamps require a minimum of a complete date plus hours, minutes, and seconds. Fractions of a second are optional. This protocol requires times expressed in local time with a time-zone offset from UTC (Coordinated Universal Time, also known as Greenwich Mean Time). The “Z” time zone designator is not allowed.

For example, 2002-04-14T13:36:00-08:00 corresponds to April 14, 2002, 1:36 p.m., U.S. Pacific Standard Time.

Further references for the date, time, and other data type formats used by cXML are:

- Microsoft’s XML Data Types Reference, msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk30/hm/xmrefxmldatatypes.asp
- The original XML Data proposal to the World Wide Web Consortium (W3C), www.w3c.org/TR/1998/NOTE-XML-data-0105

Special Characters

In cXML, as in XML, not all characters can be typed from the keyboard, such as the registered trademark symbol (®). Others, such as `<` and `&`, have special meaning to XML. These characters must be encoded using character entities.

XML defines the following built-in character entities:

Entity	Character
<	<
>	>
&	&
"	"
'	'

For characters outside of the encoding you use, use the Unicode number of the character (decimal or hexadecimal), preceded by pound (#). For example, `®` and `®` represent a registered trademark symbol, ®.

For example,

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

could be encoded as

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

Single (') or double (") quotation marks must be escaped only within attribute values that are quoted using that delimiter. It is recommended that you use only single quotes to delimit attributes, unless the content will never contain quotes.

▼ To handle special characters in documents:

1. Use a template that only uses single quotes to delimit attributes.
2. Add values to the template by doing one of the following:
 - If the document is a PunchOutOrderMessage to be transmitted by the cxml-urlencoded hidden field, fill the values in the template using US-ASCII encoding. This encoding requires XML character entities for all characters beyond that encoding. For example, as described above, enter the registered trademark symbol, which is not available in US-ASCII, as `®`.
 - Otherwise, fill the values in the document using UTF-8 encoding. UTF-8 should be used for all documents sent by HTTP Post directly, or embedded in a cXML-base64 hidden field. UTF-8 includes all of US-ASCII.
3. XML escape attribute values and element content as you create the cXML document. Characters that must be escaped are `&`, `'`, `<` and `>`.

The following steps are required if you are transmitting the document in a PunchOutOrderMessage.

4. Pay attention to all characters that browsers interpret:
 - a. If you are using a cxml-urlencoded hidden field, convert all double quotes to ".
 - b. Further (for the cxml-urlencoded field), escape all ampersands that appear in contexts significant to HTML with &. To be safe, you can escape all ampersands. For example, escape & as &amp; and ' as &apos;. Escape the trademark symbol ® as &#174;.
 - c. Otherwise, if you are using a cxml-base64 hidden field, base64 encode the entire cXML document.
5. Embed the document in the HTML form with double quotes around the string value. For example, to send a Money element with an attribute having the value ®®' and containing the value ®®', the XML document might appear as:

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

which should be encoded as follows:

```
<!-- 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='&amp;#174;&amp;#xAE;&amp;apos;&#34;&amp;#34;
&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
PCFET0NUWVBFIE1vbWV5IFNZU1RFTSAnU3BIY2lhbENoYXJzLmR0ZCc+CjxNb
25leSBhbHRlcm5hdGVBbW91bnQ9JyYjMTc0OyYjeEFFOyZhcG9zOyImIzMOOyZxd
W90OyZhbXA7Jmx0Oz4mZ3Q7Jz4KJiMxNzQ7JiN4QUU7JyZhcG9zOyImIzMOOyZx
dW90OyZhbXA7Jmx0Oz4mZ3Q7PC9Nb25leT4K">
```

The preceding examples illustrate alternatives for encoding the cXML-urlencoded field. They avoid XML escaping a few characters, such as angle brackets, that are not special to XML in all contexts. A direct implementation of the previous steps would result in an HTML field such as:

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

or the XML document:

```
<?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>
```

Header

The Header element contains addressing and authentication information. The Header element is the same regardless of the specific Request or Response within the body of the cXML message. Applications need the requestor's identity, but not validation that the information provided for identity is correct.

The following example shows the Header element:

```
<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@ariba.com</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Ariba Network 1.1</UserAgent>
</Sender>
</Header>
```

The From and To elements are synonymous with From and To in SMTP mail messages; they are the logical source and destination of the messages. Sender is the party that opens the HTTP connection and sends the cXML document.

Sender contains the Credential element, which allows the receiving party to authenticate the sending party. This credential allows strong authentication without requiring a public-key end-to-end digital certificate infrastructure. Only a user name and password need to be issued by the receiving party to allow the sending party to perform Requests.

When the document is initially sent, Sender and From are the same. However, if the cXML document travels through e-commerce network hubs, the Sender element changes to indicate current sending party.

From

This element identifies the originator of the cXML request.

To

This element identifies the destination of the cXML request.

Sender

This element allows the receiving party to identify and authenticate the party that opened the HTTP connection. It contains a stronger authentication Credential than the ones in the From or To elements, because the receiving party must authenticate who is asking it to perform work.

UserAgent

A textual string representing the UserAgent who is conducting the cXML conversation. This should be a unique per-product string, and ideally, per-version. Analogous to UserAgent for HTTP conversations.

Credential

This element contains identification and authentication values.

Credential has the following attributes:

domain	Specifies the type of credential. This attribute allows documents to contain multiple types of credentials for multiple authentication domains. For messages sent on Ariba Supplier Network, for instance, the domain can be AribaNetworkUserId to indicate an email address, DUNS for a D-U-N-S number, or NetworkId for a preassigned ID.
type (optional)	Requests to or from a marketplace identify both the marketplace and the member company in From or To Credential elements. In this case, the credential for the marketplace uses the type attribute, which is set to the value "marketplace".

Credential contains an Identity element and optionally a SharedSecret or a CredentialMac element. The Identity element states who the Credential represents, while the optional authentication elements verify the identity of the party.

SharedSecret

The SharedSecret element is used when the Sender has a password that the requester recognizes.

Note: Do not use authentication elements in documents sent through one-way communication. One-way transport routes through users' browsers, so users would be able to see the document source, including Credential elements.

CredentialMac

The CredentialMac element is used for the Message Authentication Code (MAC) authentication method. This authentication method is used in situations where the sender must prove to the receiver that it has been authenticated by shared secret by a trusted third party. For example, a direct PunchOut request can travel directly from a buyer to a supplier without going through a network commerce hub, because it contains a MAC (generated by the network commerce hub) that allows the supplier to authenticate it.

The trusted third party computes the MAC and transfers it to the sender through the Profile transaction. The MAC is opaque to the sender (it is secure and non-reversible). To see how the MAC is transmitted from the trusted third party to the sender, see ["ProfileResponse"](#) on page 58.

The receiver computes the MAC using the same inputs as the trusted third party and compares it with the MAC received in the cXML document. If the two values match, the document is authentic.

To learn how to compute the MAC value, see “[Message Authentication Code \(MAC\)](#)” on page 313.

CredentialMac has the following attributes:

type	Identifies the data being authenticated and the method in which it is formatted for authentication. The only supported value is "FromSenderCredentials".
algorithm	Identifies for the MAC algorithm used on the data. The only supported value is "HMAC-SHA1-96".
creationDate	Specifies the date and time the MAC was generated.
expirationDate	Specifies the date and time after which this MAC is no longer valid. Receivers must reject MACs that are received after the expirationDate. Receivers can optionally reject unexpired MACs. For example, a receiver might reject MACs that are scheduled to expire in less than an hour.

The following example shows a Credential element that contains a CredentialMac element:

```
<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>Ariba Buyer 8.1</UserAgent>
  </Credential>
</Sender>
```

Multiple Credentials

The From, To, and Sender elements can each optionally contain multiple Credential elements. The purpose of supplying multiple credentials is to identify a single organization using different domains. For example, an organization might be identified by including both a DUNS number and a NetworkId number.

The receiver should validate all credentials with domains it recognizes and it should reject the document if any credentials with recognized domains do not match an organization it knows. It should also reject the document if any two credentials in the same From, To, or Sender section appear to refer to different entities.

The receiver should reject the document if there are multiple credentials in a To, From, or Sender section that use different values but use the same domain.

Request

Clients send requests for operations. Only one Request element is allowed for each cXML envelope element, which simplifies the server implementations, because no demultiplexing needs to occur when reading cXML documents. The Request element can contain virtually any type of XML data.

Typical Request elements are:

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

Request has the following attributes:

deploymentMode (optional)	Indicates whether the request is a test request or a production request. Allowed values are "production" (default) or "test".
Id	This attribute can be used to call out an element and all its children as a target for a digital signing. For more information about digital signatures, see Chapter 16, "cXML Digital Signatures."

Response

Servers send responses to inform clients of the results of operations. Because the result of some requests might not have any data, the Response element can optionally contain nothing but a Status element. A Response element can also contain any application-level data. During PunchOut for example, the application-level data is contained in a PunchOutSetupResponse element.

The typical Response elements are:

- ProfileResponse
- PunchOutSetupResponse
- GetPendingResponse

Response has the following attribute:

Id	This attribute can be used to call out an element and all its children as a target for a digital signing. For more information about digital signatures, see Chapter 16, "cXML Digital Signatures."
----	---

Status

This element conveys the success, transient failure, or permanent failure of a request operation.

Status has the following attributes:

code	The status code of the request. For example, 200 represents a successful request. See the table of codes, below.
text	The text of the status. This text aids user readability in logs, and is a canonical string for the error in English.
xml:lang (optional)	The language of the data in the Status element. Optional for compatibility with cXML 1.0. Might be required in future versions of cXML.

The attributes of the Status element indicate what happened to the request. For example:

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

The content of the Status element can be any data needed by the requestor and should describe the error. For a cXML 200/OK status code, there might be no data. However, for a cXML 500/Internal Server Error status code, or other similar code, it is strongly recommended that the actual XML parse error or application error be presented. This error allows better one-sided debugging and interoperability testing. For example:

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

The following table describes the cXML status code ranges:

Range	Meaning
2xx	Success
4xx	Permanent error. Client should not retry. The error prevents the request from being accepted.
5xx	Transient error. Typically a transport error. Client should retry. The recommended number of retries is 10, with a frequency of one hour. At a minimum a six hour retry window is recommended. For high priority requests, such as rush orders, you might want to increase the retry frequency.

Servers should not include additional Response elements (for example, a `PunchOutSetupResponse` element) unless the status code is in the cXML 200 range (for example, cXML 200/OK).

Because cXML is layered above HTTP in most cases, many errors (such as HTTP 404/Not Found) are handled by the transport. All transport errors should be treated as transient and the client should retry, as if a cXML 500 range status code had been received. All HTTP replies that don't include valid cXML content, including HTTP 404/Not found and HTTP 500/Internal Server Error status codes, are considered transport errors. Other common transport problems include timeouts, TCP errors (such as "connection refused"), and DNS errors (such as "host unknown"). Validation errors in parsing a Request document would normally result in a cXML permanent error in the 400 range, preferably 406/Not Acceptable.

The following table includes possible cXML status codes:

Status	Text	Meaning
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 202, 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 <code>PunchOutOrderMessage</code> , 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.

Status	Text	Meaning
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.

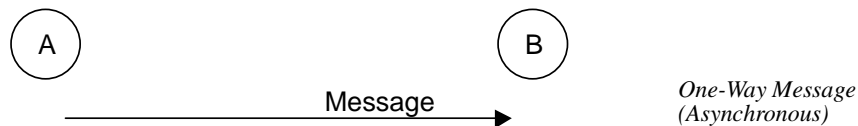
Status	Text	Meaning
500	Internal Server Error	Server was unable to complete the Request.
550	Unable to reach cXML server	Unable to reach next cXML server to complete a transaction requiring upstream connections. An intermediate hub can return this code when a supplier site is unreachable. If upstream 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.

For status codes related to catalog uploading, see [“Response”](#) on page 293.

When receiving unrecognized codes, cXML clients must handle them according to their class. Therefore, older clients should treat all new 2xx codes as 200 (success), 4xx codes as 400 (permanent failure), and 5xx codes as 500 (transient error). This behavior allows for both further expansions of the cXML protocol and server-specific codes without loss of interoperability.

One-Way (Asynchronous) Model

Unlike Request-Response transactions, One-Way messages are not restricted to the HTTP transport. One-way messages are for situations when an HTTP channel (a synchronous request-response type operation) is not appropriate. The following figure shows an example of how A and B might communicate with messages instead of the Request-Response transaction.



In this case, a possible scenario would be:

1. A formats and encodes a cXML document in a transport that B understands.
2. A sends the document using the known transport. A does not (and cannot) actively wait for a response to come back from B.

3. B receives the cXML document and decodes it out of the transport stream.
4. B processes the document.

In the One-Way model, A and B do *not* have an explicit Request-Response cycle. For example, between One-Way messages, messages from other parties might arrive and other conversations could take place.

To fully specify a one-way transaction, the transport used for the message must also be documented. For the cXML transactions that use the one-way approach, the transport and encoding are specified. A common example of a transaction that uses one-way is the `PunchOutOrderMessage`.

One-way messages have a similar structure to the Request-Response model:

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

The Header element is treated exactly as it is in the Request-Response case. The cXML element is also identical to the one described on page 36. The easiest way to tell the difference between a one-way message and a Request-Response message is the presence of a Message element (instead of a Request or Response element). The following section discusses the Message element in more detail.

The Header element in a one-way message should not contain shared secret information in the sender credential. Authentication is done using the BuyerCookie. This is different from Request-Response Header.

Message

This element carries all the body level information in a cXML message. It can contain an optional Status element, identical to that found in a Response element—it would be used in messages that are logical responses to request messages.

Message has the following attributes:

deploymentMode (optional)	Indicates whether the request is a test request or a production request. Allowed values are “production” (default) or “test”.
inReplyTo (optional)	Specifies to which Message this Message responds. The contents of the inReplyTo attribute would be the payloadID of a Message that was received earlier. This would be used to construct a two-way conversation with many messages.
Id	This attribute can be used to call out an element and all its children as a target for a digital signing. For more information about digital signatures, see Chapter 16, “cXML Digital Signatures.”

The inReplyTo attribute can also reference the payloadID of an earlier Request or Response document. When a Request-Response transaction initiates a “conversation” through multiple one-way interactions, the first message can include the payloadID of the most recent relevant Request or Response that went in the other direction. For example, a Message containing a PunchOutOrderMessage might include an inReplyTo attribute containing the payloadID of the PunchOutSetupRequest that started the PunchOut session. The BuyerCookie included in the PunchOut documents performs a similar function to that of the inReplyTo attribute.

Transport Options

There are two commonly used transports for one-way messages: HTTP and URL-Form-Encoding. These are just two of the well-defined transports today; more could become supported in the future.

HTTP

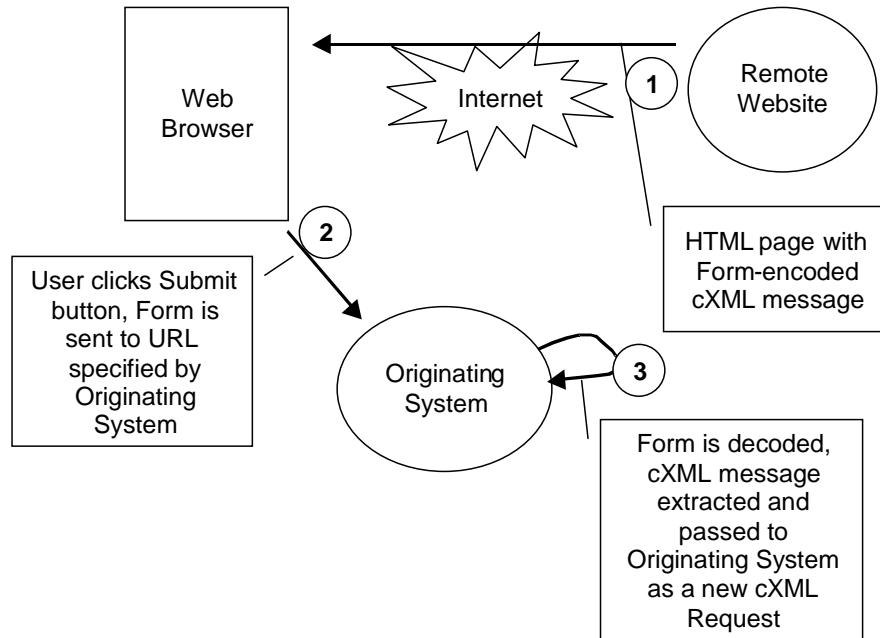
Procurement applications pull information using one-way HTTP communication. The one type of transaction that uses one-way HTTP communication is GetPendingRequest, discussed on page 297.

HTTPS is preferred, because it encrypts transmitted data for security.

URL-Form-Encoding

URL-Form-Encoding enables integration between remote websites and procurement applications. It also serves as a way to avoid requiring a listening server on the buyer’s system that is directly accessible through the Internet. This transport is best understood by examining how the PunchOutOrderMessage transaction works.

Remote websites do not directly send cXML PunchOutOrderMessage documents to procurement applications; instead, they encode them as hidden HTML Form fields and post them to the URL specified in the BrowserFormPost element of the PunchOutSetupRequest. When the user clicks a Check Out button on the website after shopping, the website sends the data to the procurement application as an HTML Form Submit. The following diagram illustrates what happens:



The semantics of packing and unpacking are described below.

Form Packing

Remote websites assign each PunchOutOrderMessage document to a hidden field on the Form named cXML-urlencoded or cXML-base64. They assign the HTML Form element a METHOD of POST and an ACTION consisting of the URL passed in the BrowserFormPost element of the PunchOutSetupRequest. For example:

```

<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>

```

Additional HTML tags on the page might contain the above fragment to describe the contents of the shopping basket in detail.

Note: When Web servers send the cXML-urlencoded field, it is not yet URL encoded. This encoding is required only when the form is submitted by Web browsers (when users click Check Out in the above example). Web browsers themselves meet this requirement. The Web server must HTML-encode only the field value, escaping quotation marks and other special characters, so the form displays properly for the user.

The names cXML-urlencoded and cXML-base64 are case insensitive.

cXML-urlencoded

The cXML-urlencoded field is URL encoded (per the HTTP specification) by the Web browser, not by the Web server or the supplier. This is because the encoding is required only when the form is submitted by a Web browser, such as when a user clicks Check Out in the previous example. However, the Web server must HTML-encode the field value, escaping quotation marks and other special characters, so that the form will display correctly.

Note: Suppliers should never URL encode the cXML-urlencoded field. This field is automatically URL-encoded by the web browser.

For cXML-urlencoded data, the receiving parser cannot assume a charset parameter beyond the default for media type text/xml. No character encoding information for the posted data is carried in an HTTP POST. The receiving Web server cannot determine the encoding of the HTML page containing the hidden field. The cXML document forwarded in this fashion must therefore use us-ascii character encoding. Any characters (including those “URI encoded” as “%XX”) found in the XML source document must be in the “us-ascii” set. Other Unicode symbols can be encoded using character entities in that source document.

cXML-Base64

The cXML-base64 hidden field supports international documents. cXML documents containing symbols outside of “us-ascii” should use this field instead of the cXML-urlencoded hidden field. This alternative has almost identical semantics, but the entire document is base64-encoded throughout transport and not HTML-encoded to the browser or URL-encoded to the receiving Web server. Base64-encoding is described in RFC 2045 “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.”

Base64-encoding from the remote website through the browser and to the receiving Web server at the client maintains the original character encoding of a cXML document. Though no charset parameter arrives with the posted information, the decoded document (after the transfer encoding is removed) can be treated as the media type application/xml. This encoding allows the receiving parser to honor any encoding attribute specified in the XML declaration. For this field (as for any application/xml documents), the default character encoding is UTF-8.

Either of these hidden fields (cXML-urlencoded or cXML-base64) must appear in the data posted to the procurement application. Though recipients should first look for cXML-base64 in the data, it is wasteful to send both fields.

Form Unpacking and Processing

The procurement application, which previously provided the appropriate URL, receives an HTML Form POST containing the Form data as described above. The Form POST processor would first look for the cXML-base64 variable, extract the value and base64-decode its contents. If that field does not exist in the data, the Form POST processor would look for the cXML-urlencoded variable, extract the URL-encoded cXML message and URL-decode it. The decoded content of the field is then processed as if it had been received through a normal HTTP Request/Response cycle.

The implied media type of the document after decoding varies, with different possible character encodings:

- The cXML-urlencoded variable is of media type text/xml with no charset attribute. It is thus restricted to the us-ascii character encoding. The receiving parser must ignore any encoding attribute in the XML declaration of the cXML document because the browser might have changed the encoding.
- The cXML-base64 variable is of media type application/xml and thus might have any character encoding (indicated by the encoding attribute of the contained XML declaration, if any). The default character encoding is UTF-8, as for any application/xml documents.

The primary difference between this transaction and a normal Request-Response transaction is that there is no response that can be generated, because there is no HTTP connection through which to send it.

Service Status Response

This transaction determines whether a particular service is currently available. When an HTTP GET is sent to a service location, the service responds with a valid, dynamically generated cXML Response document. A service can be any HTTP URL at which cXML Request documents are received.

For example, an HTTP GET sent to <https://service.ariba.com/service/transaction/cxml.asp> yields the following response:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE cXML "http://xml.cXML.org/schemas/cXML/1.2.011/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>
```

Note: This combination of transport (HTTP) and protocol (cXML) levels should be used only for the case described above.

Basic Elements

The following entities and elements are used throughout the cXML specification. Most of the definitions listed here are basic vocabulary with which the higher-order business documents are described. The common type entities and the common elements representing low-level objects are defined here.

Type Entities

Most of these definitions are from the XML-Data note submission to the World Wide Web Consortium (W3C). A few higher-level type entities that are also defined here are not from XML-Data. These types are also discussed in “[cXML Envelope](#)” on page 36.

isoLangCode

An ISO Language Code from the ISO 639 standard.

isoCountryCode

An ISO Country Code from the ISO 3166 standard.

xmlLangCode

A language code as defined by the XML 1.0 Specification (at www.w3.org/TR/1998/REC-xml-19980210.html). In the most common case, this includes an ISO 639 Language Code and (optionally) an ISO 3166 Country Code separated by a hyphen.

Unlike the full XML recommendation, IANA or private language codes should not be used in cXML. IANA and private subcodes are allowed, though they should come after a valid ISO 3166 Country Code.

The recommended cXML language code format is `xx[-YY[-zzz]*]?` where `xx` is an ISO 639 Language code, `YY` is an ISO 3166 Country Code and `zzz` is an IANA or private subcode for the language in question. Again, use of the Country Code is always recommended. By convention, the language code is lowercase and the country code is uppercase. This is not required for correct matching of the codes.

unitOfMeasure

`UnitOfMeasure` describes how the product is packaged or shipped. It must conform to UN/CEFACT Unit of Measure Common Codes. For a list of UN/CEFACT codes, see www.unetrades.net.

URL

A URL (Uniform Resource Locator) as defined by the HTTP/1.1 standard.

Base Elements

These elements, used throughout the specification, range from generic ones such as `Name` and `Extrinsic` to specific ones such as `Money`. For information about the base `TermReference` element, see page 150.

Money

The `Money` element has three possible attributes: `currency`, `alternateAmount`, `alternateCurrency`. The attributes `currency` and `alternateCurrency` must be a three-letter ISO 4217 currency code. The content of the `Money` element and of the `alternateAmount` attribute should be a numeric value. For example:

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

The optional `alternateCurrency` and `alternateAmount` attributes are used together to specify an amount in an alternate currency. These can be used to support dual-currency requirements such as the euro. For example:

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

Note: You can optionally use commas as thousands separators. Do not use commas as decimal separators.

Country

Contains the name of the country in a location. Contained by the PostalAddress element.

CountryCode

Contains the International ITU dial code for the country code. It can be entered onto a telephone keypad after the escape code to reach the country. Used by the Phone and Fax elements.

Contact

The Contact element contains information about any contact important to the current transaction. For example:

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

Chapter 3

Profile Transaction

The Profile transaction is used to retrieve cXML server capabilities, including the supported cXML version, transactions, and options on those transactions. The `ProfileRequest` and `ProfileResponse` documents must be supported by all cXML 1.1 and higher server implementations.

This chapter describes:

- [Introduction to the Profile Transaction](#)
- [ProfileRequest](#)
- [ProfileResponse](#)
- [Scenarios](#)

Introduction to the Profile Transaction

The Profile transaction enables one party to query another for cXML capabilities. On both ends, these parties include suppliers, buyers, commerce network hubs, service providers, and marketplaces. To inquire about server capabilities, send a `ProfileRequest` document. The server returns a `ProfileResponse` document containing the server information.

The Profile transaction is the only transaction that all cXML servers must support. It is intended for back-end integrations between applications, making the capabilities of cXML servers available to client systems.

The `ProfileResponse` should list all Requests supported at a particular website, not necessarily all those supported by the organization. Suppliers that can receive `OrderRequest` documents and send various messages or initiate Request/Response transactions describe their `OrderRequest` support in the profile transaction.

A ProfileRequest is generally pulled by network commerce hubs no more than once every 24 hours, and less if no request is sent to that supplier within a particular window.

The Profile transaction can also be used to simply “ping” a server within the cXML protocol.

The Profile transaction can also retrieve the locations for follow-up documents. This use replaces the Followup element used in OrderRequest documents. To obtain information about where to send any document, send a ProfileRequest document to the server.

ProfileRequest

This element has no content. It is simply routed to the appropriate cXML server using the Header. The server responds with a single ProfileResponse as described below. The only dynamic portions of this response are the payloadId and timestamp attributes of the cXML element itself. In this particular case, servers are not required to provide responses in multiple locales.

An example Request of this type is:

```
<cXML payloadID="9949494"
  xml:lang="en-US" timestamp="2000-03-12T18:39:09-08:00">
  <Header>
    Routing, identification, and authentication information.
  </Header>
  <Request>
    <ProfileRequest />
  </Request>
</cXML>
```

ProfileRequest documents should be sent to the “root” URL of a commerce network hub, which should never change. Sending a Profile Request to this root URL obtains the URL location for every other cXML Request type. The Response from a commerce network hub depends on the To element in the Profile Request.

ProfileResponse

This element contains a list of supported transactions, their locations, and any supported options. The following is a possible ProfileResponse:

```
<ProfileResponse effectiveDate="2001-03-03T12:13:14-05:00">
  <Option name="Locale">1</Option>
  ...
  <Transaction requestName="PunchOutSetupRequest">
    <URL>http://www.workchairs.com/cXML/PunchOut.asp</URL>
    <Option name="operationAllowed">create inspect</Option>
    <Option name="dynamic pricing">0</Option>
    ...
  </Transaction>
  ...
</ProfileResponse>
```

A more likely ProfileResponse from a current supplier might be:

```
<ProfileResponse effectiveDate="2000-01-01T05:24:29-08:00"
  lastRefresh="2001-09-08T05:24:29-08:00">
  <Transaction requestName="OrderRequest">
    <URL>http://workchairs.com/cgi/orders.cgi</URL>
    <Option name="service">workchairs.orders</Option>
  </Transaction>
  <Transaction requestName="PunchOutSetupRequest">
    <URL>http://workchairs.com/cgi/PunchOut.cgi</URL>
    <Option name="service">workchairs.signin</Option>
  </Transaction>
</ProfileResponse>
```

ProfileResponse has the following attributes:

effectiveDate	The date and time when these services became available. Dates should not be in the future.
lastRefresh	Indicates when the profile cache was last refreshed. When an application receives a ProfileResponse from a profile caching server, it will know the age of the data in the cache.

Option Element

The Option element contains the value for a defined option for either the overall service or a specific transaction. Option has the following attribute:

name	<p>The name of this option. This attribute should not be viewed directly (because the profile is intended for machine consumption). The client system must understand this before receiving a ProfileResponse document.</p> <p>Currently defined values for name are service, attachments, changes, and requestNames.</p>
-------------	---

MAC Options

If the ProfileResponse document is sent by a trusted third party (such as a network commerce hub) and it lists transactions that rely on MAC authentication, it should contain Option elements that list MAC authentication values. The client will insert these values in a CredentialMac element in documents it sends directly to the server.

For example:

```
<ProfileResponse>
  <Option name="CredentialMac.type">FromSenderCredentials</Option>
  <Option name="CredentialMac.algorithm">HMAC-SHA1-96</Option>
  <Option name="CredentialMac.creationDate">2003-01-17T17:39:09-08:00</Option>
  <Option name="CredentialMac.expirationDate">2003-01-17T23:39:09-08:00</Option>
  <Option name="CredentialMac.value">67mURtR6VI6YnlsK</Option>
```

If the server supports direct PunchOut, additional Option elements should appear for PunchOutSetupRequest in the ProfileResponse. For more information, see [“PunchOutSetupRequest Options”](#) on page 61.

For more information about MACs, see [“Message Authentication Code \(MAC\)”](#) on page 313.

Service

The Profile transaction can return multiple variations of a single transaction type.

If a cXML server supports multiple implementations of a particular transaction, ProfileResponse can distinguish them. For example, a marketplace might provide two services within the ProviderSetupRequest transaction: marketplace.signin and marketplace.console. The ProfileResponse must list them in a way that differentiates them:

ProfileResponse can uniquely identify a specific location for each variation of a transaction. In the case of ProviderSetupRequest, the variation is the service name. ProfileResponse uses the Option element to include the service name and value, for example:

```
<Transaction requestName="ProviderSetupRequest">
  <URL>http://service.hub.com/signin</URL>
  <Option name="service">signin</Option>
</Transaction>

<Transaction requestName="ProviderSetupRequest">
  <URL>http://service.hub.com/console</URL>
  <Option name="service">console</Option>
```

</Transaction>

If there is only one location for a particular type of transaction, then the Option element is not needed.

When looking for a particular transaction type and Option name="service" is provided, use the transaction that matches the desired service. If there is no such Option name and option value match, use the first transaction with no option name and value.

Each variation of a transaction must uniquely identify its particular location. In the case of ProviderSetupRequest, the unique identifier is "service". These unique identifiers use the Option element in the Transaction element. The Option element contains the unique identifier's name. The value for the Option element is the unique identifier's value.

PunchOutSetupRequest Options

When PunchOutSetupRequest is returned as a supported transaction, three options can be specified to indicate that direct PunchOut is supported. These options inform clients that they can send PunchOutSetupRequest documents directly to servers, without going through network commerce hubs for authentication, and which authentication methods are supported:

- To specify the URL for receiving direct PunchOutSetupRequest documents:

<Option name="Direct.URL">https://asp.workchairs.com/directPunchout</Option>

- To indicate that the server supports Message Authentication Code (MAC) authentication:

<Option name="Direct.AuthenticationMethod.CredentialMac">Yes</Option>

In addition, this option instructs the trusted third party to generate a Message Authentication Code for the server. There are additional Option elements that should appear within the ProfileResponse element for profiles sent by trusted third parties, see "[MAC Options](#)" on page 60. For information about MAC authentication, see "[Message Authentication Code \(MAC\)](#)" on page 313.

- To indicate that the server supports the digital certificate authentication method:

<Option name="Direct.AuthenticationMethod.Certificate">Yes</Option>

This option indicates that the server sends AuthRequest documents to validate PunchOut requests. For information about the Auth transaction, see "[Auth Transaction](#)" on page 318.

These options are not used for regular PunchOut.

For more information about direct PunchOut, see [“Direct PunchOut”](#) on page 110.

OrderRequest Options

When OrderRequest is returned as a supported transaction, two options must be specified: attachments and changes. The attachments option indicates whether attachments are accepted. The changes option specifies if change and delete orders are accepted. To specify acceptance of attachments:

```
<Option name = "attachments">Yes</Option>
```

To specify acceptance of change orders:

```
<Option name = "changes">Yes</Option>
```

The default for both options is No. Documents with attachments or changes set to No should be handled identically to documents that do not mention the option.

For more information about cXML document attachments, see [“Wrapping Layers”](#) on page 30.

SessionStatusRequest Options

When SessionStatusRequest is returned as a supported transaction, one option must be specified: requestNames. There is no default. This option informs the client that the server supports session checks and updates when performing any of the transactions specified in the content of the Option element. This content must be a space-separated list from the set “OrderStatusSetupRequest,” “ProviderSetupRequest” and “PunchOutSetupRequest.” Transaction elements for each of the listed requests must also be included in the ProfileResponse document.

Transaction

The description of a transaction supported by this service. The Profile definition currently indicates the locations to which to send specific requests. Future versions of cXML might add more Option definitions and extend the Profile information to include more information about supported requests.

The Transaction element must contain a URL element.

Transaction has the following attribute:

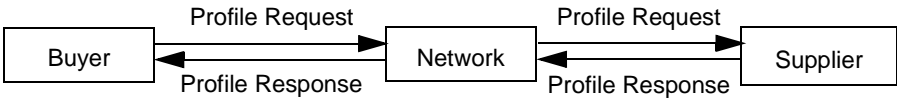
requestName	A specific request that this server accepts at the given URL. Values can be the name of any Request document defined by cXML.
--------------------	---

Scenarios

ProfileRequest documents can be sent by several possible entities to obtain server capabilities and information from suppliers, buyers, commerce network hubs, service providers, and marketplaces. The possible combinations of these parties and the kinds of transaction information that can be returned are described in the following scenarios.

From Buyer to Supplier

A Profile Request document is sent from a buyer to a supplier through a commerce network hub. The network commerce hub queries a supplier once a day, and caches the information to use in profile responses in reply to profile requests about a particular supplier.



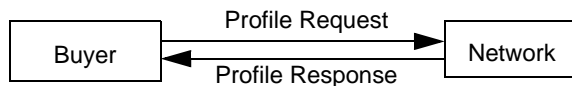
The supplier returns in the Profile Response the following possible transactions that it can support:

- OrderRequest
- PunchOutSetupRequest

The Profile Response sent to the buyer can include capabilities offered by the network on behalf of that supplier.

From Buyer to the Network

A Profile Request document is sent from a buyer to the network.



The network can return in the Profile Response the following possible transactions that it can support:

- **SupplierDataRequest**
- **SubscriptionListRequest**
- **SubscriptionContentRequest**
- **GetPendingRequest**
- **OrderStatusSetupRequest**
- **SupplierListRequest**
- **ProviderSetupRequest**
- **SessionStatusSetupRequest**

Profile Request document sample:

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

<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/cXML.dtd">  

<cXML payloadID="9949494" xml:lang="en-US"  

    timestamp="2002-02-04T18:39:09-08:00">  

    <Header>  

        <From>  

            <Credential domain="NetworkId">  

                <Identity>AN01001010101</Identity> <!-- marketplace's id -->  

            </Credential>  

        </From>  

        <To>  

            <Credential domain="NetworkId">  

                <Identity>AN01000000001</Identity> <!-- Network -->  

            </Credential>  

        </To>  

        <Sender>  

            <Credential domain="NetworkId">  

                <Identity>AN01001010101</Identity>  

                <!-- marketplace's shared secret -->  

                <SharedSecret>abracadabra</SharedSecret>
```

```

        </Credential>
        <UserAgent>Ariba Marketplace 7.5</UserAgent>
    </Sender>
</Header>
<Request>
    <ProfileRequest />
</Request>
</cXML>

```

Profile Response document sample:

```

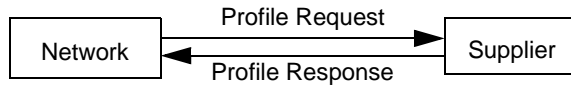
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/cXML.dtd">

<cXML payloadID="9949494" xml:lang="en-US"
    timestamp="2002-02-04T18:39:49-08:00">
    <Response>
        <Status code="200" text="OK"/>
        <ProfileResponse effectiveDate="2002-01-01T05:24:29-08:00">
            <Transaction requestName="OrderStatusSetupRequest">
                <URL>https://superduper.com/a/OrderStatusSetup</URL>
            </Transaction>
            <Transaction requestName="GetPendingRequest">
                <URL>https://superduper.com/a/GetPending</URL>
            </Transaction>
            <Transaction requestName="SubscriptionListRequest">
                <URL>https://superduper.com/b/SubscriptionList</URL>
            </Transaction>
            <Transaction requestName="SubscriptionContentRequest">
                <URL>https://superduper.com/b/SubscriptionContent</URL>
            </Transaction>
            <Transaction requestName="SupplierListRequest">
                <URL>https://superduper.com/c/SupplierList</URL>
            </Transaction>
            <Transaction requestName="SupplierDataRequest">
                <URL>https://superduper.com/c/SupplierData</URL>
            </Transaction>
            <Transaction requestName="ProviderSetupRequest">
                <URL>https://superduper.com/d/ProviderSetup</URL>
            </Transaction>
            <Transaction requestName="SessionStatusRequest">
                <URL>https://superduper.com/d/SessionStatus</URL>
                <Option name="requestNames">OrderStatusSetupRequest</Option>
            </Transaction>
        </ProfileResponse>
    </Response>
</cXML>

```

From a Network to Supplier

A Profile Request is sent from a network commerce hub to a supplier. The Network commerce hub queries a supplier once a day, and then caches the information to use in profile responses in reply to profile requests about a particular supplier.



The supplier can return in the Profile Response document the following possible transactions that it supports:

- OrderRequest
- PunchOutSetupRequest

Profile Request sample:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/cXML.dtd">

<cXML payloadID="9949494" xml:lang="en-US"
  timestamp="2002-02-04T18:39:09-08:00">
  <Header>
    <From>
      <Credential domain="NetworkId">
        <Identity>AN01001010101</Identity> <!-- Network's id -->
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkId">
        <Identity>AN01234663636</Identity> <!-- any supplier's id -->
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkId">
        <Identity>AN01001010101</Identity>
        <!-- Network's sharedsecret -->
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ariba Marketplace 7.5</UserAgent>
    </Sender>
  </Header>
  <Request>
    <ProfileRequest />
  </Request>
</cXML>

```

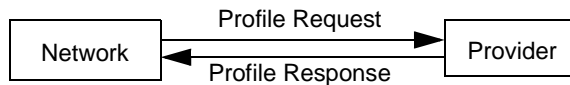
Profile Response sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/cXML.dtd">

<cXML payloadID="9949494" xml:lang="en-US"
  timestamp="2002-02-04T18:39:49-08:00">
  <Response>
    <Status code="200" text="OK"/>
    <ProfileResponse effectiveDate="2002-01-01T05:24:29-08:00">
      <Transaction requestName="PunchOutSetupRequest">
        <URL>https://www.acme.com/cxml/PunchOutSetup</URL>
      </Transaction>
      <Transaction requestName="OrderRequest">
        <URL>https:// www.acme.com/cxml /Order</URL>
        <Option name="attachments">yes</Option>
        <Option name="changes">yes</Option>
      </Transaction>
    </ProfileResponse>
  </Response>
</cXML>
```

From the Network to Service Provider

A Profile Request is sent from the network commerce hub to service provider partners. Routing service providers need to specify if one or two Profile Responses will be returned, since profile information can be returned for both the service provider and downstream supplier accounts.

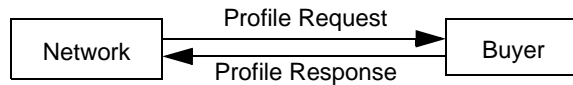


The service provider can return in the Profile Response document the following possible transactions that it supports:

- ProviderSetupRequest
- SessionStatus
- OrderRequest

From a Network to Buyer

A Profile Request is sent from a network commerce hub to a buyer. The Network commerce hub queries a buyer once a day, and then caches the information. Later, this information about the buyer is used in profile responses in reply to profile requests from providers and suppliers.

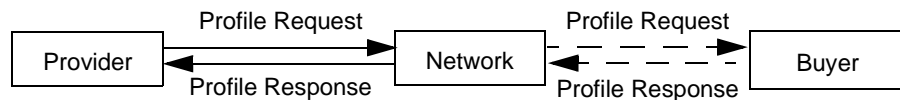


A buyer can return in the Profile Response document the following possible transactions that it supports:

- StatusUpdateRequest
- InvoiceRequest

From Service Provider to Buyer

A Profile Request is sent from a service provider to a buyer and routed through the network. This scenario is a replacement for the Followup element. The network queries a buyer once a day, and then caches the information. Later, this information about the buyer is used in profile responses in reply to profile requests from providers.



The network commerce hub can return in the Profile Response document to the service provider the following possible transactions that it supports on behalf of a buyer's account:

- StatusUpdateRequest
- InvoiceRequest

Chapter 4

PunchOut Transaction

PunchOut enables users of procurement applications to access supplier contracts for products or services that reside at the supplier's website. It eliminates the need for the suppliers to send whole catalogs to buying organizations. Instead, suppliers send just short index files that name their storefronts, product categories, or products.

This chapter shows how suppliers can modify a website to support PunchOut. It describes:

- [PunchOut Requirements](#)
- [PunchOut Event Sequence](#)
- [PunchOut Documents](#)
- [Modifications to the Supplier's Web Pages](#)
- [PunchOut Website Suggestions](#)
- [PunchOut Transaction](#)
- [Direct PunchOut](#)

PunchOut Requirements

Before buying organizations configure their procurement applications for PunchOut, or suppliers implement PunchOut websites, both parties must evaluate the benefits and requirements of PunchOut.

Buying Organizations

Setup and testing of cXML-compatible procurement applications with a PunchOut-enabled supplier can be completed in less than one day.

Therefore, PunchOut is a good solution for buying organizations of all sizes and levels of technical expertise. The decision to use PunchOut should be based on the business practices and types of commodities purchased. (See [“Content Delivery Strategy”](#) on page 21 for a list of commodities that are well suited for PunchOut.)

Business Issues

Buying organizations should consider the following questions when deciding whether to use static catalog content such as an Index or Contract documents, or PunchOut:

- Do requisitioners and approvers have Internet access? If not, would controlled access to the Internet be allowed?
- Does the buying organization want their suppliers to create and maintain catalog content (including pricing)?
- Do requisitioners currently procure goods on the Internet? If so, do these goods require a supplier-side configuration tool or contain unique attributes that cannot conform to a static content model?
- Does the buying organization use content aggregators for catalogs (for example, Aspect, TPN Register, or Harbinger)?
- Does the buying organization currently procure services (for example, consultants, temp services, or maintenance) through the Internet?
- Does the buying organization currently conduct online sourcing?

If the answer to any of the above questions is yes, PunchOut might be appropriate for the buying organization.

Technical Issues

Buying organizations must meet the following technical requirements:

- **Direct Internet Access**—Users within buying organizations must have direct Internet access. PunchOut relies on regular Web browser sessions where the user interacts with live supplier websites. This communication occurs through regular intranet/Internet infrastructure, not through the procurement application.
- **Reliable Internet Connection**—Internet access must be constantly operational and reliable. If users cannot procure products because of Internet outages, they are likely to make rogue purchases.
- **Contracts with PunchOut Suppliers**—Purchasing agents must have established contracts with PunchOut-enabled suppliers. PunchOut websites allow access only to known, authenticated buying organizations.

Suppliers

The term *supplier* in the context of PunchOut encompasses more than the traditional definition of the term. The PunchOut protocol was designed as a flexible framework capable of transmitting data about virtually any kind of product or service from any kind of supplier, distributor, aggregator, or manufacturer.

Example products and services include:

- Computers direct from a manufacturer or reseller
- Chemicals and reagents from an aggregator
- Office supplies from a distributor
- Contract services from a temp agency

The supplier might already have a transactive website capable of hosting content and receiving purchase orders. Given this capability, the supplier needs to consider both the supplier's business practices and technical resources in deciding whether to implement PunchOut.

Business Issues

Suppliers should consider the following questions:

- Does the supplier currently sell the supplier's products or services through the Internet? If so, do they offer customer-specific content (contract pricing) through their website?
- Does the supplier's products and services fall into one of the PunchOut categories as described in the chart in "[Content Delivery Strategy](#)" on page 21? To review, these categories include:

- Highly configurable products (such as computers)
 - Large number of line items (such as books)
 - Unique product attributes (such as chemicals)
 - Normalized data (such as MRO Supplies)
 - Rapidly changing or expanding items (such as temporary services or books)

- Does the supplier prefer to receive purchase orders and/or payment through their website?

If the answer to any of the above questions is yes, PunchOut might be appropriate for the supplier's organization.

Technical Issues

Suppliers must meet the following technical requirements:

- **Reliable Internet Connection**—The Web server infrastructure and Internet connection must be extremely reliable. If users cannot access remote content, they are likely to go to another supplier.
- **Competent website Administrators**—The PunchOut website and supporting applications will require periodic maintenance and modification. Users' needs and the supplier's product offerings will change, so the supplier needs personnel to modify the supplier's PunchOut infrastructure.
- **Support for Basic Transactions**—PunchOut websites do not need to support all cXML functionality, but they must support the following required transactions:

Profile Transaction
PunchOutSetupRequest
PunchOutSetupResponse
PunchOutOrderMessage

Work Estimate

The following table lists estimates of work required for cXML PunchOut integration based on estimates from suppliers:

Level of Pre-existing Infrastructure	Estimated Time for Completion
cXML enabled and integrated with network commerce hub	1-2 weeks with in-house IT staff 2-3 weeks with contractors
Transactive site with XML infrastructure	3 weeks with in-house IT staff 3-4 weeks with contractors
Transactive site without XML infrastructure	4 weeks with in-house IT staff 4-5 weeks with contractors

Understanding XML

The first step to becoming PunchOut enabled is to understand XML. For an explanation of XML, see [“cXML, an XML Implementation”](#) on page 15. To implement a PunchOut website, the supplier must have a fundamental understanding of how to create, parse, query, receive, and transmit XML data to and from a remote source.

The basic tools to process XML documents are XML parsers. These parsers are freely available from Microsoft and other companies (for example, an XML parser is standard in Microsoft Internet Explorer 5). For a list of XML tools, see “XML Utilities” on page 24.

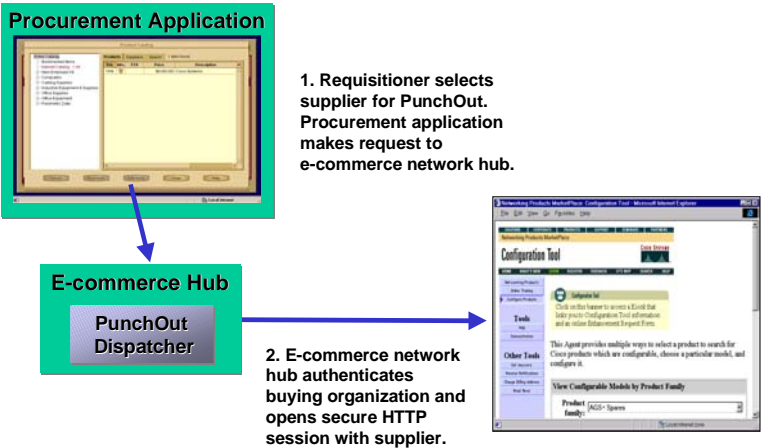
PunchOut Event Sequence

A PunchOut session is composed of several distinct steps.

Steps 1 & 2: PunchOut Request

Users log in to a procurement application and open new purchase requisitions. They find desired items by searching their local catalogs by commodity, supplier, or product description. When they select a PunchOut item, the procurement application opens a new browser window and logs them into their accounts at the supplier’s website.

The following figure illustrates the PunchOut request steps:



How does it work? When a user clicks a PunchOut item, the procurement application sends a cXML PunchOutSetupRequest document to a network e-commerce hub. Acting as the trusted third party, the hub accepts the request, verifies the buying organization, and passes the request to the supplier’s PunchOut website.

Note: All cXML documents sent through the Internet can travel through SSL (Secure Socket Layer) 3.0-encrypted HTTPS connections.

The purpose of this request is to notify the supplier's website of the buyer's identity, and to communicate the operation to be performed. Supported operations include the following:

- **create** – Initiates a new PunchOut session
- **edit** – Re-opens a PunchOut session for editing
- **inspect** – Re-opens a PunchOut session for inspection (no changes can be made to the data)
- **source** – Initiates a PunchOut session for a RFQ (Request for Quote) create/edit session in a sourcing application

After the supplier's website receives a request, it sends back a PunchOutSetupResponse containing a URL that tells the procurement application where to go to initiate a browsing session on the supplier's website.

The procurement application opens a new browser window, which displays a session logged into an account on the supplier's website. This account can be specific to a region, a company, a department, or a user.

Direct PunchOut is an alternative method for initiating PunchOut sessions, where the PunchOut site, not a network commerce hub, authenticates the PunchOut request. For more information, see [“Direct PunchOut”](#) on page 110.

Step 3: Product Selection

Users select items from the supplier's inventory using all the features and services provided by the supplier's website:



3. Requisitioner uses supplier site to find and configure products.

Depending on the product or customer, these features might include the following:

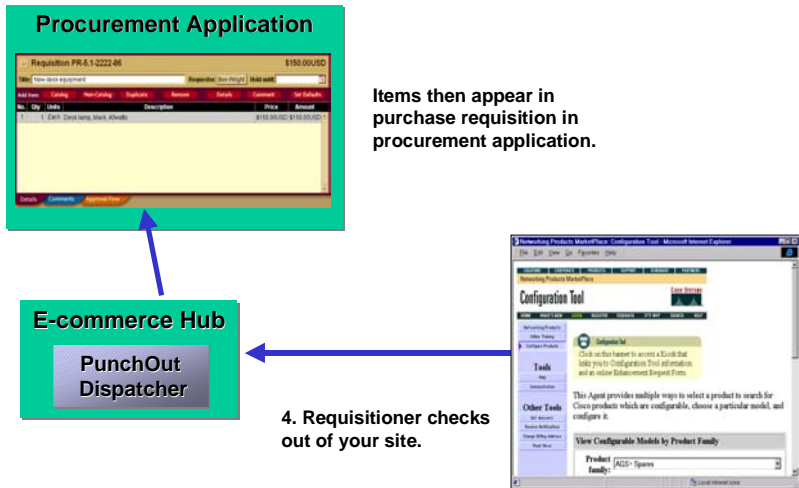
- Configurator tools for building customized products (for example, computers, organic compounds, or personalized products)
- Search engines for finding desired products from large catalogs.
- Views of normalized data for comparing products based on price, features, or availability (for example, MRO products)
- Views of attributes unique to a particular commodity (for example, printed materials, chemical and reagents, or services)
- Real-time pricing, inventory, and availability checking
- Automatic tax and freight calculations based on ship-to destination, size, or quantity of items (not necessary to calculate during the PunchOut session)

How does it work? After the procurement application directs users to the supplier’s website, the shopping experience is the same as if they had logged on to the supplier’s website directly. Thus, none of the previously listed features and services require modification.

Step 4: Check Out

The supplier’s website calculates the total cost of the user’s selections, including tax, freight, and customer-specific discounts. Users then click the supplier’s website’s “Check Out” button to send the contents of the shopping cart to the their purchase requisitions within the procurement application.

The following figure illustrates the check-out steps:



How does it work? When users click the supplier's "Check Out" button, they submit an HTML form back to their procurement application. One form field consists of a cXML PunchOutOrderMessage containing product details and prices. The supplier can also send hidden supplier cookies, which can later associate items with a specific shopping session.

Effectively, the supplier has provided a quote for the requested items—the supplier has not yet received a purchase order, so the supplier cannot yet book the order.

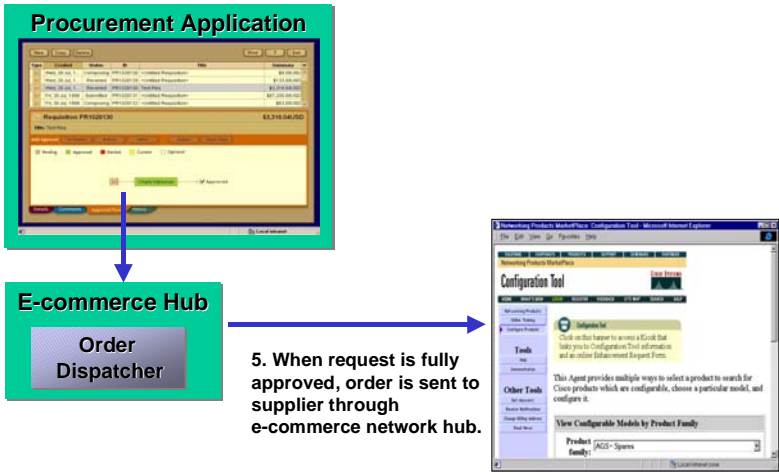
If users, including approvers, later need to edit any of the items in a purchase requisition, the supplier can allow them to "re-PunchOut" to the supplier's website. The procurement application sends back the contents of the original shopping cart to the supplier's website, and users make any changes there. Upon check out, the supplier's website returns the items to the purchase requisition.

The supplier's website is the information source for all PunchOut items. Changes to the quantity or the addition of new items to the requisition might alter tax or shipping charges, which would require recalculation at the supplier's website. Thus, any changes to the original items need to be made at the supplier's website, not in the procurement application, therefore the need to re-PunchOut. A re-PunchOut is simply a PunchOutSetupRequest with "edit" as its operation.

Step 5: Transmittal of Purchase Order

After the contents of the shopping cart have been passed from the supplier's website to the user's purchase requisition, the procurement application approval processes take over. When the purchase requisition is approved, the procurement application converts it into a purchase order and sends it back to the supplier's website for fulfillment. Purchasing card data can be transmitted along with the order, or the supplier can invoice the order separately.

The following figure illustrates purchase order transmittal:



How does it work? The procurement application sends all purchase orders to the e-commerce hub in cXML format. The hub then routes them to the supplier, using the supplier’s preferred order-routing method. When the supplier acknowledges the receipt of a purchase order, the supplier has effectively booked the order.

For PunchOut-enabled suppliers, the best order routing method is cXML for the following reasons:

- cXML purchase orders allow embedded supplier cookie information to be transmitted back to the supplier. Because the supplier cookie is of data type “any”, it does not easily map to other order routing methods such as fax, e-mail, or EDI.
- PunchOut-enabled suppliers are cXML-aware, so accepting cXML purchase orders is a small incremental effort.

Purchase orders are discussed in detail in Chapter 5, “Purchase Orders.”

PunchOut Documents

There are four types of cXML documents:

- [PunchOut Index Catalog](#)
- [PunchOutSetupRequest](#)
- [PunchOutSetupResponse](#)
- [PunchOutOrderMessage](#)

All but the PunchOut Index Catalog are considered PunchOut *session* documents because they are used to transmit data between a supplier’s PunchOut site and the buyer during a PunchOut session.

PunchOut Index Catalog

PunchOut index catalogs are files that list PunchOut items and point to the supplier’s PunchOut website.

The following example shows a PunchOut index catalog:

Type of cXML document
and URL of DTD

The supplier’s identifier
for the PunchOut item

URL of the PunchOut
website (launch page) if
not configured elsewhere

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Index SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
<Index>
  <SupplierID domain="DUNS">83528721</SupplierID>
  <IndexItem>
    <IndexItemPunchout>
      <ItemID>
        <SupplierPartID>5555</SupplierPartID>
      </ItemID>
      <PunchoutDetail>
        <Description xml:lang="en-US">Desk Chairs</Description>
        <Description xml:lang="fr-FR">Chaises de Bureau</Description>
        <URL>http://www.workchairs.com/punchout.asp</URL>
        <Classification domain="UNSPSC">5136030000</Classification>
      </PunchoutDetail>
    </IndexItemPunchout>
  </IndexItem>
</Index>
```

SupplierID identifies the supplier organization. The supplier can use any identification domain, but the recommended ones are D-U-N-S (Dun & Bradstreet Universal Naming System) and NetworkId. For more information about D-U-N-S numbers, see www.dnb.com.

Description specifies the text that the procurement application displays in product catalogs. The supplier can provide the description in multiple languages, and the procurement application displays the appropriate one for the user's locale.

Classification specifies the commodity grouping of the line item to the buyer. All the supplier's products and services must be mapped and standardized to the UNSPSC schema. For PunchOut index catalogs, the Classification determines the location of the PunchOut item within catalogs displayed to users. For a list of UNSPSC codes, see www.unspsc.org.

Creating and Publishing Index Catalogs

Create these catalogs and publish them on an e-commerce hub to the supplier's customers. The catalog manager within buying organizations downloads them and stores them for use with procurement applications.

Users see the contents of the supplier's PunchOut index catalogs alongside regular, static catalog items.

PunchOut Item Granularity

The supplier can create store-level, aisle-level, or product-level catalogs.

- Store-level catalogs list one PunchOut item for all of the supplier's products and services. Users must search the supplier site to find the desired item.
- Aisle-level catalogs list multiple PunchOut items corresponding to related products and services.
- Product-level catalogs list only one product or service. Users do not need to search.

To determine how broad to make PunchOut items, consider the supplier's business model, the makeup of the supplier's product and service offerings, and the structure of the supplier's PunchOut website.

The more search and configuration tools the supplier has on the supplier's website, the more broad they can make the PunchOut items in the supplier's index catalogs.

PunchOutSetupRequest

To initiate a PunchOut session, the user selects the supplier's PunchOut item. The procurement application generates a PunchOutSetupRequest document and sends it to an e-commerce hub, which forwards it to the supplier's PunchOut website.

Following is a sample PunchOutSetupRequest document:

	<?xml version="1.0"?>
	<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
	<cXML xml:lang="en-US" payloadID="933694607118.1869318421@jlee"
	timestamp="2002-08-15T08:36:47-07:00">
	<Header>
Originator (buying organization)	<From>
	<Credential domain="DUNS">
	<Identity>65652314</Identity>
	</Credential>
	</From>
Destination (supplier)	<To>
	<Credential domain="DUNS">
	<Identity>83528721</Identity>
	</Credential>
	</To>
Previous relaying entity (Ariba SN in this case)	<Sender>
	<Credential domain="NetworkId">
	<Identity>AN200001</Identity>
	<SharedSecret>abracadabra</SharedSecret>
	</Credential>
	<UserAgent>Procurement System 2.0</UserAgent>
	</Sender>
	</Header>
	<Request>
Type of request	<PunchOutSetupRequest operation="create">
	<BuyerCookie>1CX3L4843PPZO</BuyerCookie>
	<Extrinsic name="UserEmail">jsmith</Extrinsic>
	<Extrinsic name="UniqueName">John_Smith</Extrinsic>
	<Extrinsic name="CostCenter">610</Extrinsic>
Destination for final PunchOutOrderMessage	<BrowserFormPost>
	<URL>https://bigbuyer.com:2600/punchout?client=NAwI4Jo</URL>
	</BrowserFormPost>
	<SupplierSetup>
	<URL>http://www.workchairs.com/punchout.asp</URL>
	</SupplierSetup>
	<ShipTo>
	<Address addressID="1000467">
	<Name xml:lang="en">1000467</Name>
	<PostalAddress>
	<DeliverTo>John Smith</DeliverTo>
	<Street>123 Main Street</Street>
	<City>Sunnyvale</City>
	<State>CA</State>
	<PostalCode>94089</PostalCode>
	<Country isoCountryCode="US">United States</Country>
	</PostalAddress>
	</Address>
	</ShipTo>

Item selected by user

```

<SelectedItem>
  <ItemID>
    <SupplierPartID>5555</SupplierPartID>
  </ItemID>
</SelectedItem>
</PunchOutSetupRequest>
</Request>
</cXML>

```

The payloadID and timestamp attributes near the beginning are used by cXML clients to track documents and to detect duplicate documents.

The From, To, and Sender elements allow receiving systems to identify and authorize parties. The From and To elements in a document do not change. However, as the document travels to its destination, intermediate nodes (such as Ariba Supplier Network) change the Sender element.

Network commerce hubs can change credential domains in the From and To elements, if that change results in more reliable identification. So for example, the From credential domain might change from CustomDomain to DUNS.

Create, Edit, Inspect, and Source Operations

The operation attribute specifies the type of session the buyer initiates. It can create, edit, inspect, or source.

- create sessions generate new shopping carts, which correspond to new purchase requisitions.
- edit sessions reopen previously created shopping carts or RFQs for modification. The procurement application sends line-item data as part of the PunchOutSetupRequest. The PunchOut website can use this data to re-instantiate the shopping cart created during the original session.
- inspect sessions reopen previously created shopping carts or RFQs for viewing only. As with the edit operation, the procurement application sends line-item data as part of the PunchOutSetupRequest. However, after re-instantiating the shopping cart, the PunchOut website does not allow modification of its contents.
- source sessions generate a RFQ for a sourcing application.

The following example lists an edit request:

```

<?xml version="1.0"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML xml:lang="en-US" payloadID="933695135608.677295401@jlee"
timestamp="2002-08-15T08:45:35-07:00">

```

```

<Header>
  <From>
    <Credential domain="DUNS">
      <Identity>65652314</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="DUNS">
      <Identity>83528721</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkId">
      <Identity>AN200001</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>
    <UserAgent>Procure 2.1</UserAgent>
  </Sender>
</Header>
<Request>
  <PunchOutSetupRequest operation="edit">
    <BuyerCookie>1CX3L4843PPZO</BuyerCookie>
    <Extrinsic name="UserEmail">jsmith</Extrinsic>
    <Extrinsic name="UniqueName">John_Smith</Extrinsic>
    <Extrinsic name="CostCenter">610</Extrinsic>
    <BrowserFormPost>
      <URL>https://bigbuyer.com:2600/punchout?client=NAwliuo</URL>
    </BrowserFormPost>
    <SupplierSetup>
      <URL>http://www.workchairs.com/punchout.asp</URL>
    </SupplierSetup>
    <ItemOut quantity="2">
      <ItemID>
        <SupplierPartID>220-6338</SupplierPartID>
        <SupplierPartAuxiliaryID>E000028901
        </SupplierPartAuxiliaryID>
      </ItemID>
    </ItemOut>
  </PunchOutSetupRequest>
</Request>
</cXML>

```

If the user initiated the edit session by selecting a catalog item, the PunchOutSetupRequest would contain a SelectedItem element, like a create session.

Authentication by an E-commerce Hub

PunchOutSetupRequest documents route through an e-commerce hub for authentication and to look up the URL of the supplier's PunchOut website. The steps are:

1. The hub receives the `PunchOutSetupRequest` document from the user.
2. The hub verifies the buyer's ID (From and Shared Secret) with that buyer's e-commerce account. It also identifies the requested supplier (To).
3. The hub looks up the supplier's shared secret from the supplier's account and inserts it (Shared Secret) into the Sender element.
4. The hub finds the URL of the supplier's PunchOut website in the supplier's account and sends the `PunchOutSetupRequest` document to it.
5. The supplier's website receives the cXML document and knows that it is authenticated because it contains the supplier's own shared secret.
6. The supplier's website uses information in the From element to identify the requester at the company level (for example, acme.com).
7. The supplier can use the Contact and extrinsic data in the body of the request to uniquely identify the user (for example, John Smith in Finance at acme.com).

The `PunchOutSetupRequest` and `PunchOutSetupResponse` documents pass through the e-commerce hub for authentication. The `PunchOutOrderMessage` document (returning the contents of the shopping basket to the procurement application) travels directly between the supplier's website and the procurement application through standard HTML Form submission.

Direct PunchOut is an alternative method for initiating PunchOut sessions, where the PunchOut site, not a network commerce hub, authenticates the PunchOut request. For more information, see [“Direct PunchOut”](#) on page 110.

Supplier Setup URL and SelectedItem

In previous cXML releases, the `SupplierSetup` element provided the only way to specify the URL of the supplier's PunchOut website. Beginning with cXML 1.1, the e-commerce hub already knows the URL of the supplier's PunchOut website.

Also, starting with cXML 1.1, procurement applications can use the `SelectedItem` element to specify store-, aisle-, or product-level PunchOut.

The `SupplierSetup` element has been deprecated. However, the supplier's PunchOut website must handle both methods until all PunchOut websites and procurement applications recognize and send the `SelectedItem` element.

Contact Data and Extrinsic Data for User Identification

The PunchOutSetupRequest document can contain detailed user information in the Contact element that the supplier's website can use to authenticate and direct users, such as:

- User name and role
- E-mail address

In addition, the PunchOutSetupRequest might also contain *extrinsic* data, data that the supplier can use to further identify users, such as:

- User cost center and subaccount
- Region
- Supervisor
- Default currency

Buying organizations configure their procurement applications to insert contact and extrinsic data. Ask the supplier's customers what data the supplier can expect to receive.

PunchOutSetupResponse

After receiving a PunchOutSetupRequest, the supplier's website sends a PunchOutSetupResponse. The PunchOutSetupResponse document serves two functions:

- It indicates that the PunchOutSetupRequest was successful.
- It provides the procurement application with a redirect URL to the supplier's Start Page.

It contains a URL element that specifies the Start Page URL to pass to the user's Web browser for the interactive browsing session. This URL must contain enough state information to bind to a session context on the supplier's website, such as the identity of the requester and the contents of the BuyerCookie element. Due to URL length restrictions in many applications, this URL should refer to the state information rather than including it all.

The following example lists a PunchOutSetupResponse document:

```
<?xml version="1.0"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
```

```

<cXML xml:lang="en-US" payloadID="933694607739" timestamp="2002-08-15T08:46:00-07:00">
  <Response>
    <Status code="200" text="success"></Status>
    <PunchOutSetupResponse>
      <StartPage>
        <URL>
          http://xml.workchairs.com/retrieve?reqUrl=20626;Initial=TRUE
        </URL>
      </StartPage>
    </PunchOutSetupResponse>
  </Response>
</cXML>

```

PunchOutOrderMessage

After the user selects items on the supplier's website, configures them, and clicks the supplier's "Check Out" button, the supplier's website sends a PunchOutOrderMessage document to communicate the contents of the shopping basket to the buyer's procurement application. This document can contain much more data than the other documents, because it needs to be able to fully express the contents of any conceivable shopping basket. This document does not strictly follow the Request/Response paradigm; its use will be explained in detail.

The following example lists a PunchOutOrderMessage:

```

<?xml version="1.0"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML xml:lang="en-US" payloadID="933695160894" timestamp="2002-08-15T08:47:00-07:00">
  <Header>
    <From>
      <Credential domain="DUNS">
        <Identity>83528721</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="DUNS">
        <Identity>65652314</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="workchairs.com">
        <Identity>website 1</Identity>
      </Credential>
      <UserAgent>Workchairs cXML Application</UserAgent>
    </Sender>
  </Header>

```

```

<Message>
  <PunchOutOrderMessage>
    <BuyerCookie>1CX3L4843PPZO</BuyerCookie>
    <PunchOutOrderMessageHeader operationAllowed="edit">
      <Total>
        <Money currency="USD">763.20</Money>
      </Total>
    </PunchOutOrderMessageHeader>
    <ItemIn quantity="3">
      <ItemID>
        <SupplierPartID>5555</SupplierPartID>
        <SupplierPartAuxiliaryID>E000028901
        </SupplierPartAuxiliaryID>
      </ItemID>
      <ItemDetail>
        <UnitPrice>
          <Money currency="USD">763.20</Money>
        </UnitPrice>
        <Description xml:lang="en">
          <ShortName>Excelsior Desk Chair</ShortName>
          Leather Reclining Desk Chair with Padded Arms
        </Description>
        <UnitOfMeasure>EA</UnitOfMeasure>
        <Classification domain="UNSPSC">5136030000
        </Classification>
        <LeadTime>12</LeadTime>
      </ItemDetail>
    </ItemIn>
  </PunchOutOrderMessage>
</Message>
</cXML>

```

BuyerCookie enables the procurement application to associate a given PunchOutOrderMessage with its originating PunchOutSetupRequest. Therefore, the supplier's website should return this element whenever it appears. Do not use the BuyerCookie to track PunchOut sessions, because it changes for every session, from create, to inspect, to edit.

SupplierPartAuxiliaryID acts as a supplier cookie. This field allows the supplier to transmit additional data, such as quote number or another cXML document. The procurement application passes it back to the supplier in any subsequent PunchOut edit or inspect sessions, and in the resulting cXML purchase order. The supplier can use the supplier cookie to associate items in a purchase requisition with the corresponding items in a shopping cart at the supplier's website.

Note: Procurement applications might use SupplierPartAuxiliaryID as part of the unique identifier for items, so PunchOut sites should not change this value during PunchOut edit or inspect sessions.

UnitOfMeasure describes how the product is packaged or shipped. It must conform to UN/CEFACT Unit of Measure Common Codes. For a list of UN/CEFACT codes, see www.unetrades.net/.

Classification lists the UNSPSC (Universal Standard Products and Services Classification) commodity code for each selected item. These codes are used by back-end systems within buyer and supplier organizations for accounting and report generation. For the list of UNSPSC codes, see www.unspsc.org.

Modifications to the Supplier's Web Pages

To receive or send the three cXML PunchOut session documents, PunchOutSetupRequest, PunchOutSetupResponse, and PunchOutOrderMessage, the supplier might need to modify or create four pages on the supplier's website:

- [Launch Page](#)
- [Start Page](#)
- [Sender Page](#)
- [Order Receiver Page](#)

To illustrate how the supplier might implement these pages, this chapter uses simple Active Server Page (ASP) code samples and the Microsoft Internet Explorer 5 XML Parser. Actual implementation of these pages will vary depending on the supplier development environment (for example, CGI, JavaScript, or WebObjects).

Launch Page

The Launch Page receives all authenticated PunchOutSetupRequest documents from the e-commerce hub. It reads the HTTP stream sent from the hub and validates the cXML request embedded within that stream against the cXML DTD (in the case of ASP, using method calls to the Internet Explorer 5 XML parser). After validation, the supplier's Launch Page extracts elements from the document in order to:

1. Identify the user and determine where to redirect that user.
2. Compose a PunchOutSetupResponse document and return it to the sender.

The supplier's Launch Page should store the following data for use by the supplier's Start Page:

- Identity of the requester (Sender)
- Identity of the language of the user (xml:lang) so the supplier can provide localized content
- Type of the request (create, edit, or inspect)
- Any extrinsic data that further identifies the user and the user location

Following is a sample Launch Page. This code does not use an XML tool to dynamically generate the PunchOutSetupResponse, but instead uses a static XML template into which line item data is filled. **This code is intended for illustrative purposes only.**

```
<script language=JScript RUNAT=Server>
function elementValue(xml, elem)
{
    var begidx;
    var endidx;
    var retStr;

    begidx = xml.indexOf(elem);
    if (begidx > 0) {
        endidx = xml.indexOf('</',begidx);
        if (endidx > 0)
            retStr = xml.slice(begidx+elem.length,
                               endidx);
        return retStr;
    }
    return null;
}

function twoChar( str )
{
    var retStr;
    str = str.toString();
    if ( 1 == str.length ) {
        retStr = "0" + str;
    } else {
        retStr = str;
    }
    return retStr;
}

function timestamp( dt )
{
    var str;
    var milli;
```

```

str = dt.getFullYear() + "-" + twoChar( 1 + dt.getMonth() ) + "-";
str += twoChar( dt.getDate() ) + "T" + twoChar( dt.getHours() ) + ":";
str += twoChar( dt.getMinutes() ) + ":" + twoChar( dt.getSeconds() ) + ".";
milli = dt.getMilliseconds();
milli = milli.toString();
if ( 3 == milli.length ) {
    str += milli;
} else {
    str += "0" + twoChar( milli );
}
str += "-08:00";
return str;
}

function genProlog( cXMLvers, randStr )
{
    var dt;
    var str;
    var vers, sysID;
    var nowNum, timeStr;
    vers = "1.2.011";
    sysID = "http://xml.cXML.org/schemas/cXML/" + vers + "/cXML.dtd";
    dt = new Date();
    nowNum = dt.getTime();
    timeStr = timestamp( dt );
    str = '<?xml encoding="UTF-8"?>\n';
    str += '<!DOCTYPE cXML SYSTEM "' + sysID + '">\n';
    str += '<cXML payloadID="' + nowNum + '"';
    str += randStr + '@' + Request.ServerVariables("LOCAL_ADDR");
    str += '" timestamp="' + timeStr + '">';
    return str;
}
</script>
<%
REM Create data needed in prolog.
Randomize
randStr = Int( 100000001 * Rnd )
prologStr = genProlog( "1.0", randStr )
Response.ContentType = "text/xml"
Response.Charset = "UTF-8"
%>
<%
REM This receives the PunchOutSetup request coming from the e-commerce hub.
REM It takes the ORMSURL and buyercookie, attaches them to the Start Page URL,
REM and sends the response back to the requester.
REM punchoutredirect.asp?bc=2133hfefe&url="http://workchairs/com/..&redirect="
Dim ret
Dim punch
Dim statusText
Dim statusCode

```

```

Dim cookie
Dim url
Dim xmlstr
Dim fromUser
Dim toUser
cookie = ""
url = ""
xmlstr = ""
dir = ""
path = Request.ServerVariables("PATH_INFO")
dir = Left(path, InstrRev(path, "/"))
if IsEmpty(dir) then
    dir = "/"
end if

```

REM This command reads the incoming HTTP cXML request

```

xml = Request.BinaryRead(Request.TotalBytes)
for i = 1 to Request.TotalBytes
    xmlstr = xmlstr + String(1,AscB(MidB(xml, i, 1)))
Next
cookie = elementValue(xmlstr, "<BuyerCookie>")
url = elementValue(xmlstr, "<URL>")
fromUser = elementValue(xmlstr, "<Identity>")
newXMLStr = Right(xmlstr, Len(xmlstr) - (InStr(xmlstr, "<Identity>") + Len("<Identity>")))
toUser = elementValue(newXMLStr, "<Identity>")
%>

```

REM This formats the cXML PunchOutSetupReponse

```

<% if IsEmpty(cookie) then %>
<%= prologStr %>
<Response>
    <Status code="400" Text="Bad Request">Invalid Document. Unable to extract
    BuyerCookie.</Status>
</Response>
</cXML>
<% else %>
<%= prologStr %>
<Response>
    <Status code="200" text="OK"/>
    <PunchOutSetupResponse>
        <StartPage>
            <URL>http://<%= Request.ServerVariables("LOCAL_ADDR")%>/<%=
dir%>/punchoutredirect.asp?bc=<%= cookie%>&amp;url=<%= url%>&amp;from=<%=
fromUser%>&amp;to=<%= toUser%>&amp;redirect=<%= StartPage%></URL>
        </StartPage>
    </PunchOutSetupResponse>
</Response>
</cXML>
<%end if%>

```

The supplier's Launch Page should return a StartPage URL that is unique for that PunchOut session. In addition, this URL should be valid for only a limited amount of time. By deactivating this URL, the supplier makes it more difficult for unauthorized users to access the supplier's Start Page.

Remember to implement functionality for subsequent edit and inspect sessions. Users cannot change order details for PunchOut items (such as quantity) within their procurement application. They must re-PunchOut with an edit session. For the greatest benefit to users, inspect sessions that occur after the supplier receives the order should display order status.

Start Page

The supplier's Start Page logs the requester into an account on the supplier's website. From the supplier's Start Page, users begin their shopping experience. This page might already exist at the supplier's website, so modify it to query user name and password information from the PunchOutSetupRequest document.

Allow only authorized users into the supplier's Start Page. If the supplier waits until the check-out step to authenticate them, their confidential pricing or terms are not protected.

If the supplier uses HTTP browser cookies to track user preferences and sessions, they should be destroyed after the PunchOutOrderMessage is sent to buyers. Destroying these cookies prevents the possibility of offering privileged features to unauthorized users.

Sender Page

The Sender Page sends the contents of the user's shopping cart to the user. As described earlier, after users fill their shopping carts, they click the supplier's "Check Out" button.

Below is a simple ASP implementation of this feature. This code does not use an XML tool to dynamically generate the PunchOutOrderMessage, but instead uses a static XML template into which line item data is filled. **This code is intended for illustrative purposes only.**

This is a portion of a supplier's website product page:

```
<!--#include file="punchoutitem.inc"-->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```

<!-- saved from
url=(0093)https://secure1.shore.net/wbird/cgi/vsc.cgi/wbird/houses/urban.htm?L+wbird+w
adt4101+928011405 -->

<TABLE border=0>
  <TBODY>
    <TR>
      <TD><IMG src="UrbanHouses_files/uhjm.gif"> </TD>
      <TD><STRONG>Jefferson Memorial</STRONG>- A birdfeeder with a rotunda!
      This famous American monument will be a unique addition to any garden or yard. It attracts
      small to medium sized birds and its dimensions are 11" x 9 1/2" x 8" H.
    </TD>
    </TR>
  </TBODY>
</TABLE><BR>
-Jefferson Memorial<STRONG>
$139.95</STRONG><BR>
<% AddBuyButton 139.95,101,"Bird Feeder, Jefferson Memorial",5 %>
<BR>
<HR>

```

The AddBuyButton function sends the PunchOutOrderMessage back to the user.

The following listing is the include file (punchoutitem.inc) referenced in the previous example:

```

<%
REM This asp is included in items.asp, which specifies the item parameters, formats
REM a cXML document, and allows the user to proceed with a checkout of the item.
function CreateCXML(toUser, fromUser, buyerCookie, unitPrice, supPartId, desc)
%>
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE cXML SYSTEM
"http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML payloadID="&Now &"@ "&
Request.ServerVariables("LOCAL_ADDR")"&" timestamp="&Now
%>&">
  <Header>
    <From>
      <Credential domain="ariba.com">
        <Identity><%= toUser%></Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="ariba.com">
        <Identity><%= fromUser%></Identity>
      </Credential>
    </To>
  <Sender>

```

```

    <Credential domain=&quot;ariba.com&quot;>
      <Identity><%= toUser%></Identity>
    </Credential>
    <UserAgent>PunchoutSite</UserAgent>
  </Sender>
</Header>
<Message>
  <PunchOutOrderMessage>
    <BuyerCookie><%= buyerCookie%></BuyerCookie>
    <PunchOutOrderMessageHeader
      operationAllowed=&quot;edit&quot;>
      <Total>
        <Money currency=&quot;USD&quot;><%=
          unitPrice%></Money>
      </Total>
    </PunchOutOrderMessageHeader>
    <ItemIn quantity=&quot;1&quot;>
      <ItemID>
        <SupplierPartID><%= supPartId%></SupplierPartID>
        <SupplierPartAuxiliaryID><%= supPartAuxId%>
      </SupplierPartAuxiliaryID>
    </ItemID>
    <ItemDetail>
      <UnitPrice>
        <Money currency=&quot;USD&quot;><%= unitPrice%>
      </Money>
    </UnitPrice>
    <Description xml:lang=&quot;en&quot;><%= desc%>
    </Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification
      domain=&quot;SupplierPartID&quot;><%= supPartId%>
    </Classification>
    </ItemDetail>
  </ItemIn>
</PunchOutOrderMessage>
</Message>
</cXML>
<% end function

```

```
function AddBuyButton(unitPrice, supPartId, supPartAuxId, desc)
```

```

  toUser = Session("toUser")
  fromUser = Session("fromUser")

```

```

  buyerCookie = Session("buyercookie")
  url = Session("urlToPost")
  if not IsEmpty(buyerCookie) then
    %>
    <FORM METHOD=POST ACTION=<%= url%>>

```

```

        <INPUT TYPE=HIDDEN NAME="cxml-urlencoded" VALUE="<% CreateCXML
toUser, fromUser, buyerCookie, unitPrice, supPartId, supPartAuxId, desc%>">
        <INPUT TYPE=SUBMIT value=BUY>
    </FORM>
<%else%>
    </p>
    <%
end if
end function
%>

```

The AddBuyButton function contains the FORM POST that sends the URL-encoded PunchOutOrderMessage back to the user.

HTTP Form Encoding

To send a PunchOutOrderMessage, the supplier uses *HTML form encoding*, which is a different transport model from the traditional HTTP request/response model. This different transport facilitates easier integration between the supplier's website and the procurement application. It also enables buying organizations to receive XML data without requiring them to have a Web server available through a firewall.

Instead of sending a PunchOutOrderMessage directly to the procurement application, the supplier's website encodes it as a hidden HTML Form field and the user's browser submits it to the URL specified in the BrowserFormPost element of the PunchOutSetupRequest. The hidden HTML Form field must be named either cxml-urlencoded or cxml-base64, both case insensitive. Taken from the above example, the following code fragment inserts a hidden form field named cxml-urlencoded containing the PunchOutOrderMessage document to be posted:

```

<FORM METHOD=POST ACTION=<%= url%>>
    <INPUT TYPE=HIDDEN NAME="cxml-urlencoded" VALUE="<% CreateCXML toUser,
fromUser, buyerCookie, unitPrice, supPartId, supPartAuxId, desc%>">
    <INPUT TYPE=SUBMIT value=BUY>
</FORM>

```

This encoding permits the supplier to design a checkout Web page that contains the cXML document. When users click the supplier's "Check Out" button, the supplier's website presents the data, invisible to users, to the procurement application as an HTML Form Submit.

Cancelling PunchOut

The supplier might want to add a "Cancel" button to their pages so that users can cancel their PunchOut session. The "Cancel" button sends an empty PunchOutOrderMessage that tells the procurement application that no items will be

returned, and to delete existing PunchOut items from the requisition. The supplier can also use it to perform any housekeeping needed by the supplier's website, such as clearing the shopping cart and closing the user session.

Order Receiver Page

The Order Receiver Page accepts cXML purchase orders sent by buying organizations. It could be similar to the Launch Page discussed above. For information about receiving purchase orders, see Chapter 5, [“Purchase Orders.”](#)

PunchOut Website Suggestions

This section provides suggestions and information you should consider when planning the implementation of a PunchOut website.

Implementation Guidelines

Follow these guidelines when developing the supplier's PunchOut website:

- Study the *cXML User's Guide* (this document).
- Use an XML parser and validate documents against the cXML DTD.
- Use the `xml:lang=` property to identify users' languages so the supplier can provide localized content.
- Use the From credential to identify buying organizations.
- Send a unique, temporary URL for the session on redirect.
- Do not persist browser cookies.
- Do not overburden the supplier's customers with extrinsic data requirements.
- For each line item, use UNUOM (United Nations Units of Measure) and UNSPSC (Universal Standard Products and Services Classification).
- Provide real value to the supplier's customers. Display product availability, order status, and special promotions.
- Checkout should be easy and intuitive. Ideally, users should need to click only three buttons to buy.
- Code for subsequent edit and inspect sessions. Users cannot change order details for PunchOut items (such as quantity) within their procurement application. They must re-PunchOut with an edit session.

- For the greatest benefit to users, inspect sessions should display order status.
- Test the supplier's PunchOut website. Allow time for testing with the supplier's customers' procurement applications.
- PunchOut transactions produce quotes, not purchase orders. Implement a cXML purchase-order receiving page to accept orders.

Buyer and Supplier Cookies

The buyer and supplier cookies enable both buyers and suppliers to re-instantiate their own line-item data for their back-end systems.

- The supplier should return the BuyerCookie element they receive. It should not be changed.
- Make use of the supplier cookie (SupplierPartAuxiliaryID).

The buyer cookie is analogous to a purchase requisition number; it conveys state information that allows the buying organization's system to maintain the relationship between a requisition and a shopping basket.

Likewise, the supplier cookie is analogous to a quote number; it conveys state information that allows the supplier's system to maintain a relationship between a shopping basket and the buyer's requisition and purchase order. Procurement applications pass the supplier cookie back to the supplier in subsequent PunchOut edit or inspect sessions, and in the resulting purchase order. The supplier's website should take advantage of the supplier cookie to eliminate the need to pass visible, supplier-specific data back to the buyer.

Personalization

The header of the PunchOutSetupRequest always identifies the buying organization, but the request might also contain Contact and Extrinsic data (such as user's cost center, user's location, or product category) that the supplier can use to determine the dynamic URL to serve to the user.

Although not all buying organizations send this extrinsic data, it can enable the supplier to customize the supplier's Web store beyond the simple organization level. For example, the supplier could provide a separate Web store for each cost center within the buying organization (or each product category or each user).

The supplier could also store and display the user’s previous quotes. The supplier could allow users to reuse quotes, check the status of orders, and create reports on past activity. To avoid security problems, store quote history only at the per-user level.

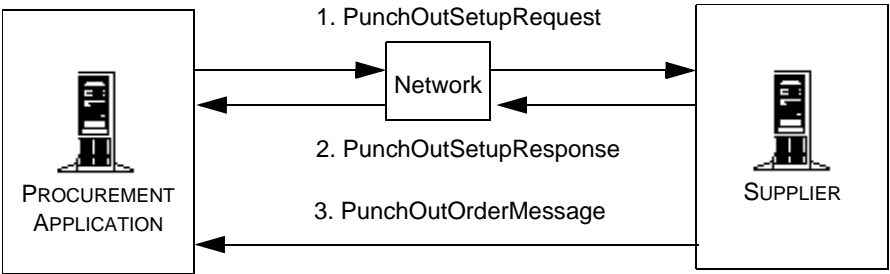
A key consideration during planning is the amount of effort required to implement a highly dynamic and customized PunchOut website. The supplier needs to balance between customization and complexity—a complex website takes longer to implement and maintain, but it could offer more value to users. It is recommended that the supplier start with a simple PunchOut website and enhance it over time.

PunchOut Transaction

The PunchOut message definitions are request/response messages within the Request and Response elements. All of the following messages must be implemented by suppliers to support PunchOut.

PunchOutSetupRequest and PunchOutSetupResponse are the request/response pair used to set up a PunchOut session to a remote system. The client uses them to identify the procurement application, send setup information, and receive a response indicating where to go to initiate an HTML browsing session on the remote website.

The order of cXML message flow in the PunchOut transaction is shown in the following diagram:



Sourcing

PunchOut can also be used for sourcing. A user can PunchOut from a procurement application to a sourcing application to initiate a RFQ (Request For Quote) session. The sourcing application will return a PunchOutSetupResponse with the URL of the start page of the sourcing application. With the URL, the end user goes to the sourcing application to provide more configuration information for RFQ.

At the end of each user session, a PunchOutOrderMessage is sent by the sourcing application to the procurement application and contains either a new RFQ, update information for an existing RFQ, or a completed RFQ. For information on using this feature, see [“PunchOutOrderMessage”](#) on page 104.

PunchOutSetupRequest

The PunchOutSetupRequest document contains a Header element and a PunchOutSetupRequest element.

Header

The Header element contains addressing and authentication information. Following is a sample Header element in a PunchOutSetupRequest.

```
<Header>
  <From>
    <Credential domain="DUNS">
      <Identity>65652314</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="DUNS">
      <Identity>83528721</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkId">
      <Identity>AN12345</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>
    <UserAgent>Procure Software 3.3</UserAgent>
  </Sender>
</Header>
```

From

The buying organization originating the PunchOutSetupRequest.

To

The supplier destination of the PunchOutSetupRequest.

Sender

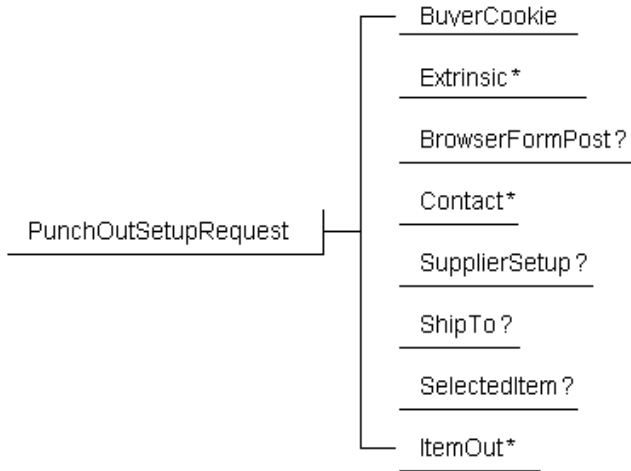
Authentication details of the buying organization including Identity, SharedSecret (password), and AribaNetworkId, which is specified by Credential domain. The SharedSecret is the supplier’s password or login to the PunchOut site.

UserAgent

An unique identifier for application sending the PunchOutSetupRequest. Consists of the software company name, product name, and version. Version details can appear in parentheses.

PunchOutSetupRequest

A PunchOutSetupRequest element is contained within the Request element. The following diagram shows the structure of the PunchOutSetupRequest element.



The following example shows a PunchOutSetupRequest:

```
<PunchOutSetupRequest operation="create">
  <BuyerCookie>34234234ADFSDf234234</BuyerCookie>
  <Extrinsic name="UserEmail">betty</Extrinsic>
  <Extrinsic name="UniqueName">BettyBuyer</Extrinsic>
  <Extrinsic name="CostCenter">Marketing</Extrinsic>
</PunchOutSetupRequest>
```

```
<BrowserFormPost>
  <URL>http://orms.acme.com:1616/punchoutexit</URL>
</BrowserFormPost>
<SelectedItem>
  <ItemID>
    <SupplierPartID>54543</SupplierPartID>
  </ItemID>
</SelectedItem>
<SupplierSetup>
  <URL>http://workchairs.com/cxml</URL>
</SupplierSetup>
<ShipTo>
  <Address addressID="1000467">
    <Name xml:lang="en">1000467</Name>
    <PostalAddress>
      <DeliverTo>Betty Buyer</DeliverTo>
      <Street>123 Main Street</Street>
      <City>Sunnyvale</City>
      <State>CA</State>
      <PostalCode>94089</PostalCode>
      <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
  </Address>
</ShipTo>
</PunchOutSetupRequest>
```

PunchOutSetupRequest has the following attribute:

operation	Specifies the type of PunchOutSetupRequest: "create", "inspect", "edit" or "source".
------------------	--

This element also contains the following elements: BuyerCookie, Extrinsic, BrowserFormPost, Contact, ShipTo, SelectedItem, SupplierSetup and an ItemOut list. Only the BuyerCookie element is required. The structure of Extrinsic, Contact, and ShipTo elements is discussed in more detail in [“OrderRequestHeader Element”](#) on page 116. The ItemOut element is discussed in [“ItemOut”](#) on page 125. In this context (outside of an OrderRequest), the Distribution and Comments elements and lineNumber, requisitionID, and requestedDeliveryDate attributes of an ItemOut add little or no value and should not be included. Because PunchOut sessions take place before ordering, this information is not relevant within a PunchOutSetupRequest.

An ItemOut list describes an existing shopping cart (items from a previous PunchOut session). The inspect operation initiates a read-only PunchOut session (enforced by both the client and the server) to view details about the listed items. The edit operation also starts from the previous shopping cart (described using the ItemOut list), but allows changes. Support for the edit operation implies inspect support (see

“[PunchOutOrderMessageHeader](#)” on page 105 and “[Empty Shopping Carts](#)” on page 106). This list can also describe items to be sourced. For more information, see “[Sourcing](#)” on page 98.

The Credential of the supplier is used to obtain the PunchOut location from the E-commerce network hub where suppliers can store the URLs of their PunchOut websites. E-commerce network hubs receive the PunchOutSetupRequest document, read the supplier’s ID, find the URL of the PunchOut website from the supplier’s account information, and send the PunchOutSetupRequest document to that URL. The e-commerce network hub, not the buyer, specifies the URL of the PunchOut website, which is more flexible. The URL specified in the SupplierSetup element of the PunchOutSetupRequest has been deprecated; cXML servers will ignore this element in the future.

BuyerCookie

This element transmits information that is opaque to the remote website, but it must be returned to the originator for all subsequent PunchOut operations. This element allows the procurement application to match multiple outstanding PunchOut requests. BuyerCookie is unique per PunchOut session.

BrowserFormPost

This element is the destination for the data in the PunchOutOrderMessage. It contains a URL element whose use will be further explained in the PunchOutOrderMessage definition. If the URL-Form-Encoded method is not being used, this element does not have to be included.

Extrinsic

This optional element contains any additional data that the requestor wants to pass to the external website. The cXML specification does not define the content of Extrinsic elements—it is something that each requestor and remote website must agree on and implement.

Extrinsic elements are intended to provide additional machine-readable information. They extend the cXML protocol to support features not required by all implementations. In the following context, the new data further describes the user initiating the PunchOut request.

```
<Extrinsic name="department">Marketing</Extrinsic>
```

The following example passes the user initiating the PunchOut and their department.

```
<Extrinsic name="CostCenter">450</Extrinsic>
<Extrinsic name="User">jsmith</Extrinsic>
```

With cXML 1.1 and higher, the Contact element obsoletes the “Cost Center” and “User” extrinsics.

The Extrinsic element can also appear in the OrderRequestHeader, ItemDetail, SpendDetail, and ContractItem elements. These contexts are described further elsewhere in this document.

SelectedItem

An optional SelectedItem element allows suppliers to specify PunchOut for an entire store or any subset of product offerings. Suppliers can create their catalogs so that SelectedItem leads to store-, aisle-, or product-level PunchOut. Procurement applications can include the SelectedItem element in PunchOutSetupRequest documents, and PunchOut sites can use it to determine which products to display to users. The more specific the item is in the catalog, the less searching users have to do at the supplier’s website. If there is no SelectedItem, suppliers should present their entire (store-level) product offerings.

A SelectedItem contains an ItemID, for example:

```
<SelectedItem>
  <ItemID>
    <SupplierPartID>5555</SupplierPartID>
  </ItemID>
</SelectedItem>
```

For the contents of the SelectedItem element, procurement applications use the ItemID (SupplierPartID and SupplierPartAuxiliaryID) from the PunchOut index catalog. No catalog changes are required.

Procurement applications should initially send both the new SelectedItem element and the old PunchOut URL in the PunchOutSetupRequest. E-commerce network hubs use the old URL only for suppliers that have not yet stored their PunchOut URL destinations.

This element is usually present in create operations. Procurement applications that allow users to punch out directly from a supplier listing should leave out SelectedItem in that case.

For edit and inspect operations, `SelectedItem` should appear only if the user chose to return to the supplier's website while viewing new information in the local catalog rather than items in an existing requisition. In either case, the current shopping cart must appear in the `ItemOut` list.

`SelectedItem` should not be used in a source operation.

SupplierSetup

This optional element specifies the URL to which to post the `PunchOutSetupRequest`. This element is not needed if the e-commerce network hub knows the supplier's `PunchOut` URL.

ShipTo

This optional element specifies the Ship To address for the item. Suppliers might want to use this information to formulate delivery time or price estimates.

PunchOutSetupResponse

After the remote website has received a `PunchOutSetupRequest`, it responds with a `PunchOutSetupResponse`, as shown below:

```
<PunchOutSetupResponse>
  <StartPage>
    <URL>
      http://premier.workchairs.com/store?23423SDFSDF23
    </URL>
  </StartPage>
</PunchOutSetupResponse>
```

StartPage

This element contains a `URL` element that specifies the URL to pass to the browser to initiate the `PunchOut` browsing session requested in the `PunchOutSetupRequest`. This URL must contain enough state information to bind to a session context on the remote website, such as the requestor identity and the appropriate `BuyerCookie` element.

At this point, the user who initiated the `PunchOutSetupRequest` browses the external website, and selects items to be transferred back to the procurement application through a `PunchOutOrderMessage`.

PunchOutOrderMessage

This element sends the contents of the remote shopping basket or sourcing RFQ to the originator of a PunchOutSetupRequest. It can contain much more data than the other messages because it must be able to fully express the contents of any conceivable shopping basket on the external website. This message does not strictly follow the Request/Response model.

The remote website generates a PunchOutOrderMessage when the user checks out. This message communicates the contents of the remote shopping basket to the procurement application; for example:

```
<PunchOutOrderMessage>
  <BuyerCookie>34234234ADFSD234234</BuyerCookie>
  <PunchOutOrderMessageHeader operationAllowed="create">
    <Total>
      <Money currency="USD">100.23</Money>
    </Total>
  </PunchOutOrderMessageHeader>
  <ItemIn quantity="1">
    <ItemID>
      <SupplierPartID>1234</SupplierPartID>
      <SupplierPartAuxiliaryID>
        additional data about this item
      </SupplierPartAuxiliaryID>
    </ItemID>
    <ItemDetail>
      <UnitPrice>
        <Money currency="USD">10.23</Money>
      </UnitPrice>
      <Description xml:lang="en">
        Learn ASP in a Week!
      </Description>
      <UnitOfMeasure>EA</UnitOfMeasure>
      <Classification domain="SPSC">12345</Classification>
      <LeadTime>1</LeadTime>
    </ItemDetail>
  </ItemIn>
</PunchOutOrderMessage>
```

A PunchOutOrderMessage document can be empty, which allows users to end PunchOut shopping sessions without selecting any items. Suppliers can implement a **Cancel** button that generates an empty PunchOutOrderMessage document. Then, both the PunchOut site and the procurement application know when a user has canceled a shopping session, and they can delete the shopping cart, delete items from the requisition, and perform other housekeeping tasks.

BuyerCookie

This element is the same element that was passed in the original PunchOutSetupRequest. It must be returned here to allow the procurement application to match the PunchOutOrderMessage with an earlier PunchOutSetupRequest.

PunchOutOrderMessageHeader

This element contains information about the entire shopping basket contents being transferred. The only required element is Total, which is the overall cost of the items being added to the requisition, excluding tax and shipping charges.

Additional elements that are allowed are Shipping and Tax, which are the amount and description of any shipping or tax charges computed on the remote website.

ShipTo is also optional, and it specifies the Ship-To addressing information the user selected on the remote site or that was passed in the original PunchOutSetupRequest.

All monetary amounts are in a Money element that always specifies currency in a standardized format.

The SourcingStatus element is optional, and relays updated information about a sourced RFQ. The content of the element could be a textual description of the update, such as the actual status update string displayed to the user.

PunchOutOrderMessageHeader has the following attributes:

operationAllowed	Specifies the operations allowed in subsequent PunchOutOrderRequests: create, inspect, or edit.
quoteStatus	Optional attribute specifies whether the order is "pending" or "final". If quoteStatus is "final", the transaction is complete.

The operationAllowed attribute controls whether the user can initiate later PunchOut sessions containing data from this PunchOutOrderMessage:

- operationAllowed="create": disallows subsequent PunchOut sessions for these items. Users cannot inspect or edit these items.
- operationAllowed="inspect": allows subsequent PunchOut sessions only to inspect these items. The items cannot be changed.
- operationAllowed="edit": allows subsequent PunchOut sessions to both inspect and change these items.

The `quoteStatus` attribute is used for an sourced RFQ or any other long-running operation. The `PunchOutOrderMessage` will contain the results of an end user session in the sourcing application and contains either status update information for a particular RFQ, a new RFQ, or an update to a completed RFQ.

Empty Shopping Carts

The `PunchOutOrderMessage` can contain a list of items corresponding to a shopping cart on the supplier website. It always indicates the end of the interactive `PunchOut` session. The following list describes a few cases when there are no items in the `PunchOutOrderMessage`. These messages allow clients to resume immediately when the user leaves the supplier website.

- If the operation in the original `PunchOutSetupRequest` was `inspect`, the item list of the `PunchOutOrderMessage` must be ignored by the procurement application. The supplier site should return no `ItemIn` elements in this case.
- If a `PunchOutOrderMessage` contains no `ItemIn` elements and the operation was `create`, no items should be added to the requisition because the supplier site or the user has cancelled the `PunchOut` session without creating a shopping cart.
- If the operation was `edit` and the `PunchOutOrderMessage` contains no `ItemIn` elements, existing items from this `PunchOut` session must be deleted from the requisition in the procurement application.

The Status code “204/No Content” indicates the end of a session without change to the shopping cart. Again, the `PunchOutOrderMessage` (which is always needed for the `BuyerCookie`) should not contain `ItemIn` elements. This code would be handled identically to the other “empty” cases detailed above unless the operation was `edit`. In that case, the user cancelled the session without making any change and no change should be made to the requisition in the procurement application.

ItemIn

This element adds an item from a shopping basket to a requisition in the procurement application. It can contain a variety of elements, only two of which are required: `ItemID` and `ItemDetail`.

ItemIn has the following attributes:

quantity	The number of items selected by the user on the remote website. Because the supplier site can enforce rules for partial units, the protocol allows fractional quantities. Should never be negative.
lineNumber (optional)	The position of this item within an order. Because PunchOut sessions normally take place prior to ordering and the server cannot control placement of items within an order in any case, this attribute is not relevant within a PunchOutOrderMessage.

The optional elements are ShipTo, Shipping, and Tax, which are the same elements as those specified in PunchOutOrderMessage, above. In addition, ItemIn can contain the optional SpendDetail, which can contain the optional TravelDetail, FeeDetail, LaborDetail, and Extrinsic elements. TravelDetail provides detailed information about travel and expense line items, FeeDetail provides information about fees not defined elsewhere, and LaborDetail provides detailed information about temporary labor line items. For more information, see [“TravelDetail”](#) on page 129, [“FeeDetail”](#) on page 149, [“LaborDetail”](#) on page 150, and [“Extrinsic”](#) on page 124.

The ItemIn and ItemOut structures match one-to-one, except for the Distribution and Comments elements and requisitionID and requestedDeliveryDate attributes available in the ItemOut element. The procurement application can convert directly between ItemIn and ItemOut lists when initiating an inspect or edit operation. Suppliers can convert one to the other (dropping the listed extensions available in the ItemOut element) when executing an edit operation. The procurement application can perform the direct conversion and add additional shipping and distribution information and comments when initiating an OrderRequest transaction. ItemDetail data (with the possible exception of Extrinsic elements) contained within ItemIn elements must not be removed when converting from ItemIn to ItemOut.

ItemID

This element uniquely identifies the item to the remote website. It is the only element required to return to the remote website to re-identify the item in later PunchOut sessions.

ItemID contains two elements: SupplierPartID and SupplierPartAuxiliaryID. Only SupplierPartID is required. SupplierPartAuxiliaryID helps the remote website transport complex configuration or bill-of-goods information to re-identify the item when it is presented to the remote website in the future.

If `SupplierPartAuxiliaryID` contains special characters (for example, if it contains additional XML elements not defined in the cXML protocol), they must be escaped properly. Due to the necessity to pass `SupplierPartAuxiliaryID` information through applications and back to the originating supplier, an internal subset containing any additional XML elements is insufficient.

ItemDetail

This element contains descriptive information about the item that procurement applications present to users. The contents of an `ItemDetail` element can be quite complex, but the minimum requirements are simple: `UnitPrice`, `Description`, `UnitOfMeasure`, and `Classification`. Optional elements include a `ManufacturerPartID`, a `ManufacturerName`, a `URL`, a `LeadTime`, and any number of Extrinsic elements.

In the context of an `ItemIn` element, the Extrinsic elements contained within an `ItemDetail` function identically to those found within an `Index` (specifically an `IndexItemAdd`).

The optional `LeadTime` element describes the number of days needed for the buyer to receive the product. For example:

```
<LeadTime>14</LeadTime>
```

Note that in an `IndexItemAdd` element, duplicate `LeadTime` information might come from both `ItemDetail`, where it is optional, and `IndexItemDetail`, where it is mandatory. If the `LeadTime` elements are defined in both cases, then they should be identical.

Description

This element describes the item in a textual form. Because this text might exceed the limits of a short table of line items (or other constrained user interface) and random truncations could occur, the `Description` element contains an optional `ShortName` element.

`ShortName` is a short (30-character recommended, 50-character maximum) name for the item, which fits product lists presented to users. If provided, clients should present the `ShortName` instead of a truncation of the `Description` text in any restricted fields. Clients must continue to truncate the `Description` text if no `ShortName` is provided.

For example:

```
<Description xml:lang="en-US">  
  <ShortName>Big Computer</ShortName>
```

This wonder contains three really big disks, four CD-Rom drives, two Zip drives, an ethernet card or two, much more memory than you could ever use, four CPUs on two motherboards. We'll throw in two monitors, a keyboard and the cheapest mouse we can find lying around.

```
</Description>
```

might appear as “Big Computer” where space is tight, and “Big Computer: This wonder ... lying around.” (or as two separate but complete fields) where there is space to display more text.

Catalog creators should not use ShortName to duplicate the information in Description. Instead, they should use ShortName to name the product, and Description to describe product details.

CIF 3.0 catalog format also supports ShortName. The CIF field name is Short Name.

SupplierList

In a sourced RFQ PunchOutOrderMessage, the ItemOut and ItemIn elements can specify a list of suppliers that can be involved in a sourcing process. SupplierList contains the Name and the list of SupplierIDs for each supplier. The following ItemOut example shows a SupplierList with two suppliers.

```
<ItemOut quantity="6" lineNumber="1">
  <ItemID>
    <SupplierPartID>unknown</SupplierPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">10.23</Money>
    </UnitPrice>
    <Description xml:lang="en">Learn ASP in a Week!</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="SPSC">12345</Classification>
    <ManufacturerPartID>ISBN-23455634</ManufacturerPartID>
    <ManufacturerName>O'Reilly</ManufacturerName>
    <URL> URL for more information </URL>
    <LeadTime>7</LeadTime>
  </ItemDetail>
  <SupplierList>
    <Supplier>
      <Name xml:lang="en">Supplier #1 </Name>
      <SupplierID domain="duns">0000000</SupplierID>
    </Supplier>
    <Supplier>
      <Name xml:lang="en">Supplier #2 </Name>
      <SupplierID domain="duns">1111111</SupplierID>
      <SupplierID domain="duns">2222222</SupplierID>
    </Supplier>
  </SupplierList>
</ItemOut>
```

Direct PunchOut

Direct PunchOut is a cXML capability that can reduce the time it takes for users to display the first page of a supplier's PunchOut site.

It offers faster PunchOut session initiation than regular PunchOut by allowing clients to send PunchOutSetupRequest documents directly to PunchOut sites for authentication, without first going through a network commerce hub for authentication and forwarding.

If suppliers indicate (through their cXML profile) that they support direct PunchOut, clients send PunchOut requests directly to them. Clients enable PunchOut sites to authenticate these requests by either including a Message Authentication Code (MAC) generated by a trusted third party, or by making a client digital certificate available.

Authentication Methods

Direct PunchOut is made possible by two alternative authentication methods:

- **MAC Authentication** — The server interprets a Message Authentication Code (MAC) in the Sender credential in PunchOutSetupRequest documents. This method is discussed in [“Message Authentication Code \(MAC\)”](#) on page 313.
- **Auth Transaction** — The server asks a network commerce hub to authenticate the client's digital certificate and caches the response for subsequent PunchOut requests. This method is discussed in [“Auth Transaction”](#) on page 318.

Servers indicate the authentication method they support through their cXML Profile.

ProfileResponse

PunchOut sites indicate that they support direct PunchOut and specify the authentication methods they support by including the following options for PunchOutSetupRequest in their ProfileResponse documents.

```
<Transaction requestName="PunchOutSetupRequest">
  <URL>https://service.bighub.com/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>
```

For more information about these options, see [“PunchOutSetupRequest Options”](#) on page 61.

Chapter 5

Purchase Orders

This chapter describes how to set up a website to receive cXML-format purchase orders. It also describes how to send purchase order status messages to buying organizations or marketplaces.

This chapter describes:

- [Purchase Order Process](#)
- [OrderRequest Documents](#)
- [Response to an OrderRequest](#)
- [Accepting Order Attachments](#)

Purchase Order Process

Procurement applications convert approved purchase requisitions into one or more purchase orders. A purchase order is a formal request from a buying organization to a supplier to fulfill a contract.

cXML is just one format for transmitting purchase orders. Other common formats are e-mail, fax, and ANSI X.12 EDI (Electronic Data Interchange). cXML is the best format for purchase orders because it allows you to easily automate order processing. cXML's well-defined structure allows order-processing systems to easily interpret the elements within a purchase order. With little or no human intervention, the appropriate data within purchase orders can be routed to your shipping, billing, and sales departments, as needed.

In addition, the cXML order-routing method allows the transmittal of any supplier cookies (SupplierPartAuxiliaryID) and purchase order attachments.

When you configure your account on a network commerce hub, you specify a URL to which all cXML purchase orders will be sent. Upon receiving a purchase order, you send it to your internal order management system and fulfill it as you normally would.

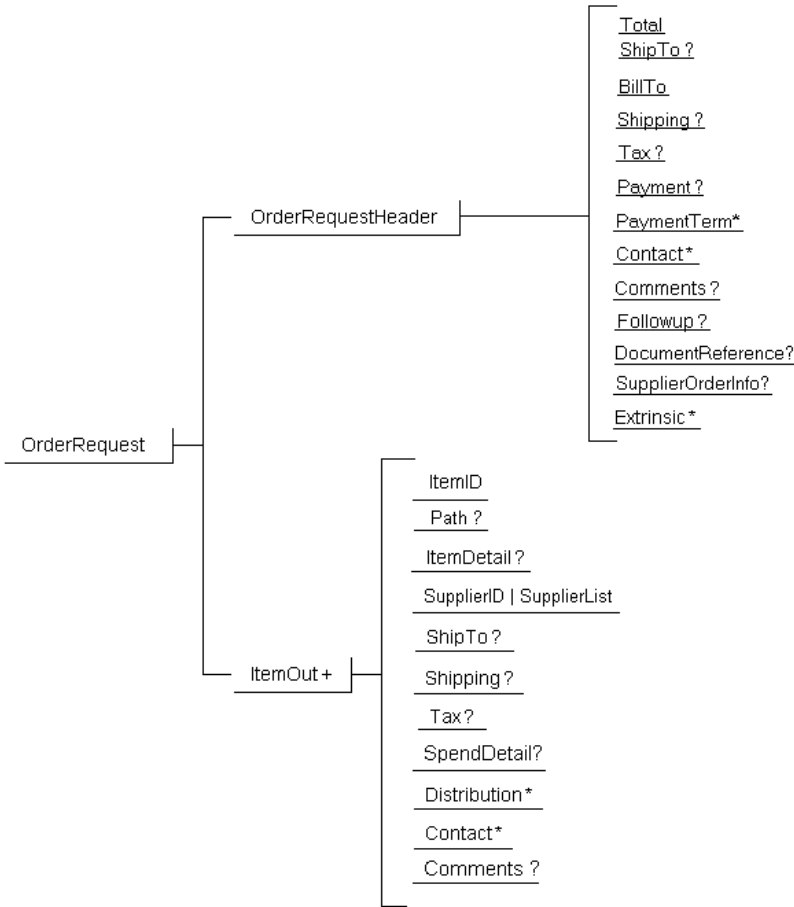
Your website must also return an Order Response document to the network commerce hub, which tells the buyer that you successfully received and parsed the purchase order.

You do not need a PunchOut website in order to receive cXML purchase orders; PunchOut and cXML order-receiving are distinct capabilities. However, the infrastructure and applications required for supporting PunchOut are the same for receiving cXML purchase orders.

There are two types of cXML documents used in the transaction of purchase orders. Procurement applications send OrderRequest documents, and you respond with generic Response documents. These documents pass through the network commerce hub for authentication and routing.

OrderRequest Documents

The OrderRequest document is analogous to a purchase order. The following diagram shows the structure of the OrderRequest element:



The following example shows the structure of the OrderRequest element:

```
<OrderRequest>
  <OrderRequestHeader ... >
  ...
</OrderRequestHeader>
<ItemOut ... >
  ...
</ItemOut>
<ItemOut ... >
  ...
```

```

</ItemOut>
</OrderRequest>

```

The following example shows an OrderRequest for an item:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML xml:lang="en-US" payloadID="93369535150910.10.57.136" timestamp="2000-08-
03T08:49:11+07:00">
<Header>
  <From>
    <Credential domain="AribaNetworkUserId">
      <Identity>admin@acme.com</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="DUNS">
      <Identity>114315195</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="AribaNetworkUserId">
      <Identity>sysadmin@ariba.com</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>
    <UserAgent>Ariba Network V1.1</UserAgent>
  </Sender>
</Header>
<Request>
  <OrderRequest>
    <OrderRequestHeader orderID="DO102880"
orderDate="2000-08-03T08:49:09+07:00" type="new">
      <Total>
        <Money currency="USD">4688.00</Money>
      </Total>
      <ShipTo>
        <Address isoCountryCode="US" addressID="1000467">
          <Name xml:lang="en">Acme, Inc.</Name>
          <PostalAddress name="default">
            <DeliverTo>John Q. Smith</DeliverTo>
            <DeliverTo>Buyers Headquarters</DeliverTo>
            <Street>123 Main Street</Street>
            <City>Mountain View</City>
            <State>CA</State>
            <PostalCode>94089</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="default">john_smith@acme.com</Email>
          <Phone name="work">
            <TelephoneNumber>

```

```

        <CountryCode isoCountryCode="United States">1
        </CountryCode>
        <AreaOrCityCode>800</AreaOrCityCode>
        <Number>5555555</Number>
    </TelephoneNumber>
</Phone>
</Address>
</ShipTo>
<BillTo>
    <Address isoCountryCode="US" addressID="12">
        <Name xml:lang="en">Acme Accounts Payable</Name>
        <PostalAddress name="default">
            <Street>124 Union Street</Street>
            <City>San Francisco</City>
            <State>CA</State>
            <PostalCode>94128</PostalCode>
            <Country isoCountryCode="US">United States</Country>
        </PostalAddress>
        <Phone name="work">
            <TelephoneNumber>
                <CountryCode isoCountryCode="US">1
                </CountryCode>
                <AreaOrCityCode>415</AreaOrCityCode>
                <Number>6666666</Number>
            </TelephoneNumber>
        </Phone>
        </Address>
    </BillTo>
    <Shipping>
        <Money currency="USD">12.34</Money>
        <Description xml:lang="en-US">FedEx 2-day</Description>
    </Shipping>
    <Tax>
        <Money currency="USD">10.74</Money>
        <Description xml:lang="en">CA State Tax</Description>
    </Tax>
    <Payment>
        <PCard number="1234567890123456" expiration="2002-03-12"/>
    </Payment>
</OrderRequestHeader>
<ItemOut quantity="2" lineNumber1>
    <ItemID>
        <SupplierPartID>220-3165</SupplierPartID>
        <SupplierPartAuxiliaryID>E000028901</SupplierPartAuxiliaryID>
    </ItemID>
    <ItemDetail>
        <UnitPrice>
            <Money currency="USD">2344.00</Money>
        </UnitPrice>
    </ItemDetail>
</ItemOut>

```

```

    <Description xml:lang="en">Laptop Computer Notebook Pentium® II
    processor w/AGP, 300 MHz, with 12.1" TFT XGA Display
  </Description>
  <UnitOfMeasure>EA</UnitOfMeasure>
  <Classification domain="UNSPSC">43171801</Classification>
  <URL>http://www.supplier.com/Punchout.asp</URL>
  <Extrinsic name="ExtDescription">Enhanced keyboard</Extrinsic>
</ItemDetail>
<Distribution>
  <Accounting name="DistributionCharge">
    <AccountingSegment id="7720">
      <Name xml:lang="en-US">Account</Name>
      <Description xml:lang="en-US">Office Supplies
    </Description>
    </AccountingSegment>
    <AccountingSegment id="610">
      <Name xml:lang="en-US">Cost Center</Name>
      <Description xml:lang="en-US">Engineering Management
    </Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">4688.00</Money>
  </Charge>
</Distribution>
</ItemOut>
</OrderRequest>
</Request>
</cXML>

```

OrderRequestHeader Element

The following example shows an OrderRequestHeader in full detail:

```

<OrderRequestHeader
  orderID="DO1234"
  orderDate="1999-03-12T13:30:23+8.00"
  type="new"
  requisitionID="R1234"
  shipComplete="yes">
  <Total>
    <Money currency="USD">12.34</Money>
  </Total>
  <ShipTo>
    <Address>
      <Name xml:lang="en">Acme Corporation</Name>
      <PostalAddress name="Headquarters">
        <DeliverTo>Joe Smith</DeliverTo>
        <DeliverTo>Mailstop M-543</DeliverTo>
      </PostalAddress>
    </Address>
  </ShipTo>
</OrderRequestHeader>

```

```

        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>90489</PostalCode>
        <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
</Address>
</ShipTo>
<BillTo>
    <Address>
        <Name xml:lang="en">Acme Corporation</Name>
        <PostalAddress name="Finance Building">
            <Street>124 Anystreet</Street>
            <City>Sunnyvale</City>
            <State>CA</State>
            <PostalCode>90489</PostalCode>
            <Country isoCountryCode="United States">United States</Country>
        </PostalAddress>
    </Address>
</BillTo>
<Shipping>
    <Money currency="USD">12.34</Money>
    <Description xml:lang="en-US">FedEx 2-day</Description>
</Shipping>
<Tax>
    <Money currency="USD">12.34</Money>
    <Description xml:lang="en">CA State Tax</Description>
</Tax>
<Payment>
    <PCard number="1234567890123456" expiration="1999-03-12"/>
</Payment>
<PaymentTerm payInNumberOfDays="45">
</PaymentTerm>
<PaymentTerm payInNumberOfDays="30">
    <Discount>
        <DiscountPercent percent="2">
    </Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays="20">
    <Discount>
        <DiscountPercent percent="3">
    </Discount>
</PaymentTerm>
<Contact role="purchasingAgent">
    <Name xml:lang="en-US">Mr. Purchasing Agent</Name>
    <Email>puragent@acme.com</Email>
    <Phone name="Office">
        <TelephoneNumber>
            <CountryCode isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode>800</AreaOrCityCode>

```

```

        <Number>555-1212</Number>
      </TelephoneNumber>
    </Phone>
  </Contact>
  <Comments xml:lang="en-US">
    Anything well formed in XML can go here.
  </Comments>
  <DocumentReference payloadID="1234567890123--
    1234567890123456789@10.10.10.100">
  </DocumentReference>
  <SupplierOrderInfo orderID=12345>
</OrderRequestHeader>

```

OrderRequestHeader has the following attributes:

orderID	The identifier for this order. Analogous to the purchase order number.
orderDate	The date and time this order was placed, in ISO 8601 format.
orderType (optional)	Type of the order: regular or release (a release against a master agreement).
type (optional)	Type of the request: new (default), update, or delete. Update and delete orders must use the DocumentReference element with the PayloadId to refer to the original purchase order. See "DocumentReference Element" on page 203.
orderVersion (optional)	Specifies the order version number of change orders, starting with "1" for the original order.
isInternalVersion (optional)	Indicates whether the order includes changes that are relevant only within the buying organization. For example, a minor change was made that does not affect information used by the supplier. Suppliers might not see internal order versions, depending on their customers' configuration.
agreementID (optional)	Buyer's identifier for associated master agreement.
agreementPayloadID (optional)	cXML document payload ID for the associated master agreement.
requisitionID (optional)	The buyer's requisition identifier for this entire order. It might be the same as orderID, and it might not be included at all. Must not be included if requisitionID is specified in any ItemOut elements.
shipComplete (optional)	A preference against partial shipments. The only allowed value is "yes". By default, items are shipped when available. Because orders might include items with varying ShipTo elements, only groups of items with common shipping locations should be held until complete when shipComplete="yes".

OrderRequestHeader and ItemOut (when extended with ItemDetail) contain similar information. Where OrderRequestHeader includes overall billing (BillTo) and payment (Payment, PaymentTerm) information, ItemOut describes the individual items (in ItemID, ItemDetail, SpendDetail, and Distribution).

Do not use the information in OrderRequestHeader as the default for item-specific elements. If present, ShipTo, Shipping, Contact, and each named Extrinsic must appear either with every ItemOut or in the OrderRequestHeader. The Comments and Tax elements can appear simultaneously at both levels; however, the header-level Tax element contains a total for the order, whereas the item-level Tax element contains the tax just for the item. Do not include duplicate information in Comments elements at both levels.

Total

This element contains the total cost for the items in the order, excluding any tax and shipping. It is a container for the Money element.

ShipTo/BillTo

These elements contain the addresses of the Ship To and Bill To entities on the OrderRequest.

All items must be billed to a single entity. Therefore, the BillTo element appears only in the OrderRequestHeader. Items from an order can be sent to multiple locations. Like the Shipping element (see next section), the ShipTo element can therefore appear either in the OrderRequestHeader or in individual ItemOut elements.

The Address element contains an addressID attribute that specifies an ID for the address. This attribute is used to support address codes for relationships that require ID references. This value should not be the name of a company or person. It is intended to deepen application-to-application integration. For example, a ShipTo location identifier could be:

```
<Address isoCountryCode="US" addressID="1000487">
```

The Name element contained within an Address element should always specify the company name.

The DeliverTo element is listed twice, the first line specifying the name of the person to receive the goods, and the second specifying their location (building, city, office, mailstop) where the items should be delivered. The location should always be complete enough to be used in a mailing label. For example,

```
<PostalAddress name="Headquarters">
  <DeliverTo>Joe Smith</DeliverTo>
  <DeliverTo>Mailstop M-543</DeliverTo>
  <Street>123 Anystreet</Street>
  <City>Sunnyvale</City>
  <State>CA</State>
  <PostalCode>90489</PostalCode>
  <Country isoCountryCode="US">UnitedStates</Country>
</PostalAddress>
```

Country contains a human readable name. The isoCountryCode attribute value is the ISO country code from the ISO 3166 standard.

Avoid empty or whitespace elements because missing values can affect EDI and cXML suppliers.

Shipping

This element describes how to ship line items and the shipping cost. If the Shipping element is present in the OrderRequestHeader, it must not appear in the ItemOut elements. If it is not present in the OrderRequestHeader, it must appear in the ItemOut elements.

Tax

This element contains the tax associated with the order. This element is present if the buying organization computes tax. When appearing within the OrderRequestHeader, Tax describes the total tax for an order. Tax elements at the item level can describe line item tax amounts.

Payment

Describes the payment instrument used to pay for the items requested. In the above example, the Payment element contains a PCard element, which encodes a standard purchasing card into the cXML document. In the future, other payment instruments might be defined.

PaymentTerm

Defines the payment term in orders and invoices. Use PaymentTerm instead of the InvoiceDetailPaymentTerm previously defined. PaymentTerm defines either the net term (without discount) or the discount term (with discount).

PaymentTerm has one attribute:

payInNumberOfDays	Indicates the invoice must be paid in a certain number of days after the invoice effective date.
--------------------------	--

Discount

The percentage or amount of the discount term. The discount rate applies if the invoice total is paid within the time specified by payInNumberOfDays. Positive rates denote discounts and negative rates denote penalties. Do not use a percentage sign (%) or divide by 100; for example “2” means 2%.

Do not use the Discount element if the PaymentTerm is a net term.

Contact

The supplier uses Contact element information to follow up on an order. This element identifies a person and provides a list of ways to reach that person or entity. The only required element is the Name of the contact. Optional and repeating possibilities include PostalAddress (not recommended for immediate correction of order problems), Email, Phone, Fax, and URL.

In cXML 1.0, the extrinsics User and CostCenter elements often provided contact information. With cXML 1.1 and higher, the Contact element provide alternatives to these extrinsics.

Buying organizations might choose to use this element to identify the original requestor, the procurement application system administrator, or some other contact who can take responsibility for correcting problems with orders. Contact can differ from both BillTo and ShipTo information for an order.

Contact has the following attributes:

role (optional)	The position of this person within the procurement process.
addressID (optional)	An ID for the address. addressID is supports address codes for relationships that require id references.

Possible values for the role attribute:

role value	Description
technicalSupport	Technical support contact
customerService	Customer service contact

role value	Description
sales	Sales contact
buyerCorporate	Contact details the supplier has about the buying organization.
supplierCorporate	Contact details about the supplier.

The same Contact role must not appear at both the header and item levels.

There is no default role, due to the disparate contents of the Contact element. So, cXML applications treat a Contact without a role attribute as an additional role.

TelephoneNumber

The TelephoneNumber element contains the telephone number of the person or department where the goods are to be shipped or billed. For example, a telephone number in the United States:

```
<TelephoneNumber>
  <CountryCode isoCountryCode="US">1</CountryCode>
  <AreaOrCityCode>800</AreaOrCityCode>
  <Number>555-1212</Number>
</TelephoneNumber>
```

For international dialing, the CountryCode contains the dial code for a country after any escape codes. England, for example, would be represented as:

```
<CountryCode isoCountryCode="UK">44</CountryCode>
```

The following, therefore, is an example for London:

```
<TelephoneNumber>
  <CountryCode isoCountryCode="UK">44</CountryCode>
  <AreaOrCityCode>137</AreaOrCityCode>
  <Number>2801007</Number>
</TelephoneNumber>
```

Fax

The Fax element specifies the Fax number of the person or department where goods are to be shipped or billed. This element contains the TelephoneNumber element described above.

Comments

Arbitrary human-readable information buyers can send within purchase orders. This string data is not intended for the automated systems at supplier sites.

The Comments element can contain an Attachment element for including external files.

Attachment

Comments can attach external files to augment purchase orders. The Attachment element appears within Comments, and it contains only a reference to the external MIME part of the attachment. All attachments should be sent in a single multipart transmission with the OrderRequest document. Even if this is not possible, the contentID provided by the Attachment element must be usable to retrieve the attachment.

For details about the transfer of attached files, see [“Attachments”](#) on page 31.

Attachment contains a single URL with scheme “cid:”. An attached file in a cXML document might appear as:

```
<Comments>
  <Attachment>
    <URL>cid: uniqueCID@cxml.org</URL>
  </Attachment>
  Please see attached image for my idea of what this
  should look like
</Comments>
```

The Comments element appears in many places within the cXML protocol, but it can contain the Attachment element only within OrderRequest documents.

Use Comments to provide local comments specific to the current document.

Followup

The use of the Followup element is strongly discouraged. In early implementations, Followup was used to specify the URL to which future StatusUpdateRequest documents should be posted.

All cXML implementations should use the more robust Profile transaction to retrieve and convey information about server capabilities, including supported cXML version, supported transactions, and options to those transactions. See [“Profile Transaction”](#) on page 24 for more information.

DocumentReference

This element provides an exact reference to an earlier document (for example, OrderRequest,MasterAgreementRequest, or InvoiceReference). In a StatusUpdateRequest, DocumentReference identifies the purchase order to be updated.

SupplierOrderInfo

This element is used in OrderRequestHeader to define supplier sales order information related to the current order. SupplierOrderInfo is used in OrderRequest and InvoiceDetailRequest documents.

When SupplierOrderInfo is used in a PunchOutOrderMessage, it indicates that the supplier has created an order associated with the PunchOut order message. The buyer can later cancel the order by sending an OrderRequest of type “delete” and including the SupplierOrderInfo element in the OrderRequestHeader. to refer to the sales order to be deleted.

SupplierOrderInfo has one attribute:

orderId	Supplier sales order id of this order.
----------------	--

Extrinsic

This element contains machine-readable information related to the order, but not defined by the cXML protocol. In contrast, the Comments element passes information for human use. Extrinsic elements contain data that is likely to appear in later documents; the Comments element does not. At this level, Extrinsic extends the description of all items contained in the purchase order. Some Extrinsic information might also describe the overall purchase order without affecting the meaning of any contained ItemOut.

Each named Extrinsic can appear only once within the lists associated with the OrderRequestHeader and individual ItemOut elements (within the contained ItemDetail elements). The same name must not appear in both the OrderRequestHeader list and any list associated with the ItemOut elements. If the same Extrinsic name and value is repeated in all ItemOut lists, it should be moved to the OrderRequestHeader.

The Extrinsic element can also appear in the IndexItem, PunchOutSetupRequest and ContractItem elements. These contexts are described later in this document. Extrinsic values are case-insensitive.

ItemOut

The following example shows a minimum valid ItemOut element.

```
<ItemOut quantity="1"
  lineNumber="1">
  <ItemID>
    <SupplierPartID>5555</SupplierPartID>
  </ItemID>
</ItemOut>
```

ItemOut has the following attributes:

quantity	The number of items desired. Fractions are allowed for some units of measure. The value might have already been checked by the supplier during a PunchOut session. This value should never be negative.
lineNumber (optional)	Position of the item within an order. This ordinal value increases once per ItemOut in a “new” OrderRequest. Clients should always specify this attribute in an OrderRequest, although it might not be useful in other ItemOut contexts.
requisitionID (optional)	The buyer’s requisition identifier for this line item. Must not be included if requisitionID is specified in the OrderRequestHeader.
agreementItemNumer (optional)	The buyer’s master agreement identifier for the line item.
requestedDeliveryDate (optional)	The date item was requested for delivery, which allows item-level delivery dates in the OrderRequest. It must be in ISO 8601 format.
isAdHoc (optional)	Indicates that the item is a non-catalog (ad-hoc) item. Non-catalog purchase orders contain items entered manually by requisitioners, not items selected from electronic catalogs. Often, these items do not have valid part numbers. Non-catalog orders usually require special validation and processing. Users enter non-catalog items to purchase products and services on an ad-hoc basis or because they could not find them in electronic catalogs.

The lineNumber attribute remains constant for any item through updates to the order. Deletion of items from an order never changes the lineNumber of remaining items. New items have higher numbers than those previously included in the order. A change to an existing item (an increased quantity, for example) does not affect the lineNumber of that item.

The following example shows a more complicated ItemOut.

```

<ItemOut quantity="2" lineNumber="1"
  requestedDeliveryDate="1999-03-12">
  <ItemID>
    <SupplierPartID>1233244</SupplierPartID>
    <SupplierPartAuxiliaryID>ABC</SupplierPartAuxiliaryID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">1.34</Money>
    </UnitPrice>
    <Description xml:lang="en">hello</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="UNSPSC">12345</Classification>
    <ManufacturerPartID>234</ManufacturerPartID>
    <ManufacturerName xml:lang="en">foobar</ManufacturerName>
    <URL>www.bar.com</URL>
  </ItemDetail>
  <ShipTo>
    <Address>
      <Name xml:lang="en">Acme Corporation</Name>
      <PostalAddress name="Headquarters">
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>90489</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Address>
  </ShipTo>
  <Shipping>
    <Money currency="USD">1.34</Money>
    <Description xml:lang="en-US">FedEx 2-day</Description>
  </Shipping>
  <Tax>
    <Money currency="USD">1.34</Money>
    <Description xml:lang="en">foo</Description>
  </Tax>
  <Distribution>
    <Accounting name="DistributionCharge">
      <AccountingSegment id="23456">
        <Name xml:lang="en-US">G/L Account</Name>
        <Description xml:lang="en-US">Entertainment</Description>
      </AccountingSegment>
      <AccountingSegment id="2323">
        <Name xml:lang="en-US">Cost Center</Name>
        <Description xml:lang="en-US">Western Region Sales</Description>
      </AccountingSegment>
    </Accounting>
  </Distribution>
  <Charge>

```



```

        <Money currency="USD">.34</Money>
      </Charge>
    </Distribution>
  <Distribution>
    <Accounting name="DistributionCharge">
      <AccountingSegment id="456">
        <Name xml:lang="en-US">G/L Account</Name>
        <Description xml:lang="en-US">Travel</Description>
      </AccountingSegment>
      <AccountingSegment id="23">
        <Name xml:lang="en-US">Cost Center</Name>
        <Description xml:lang="en-US">Europe Implementation
      </Description>
      </AccountingSegment>
    </Accounting>
  <Charge>
    <Money currency="USD">1</Money>
  </Charge>
</Distribution>
<Comments xml:lang="en-US">Comment</Comments>
</ItemOut>

```

The `ItemDetail` element allows additional data to be sent to suppliers instead of just the unique identifier for the item represented by the `ItemID`.

If `isAdHoc="yes"` exists for some items and not for others, the requisition should be broken into two requisitions: one for catalog items and one for non-catalog items. Suppliers will then be able to automatically process as many requisition items as possible, instead of having to manually process both catalog and non-catalog items.

The `ShipTo`, `Shipping`, `Tax`, `Contact`, `Comments`, and `Extrinsic` elements (some nested within `ItemDetail` or `SpendDetail`) are identical to the ones that can be in the `OrderRequestHeader`. These elements specify per-item data such as shipping, shipping type, and associated cost. Use these elements either at the `OrderRequestHeader` level, or at the `ItemOut` level, but not at both levels. Tax is the only exception, for more information, see [“Tax”](#) on page 120.

ItemID

The basic `ItemID` element, which provides unique identification of an item. `ItemID` is defined at [“ItemID”](#) on page 107.

Path

The basic `Path` element, which provides node and path information for a document. `Path` is defined at [“Path Element”](#) on page 158.

ItemDetail

The basic ItemDetail element, which contains descriptive information about a line item that procurement applications present to users. ItemDetail is defined at “ItemDetail” on page 108.

SupplierID

See “SupplierID” on page 128.

SupplierList

Defines a list of suppliers that might be associated with a quote item in ItemOut.

SupplierList has no attributes.

Supplier

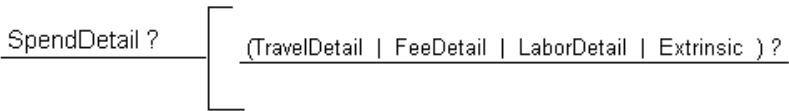
The common Supplier element is optional in ItemOut.

ShipTo, Shipping, and Tax

The common elements, described elsewhere in this document.

SpendDetail

This optional element provides detailed information regarding travel, fee, and labor line items. The following diagram shows the structure of the SpendDetail element:



SpendDetail can be present in ItemIn and ItemOut elements for the following types of messages:

- PunchOutSetupRequest
- PunchOutOrderMessage
- OrderRequest

- ConfirmationRequest

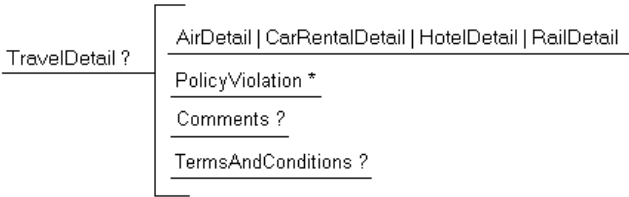
SpendDetail has no attributes.

The basic ItemIn element adds an item from a shopping basket to a requisition in the procurement application during a PunchOut session. ItemIn is defined at “ItemIn” on page 106.

TravelDetail

TravelDetail describes information about travel line items.

The following diagram shows the structure of TravelDetail:



The following example shows the location of SpendDetail and TravelDetail within an OrderRequest document:

```
<OrderRequest... >
  <OrderRequestHeader >
    ...
  </OrderRequestHeader >
  <ItemOut>
    <ItemDetail >
      ...
    </ItemDetail>
    <SpendDetail>
      <TravelDetail>
        ...
      </TravelDetail>
    </SpendDetail>
  </ItemOut>
</OrderRequest>
```

TravelDetail has the following attributes:

confirmationNumber	A unique confirmation number that is understood by both the traveler and the vendor who is providing the service for this travel line item. For example, hotel reservation number or e-ticket number from the airline.
pnrLocator (optional)	Passenger Name Record (PNR) locator used by the travel booking provider.
quoteExpirationTime (optional)	Date and time that this quote will expire. This value is normally supplied in the PunchoutOrderMessage. If no value is supplied, it is assumed that this quote will not expire.

Common Elements

Several common elements are used throughout TravelDetail.

Date and Time in cXML

Dates and times in cXML must be formatted in the restricted subset of ISO 8601. This is described in the Word Wide Web Consortium (W3C) Note entitled “Date and Time Formats” available at www.w3.org/TR/NOTE-datetime-970915.html. See “Date, Time, and other Data Types” on page 37 for more information.

Vendor

The common Vendor element. When used in TravelDetail, Vendor provides information about a vendor of a service. Vendor can be used in AirLeg, CarRentalDetail, HotelDetail, and RailLeg.

Vendor has one attribute:

preferred	Is this a preferred vendor? yes no
------------------	---

Address

The basic Address element provides the physical address of the vendor. Typically, this is the address the vendor’s a business location or headquarters. Address is described further in “cXML Conventions” on page 29.

SupplierID

Supplier ID for this vendor. This is a (domain, value) pair so that travel booking providers can define their SupplierID elements according to the convention they prefer, such as D-U-N-S or TaxID.

SupplierID has one required attribute: domain.

Each travel booking provider can specify multiple Supplier ID values. This capability enables a provider to use a single implementation to coordinate with various enterprise implementations that use different SupplierID domains.

TermsAndConditions

Text descriptions of terms and conditions associated with a travel line item. For example, a car rental TermsAndConditions normally includes boundary limit, additional mileage charges, gasoline charges, and other restriction information. Multiple TermsAndConditions can be included in a single travel line item.

Description

Text description of terms and conditions. If TermsAndConditions is present, Description is required.

PolicyViolation

Line-item level policy violation that results from the user selecting this particular travel item. Policy violations are not associated at the header level to ensure clear identification of a violation with the appropriate line item.

PolicyViolation has one attribute:

level	The level of the PolicyViolation: warning violation warning - a non-serious violation violation - a serious violation of company policy
-------	---

Description

Description is a common free-text element, which provides a textual description of an element, such as PolicyViolation.

PolicyViolationJustification

Justification for this PolicyViolation. Typically, the user selects a PolicyViolationJustification from a standard list of justifications at the travel booking provider’s web site.

- **Comments**
Additional comments to further clarify the PolicyViolationJustification, given by the user.

Penalty

Penalty (if any) for this travel segment.

Money

The penalty amount.

Description

Textual description of the cause of the penalty. For example, a change fee associated with an air ticket.

AvailablePrice

The common AvailablePrice element describes other available prices that the user did not select.

AvailablePrice has one attribute:

type	Description of the level of compliance with company policy.
-------------	---

Possible values for the type attribute of AvailablePrice:

Value	Description
lowest	Lowest price available regardless of the travel policies
lowestCompliant	Lowest price available that is compliant with travel policies
highestCompliant	Highest price available that is compliant with travel policies
highest	Highest price available regardless of the travel policies
other	Other, specify in the Description

Money

The amount of an available price.

Description

A text description of an available price, including information on how the price was found or particular requirements for the price.

Rate

Defines the rate for a travel item. The following example shows a Rate element for a CarRentalFee:

```
<CarRentalFee type="baseRate">
  <Total>
    <Money currency="USD">215.99</Money>
  </Total>
  <Rate quantity="4">
    <Total>
      <Money currency="USD">119.96</Money>
    </Total>
    <UnitRate>
      <Money currency="USD">215.99</Money>
      <UnitOfMeasure>WEE</UnitOfMeasure>
    </UnitRate>
  </Rate>
</CarRentalFee type="baseRate">
```

Rate has one attribute, which is required:

quantity	The quantity for the rate. For example, a four-night stay at a hotel would be expressed as: quantity = 4 UnitofMeasure in UnitRate = DAY
----------	--

Total

The total amount for the rate. The total amount must equal to quantity x UnitRate. All Rate amounts for a line item must add up to the Total for that line item.

UnitRate

UnitRate defines the rate for a single unit according to the unit of measure. For example, a single nightly rate for a hotel room can be expressed with Money equal to the nightly rate amount and the UnitOfMeausre equal to DAY.

The amount to be paid per unit (of time or other measure). In the case of multiple UnitRates (a rate schedule), it is customary and expected that each UnitRate will include a TermReference to distinguish it.

Description

Textual description for the rate. For a hotel stay, the Description could contain “hotel nightly rate.”

BookingClassCode

BookingClassCode is a common element. When used in a travel line item, it indicates the class of the line item. For example, BookingClassCode is commonly used to convey frequent flyer information for air travel reservations.

Each buyer-travel booking provider pair can use any industry standard they choose. The following example shows a minimal BookingClassCode element:

```
<BookingClassCode code="W">
  <Description xml:lang="en">Coach class</Description>
</BookingClassCode>
```

For information on car rental codes, see [CarRentalDetail](#).

BookingClassCode has the following attributes:

domain	The domain for this code, for example, IATA.
code	An industry standard code, or per agreement of buyer-travel booking provider pair.

Description

The Description element of BookingClassCode contains a text description of the code.

Airport

The common Airport element, which contains the three-letter IATA airport code, is used in AirLegOrigin, AirLegDestination, CarRentalPickup, CarRentalDropoff, HotelDetail, RailLegOrigin and RailLegDestination.

Airport has one attribute:

airportCode	The three-letter IATA airport code. For information on the International Air Transport Association (IATA) standard, see: http://www.iata.org/codes/index .
--------------------	---

- **Address**
The optional Address element provides the physical address of the airport.

Meal

The Meal element of an AirLeg can contain two optional, common elements: BookingClassCode and Description. The following example represents a heated vegetarian dinner for an AirLeg.

```
<Meal>
  <Description xml:lang="en">vegetarian dinner</Description>
  <BookingClassCode code="H"></BookingClassCode>
</Meal>
```

- **Description**
A text description of the meal, including any special needs such as vegetarian, gluten-free, or dairy-free.
- **BookingClassCode**
The common BookingClassCode element is defined at “BookingClassCode” on page 134. The BookingClassCode of Meal defines the code for the meal. For example, airlines typically use the following meal codes:

Code	Description
B	Breakfast
C	Complimentary liquor
D	Dinner
F	Food for purchase
G	Food and beverage for purchase
H	Hot meal
K	Continental breakfast
L	Lunch
M	Meal
N	No meal service
O	Cold meal

Code	Description
P	Liquor for purchase
R	Refreshments
S	Snack or brunch
V	Refreshments for purchase

- **Description**
A text description of the meal. Use this description to provide details not included in BookingClassCode.

For more information about xml standards and basic elements, see Chapter 2, “cXML Basics.”

AirDetail

The AirDetail element provides information about an air trip. The following diagram shows the structure of AirDetail:



AirDetail has no attributes.

TripType

TripType is a container for the type attribute, which is required in both AirDetail and RailDetail to indicate a round trip, one way, or multi-leg trip.

For example, a TripType for a round trip would appear as:

```
<TripType type="round"></TripType>
```

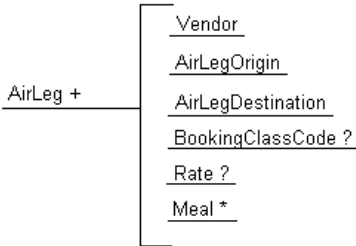
Possible values for type attribute of TripType:

Attribute	Description
round	round trip
oneWay	one-way trip
multiLeg	multi-leg or open-jaw trip

AirLeg

Each AirDetail must include at least one AirLeg element.

The following diagram shows the structure of AirLeg:



The AirLeg element provides detailed information about a trip that includes one or more airplane flights. The following example shows an AirLeg element for a one-way flight:

```
<AirLeg travelSegment="1"
  departureTime="2004-12-01T16:10:00-08:00"
  arrivalTime="2004-12-01T17:10:00-08:00"
  flightNumber="SW 990"
  seatNumber="20F"
  seatType="aisle"
  stops="0"
  equipment="Boeing 737">
  <Vendor preferred="no">
    <Address>
      ...
    </Address>
  </Vendor>
  <AirLegOrigin>
    <Airport airportCode="SFO">
      <Address>
        ...
      </Address>
```

```

        </Airport>
    </AirLegOrigin>
    <AirLegDestination>
        <Airport airportCode="BUR">
            <Address>
                ...
            </Address>
        </Airport>
    </AirLegDestination>
    <BookingClassCode code="W">
        <Description xml:lang="en">Coach class</Description>
    </BookingClassCode>
    <Meal type="snack">
        <Description xml:lang="en">Vegetarian snack</Description>
    </Meal>
</AirLeg>

```

AirLeg has the following attributes:

travelSegment	Text description to identify this travel segment. This information is specific to the travel booking provider.
departureTime	Local departure date and time for this air leg.
arrivalTime	Local arrival date and time for this air leg.
flightNumber	Flight number for this air leg.
seatNumber (optional)	Seat number for this air leg.
seatType (optional)	Seat type: window, aisle, or middle
upgrade (optional)	Is this ticket an upgrade: no (default), or yes
stops (optional)	The number of stop for this air leg. Use a numeral for the number of stops, or '0' (zero) for a direct flight. If no numeral is entered, '0' (zero) is implied.
equipment (optional)	The plane equipment information for this air leg. For example, name the type of airplane used.

Vendor

The common Vendor element, which provides information about the vendor of a service, is defined at “Vendor” on page 130.

AirLegOrigin / AirLegDestination

These elements contain the addresses of the AirLegOrigin and AirLegDestination entities on the AirLeg.

The following example shows a detailed AirLeg for a flight from San Francisco to Miami.

```

<AirLegOrigin>
  <Airport airportCode="SFO">
    <Address>
      <Name xml:lang="en">San Francisco Internal Airport</Name>
      <PostalAddress>
        <Street>San Francisco International Airport</Street>
        <City>San Francisco</City>
        <State>CA</State>
        <PostalCode>94128</PostalCode>
        <Country isoCountryCode="US">UnitedStates</Country>
      </PostalAddress>
    </Address>
  </Airport>
</AirLegOrigin>
<AirLegDestination>
  <Airport airportCode="MIA">
    <Address>
      <Name xml:lang="en">Miami International Airport</Name>
      <PostalAddress>
        <Street>4200 NW 21 Street</Street>
        <City>Miami</City>
        <State>FL</State>
        <PostalCode>33122</PostalCode>
        <Country isoCountryCode="US">UnitedStates</Country>
      </PostalAddress>
    </Address>
  </Airport>
</AirLegDestination>

```

Airport

The common Airport element, which contains the three-letter IATA airport code in the airportCode attribute, and an optional Address element, is defined at “Airport” on page 134.

For information on the International Air Transport Association (IATA) standard, see: <http://www.iata.org/codes/index>.

BookingClassCode

The common BookingClassCode element is defined at “[BookingClassCode](#)” on page 134. The BookingClassCode element of an AirLeg defines the class of travel for the AirLeg according to the de-facto airline standard. The following table shows sample IATA codes:

F, FN, P, R, A	first class
C, CN, D, J, I, Z	business class
Y, YN, B, BN, M, H, V, VN, O, Q, QN, S, K, KN, L, U, T, W	coach class

The sample codes are not guaranteed to be accurate or current. For information on the International Air Transport Association (IATA) standard, see: <http://www.iata.org/codes/index>.

Rate

The common Rate element is defined at “[Rate](#)” on page 133. The total of all specified AirLeg rates must equal the line item total.

Meal

The common Meal element, which describes one meal in a travel line item, is defined at “[Meal](#)” on page 135.

AvailablePrice

The optional, common AvailablePrice element, which defines available prices that the user did not select, is defined at “[AvailablePrice](#)” on page 132. The AvailablePrice element of AirDetail defines available price information for a single-leg, multi-leg, or round trip.

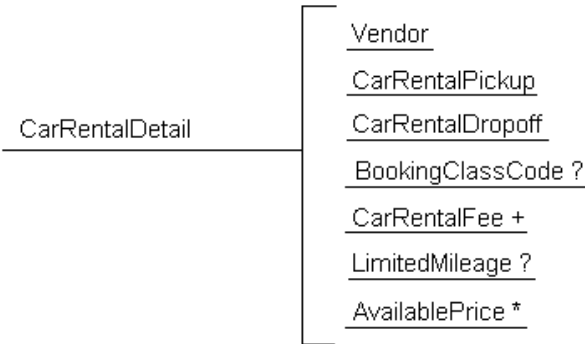
Penalty

The common Penalty element, which describes extra charges assessed by vendors for user changes to travel line items, is defined at “[Penalty](#)” on page 132. The Penalty element of an AirLeg describes extra charges for changes to, or cancellation of, an air travel reservation.

CarRentalDetail

CarRentalDetail provides information about a single car rental event.

The following diagram shows the structure of the CarRentalDetail element.



CarRentalDetail has the following attributes:

travelSegment	Text description used to identify this travel segment. The description is specific to the travel booking provider.
pickupTime	Intended local pickup date and time.
dropoffTime	Intended local drop-off date and time.

Vendor

The common Vendor element, which provides information about the vendor of a service, is defined at “Vendor” on page 130.

CarRentalPickup / CarRentalDropoff

These elements contain the addresses of the CarRentalPickup and CarRentalDropoff entities on the CarRentalDetail. Both CarRentalPickup and CarRentalDropoff require the common Airport element, which specifies the airport location.

BookingClassCode

A four-letter code, which indicates the rental car class. Each buyer-travel booking provider pair can use the standard they choose. For example, a common U.S. standard for car rental:

1st Letter	M (Mini) E (Economy) C (Compact) S (Standard) I (Intermediate) F (Full size) P (Premium) L (Luxury) V (MiniVan) X (Special)
2nd Letter	B (2 door) C (2/4 door) D (4 door) T (Convertible) F (Four wheel drive) V (Van) W (Wagon) S (Sport) X (Special)
3rd Letter	A (Automatic) M (Manual)
4th Letter	R (A/C) N (No A/C)

CarRentalFee

CarRentalFee defines the actual charges and fees that apply to this car rental. To capture the breakdown of various fees, use multiple CarRentalFee elements within one CarRentalDetail element. The total of these fees must add up to the total at the line item level.

Note: Use TermsAndConditions text to specify conditional charges for items such as extra mileage that are over the mileage limit.

CarRentalFee has one attribute:

type	The type of fee, expressed in the form "baseRate".
-------------	--

Possible values for the type of a CarRentalFee are:

Value	Description
additionalDriver	Additional driver fee
airportAccessFee	Airport access fee
baseRate	Base rental rate
childSeat	Child seat charge
collisionDamageInsurance	Collision damage insurance
dropOffCharge	Drop off charge
liabilityInsurance	Liability insurance
luggageRack	Luggage rack charge
mobilePhone	Mobile phone base charge
navigationSystem	Navigation system
other	Other charges
prepaidGasoline	Prepaid gasoline charge
touristTax	Tourist tax
vehicleLicensingFee	Vehicle licensing fee

Total

Total amount for this CarRentalFee. All Rate amounts for a line item must must add up to the Total for that line item.

Rate

Fee information for individual charges for this CarRentalFee.

LimitedMileage

LimitedMileage specifies the quantity and unit of measure of the mileage limit.

LimitedMileage has one attribute:

quantity	The mileage limit amount, expressed as a numeral
-----------------	--

UnitOfMeasure

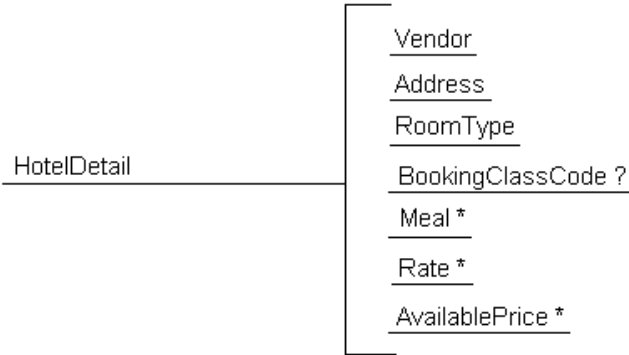
Unit of measure, expressed in miles or kilometers.

AvailablePrice

The optional, common AvailablePrice element, which defines available prices that the user did not select, is defined at “AvailablePrice” on page 132.

HotelDetail

The following diagram shows the structure of the HotelDetail element.



HotelDetail has the following attributes:

travelSegment	Text information to identify this travel segment. This information is specific to the travel booking provider.
arrivalTime	Local date and time of arrival at the hotel. This is used as an advisory to the hotel vendor for the arrival time.
departureTime	Local date and time of departure from the hotel. This is an advisory to the hotel vendor for the departure time.
checkinTime	Local official hotel checkin time.
checkoutTime	Local official hotel checkout time.
earlyCheckinAllowed (optional)	Does the hotel allow early checkin? no, or yes (default).
lateCheckoutAllowed (optional)	Does the hotel allow late checkout? no, or yes (default)

Vendor

The common Vendor element, which provides information about the vendor of a service, is defined at “Vendor” on page 130. For HotelDetail, the Vendor element defines the hotel provider.

Address

Physical address of the hotel. This might be different from the address specified in the Vendor field. The Address in Vendor might be the address of the hotel’s corporate headquarters, for example, while the Address in HotelDetail would be the address of the individual hotel.

RoomType

Information about the type of hotel room reserved.

RoomType has the following attributes:

smoking	Smoking or non-smoking room: yes no
numberOfBed (optional)	The number of beds in the room.
bedType (optional)	The type of bed in the room: king queen full double single other

Description

Text description of the hotel room.

Amenities

Text description of amenities. For example, DSL connection, two telephone lines, and other information about a hotel room.

Amenities has no attributes.

Description

Text description of the amenities. For example, DSL connection, two telephone lines, and other information about the hotel room.

BookingClassCode

The common BookingClassCode element is defined at “[BookingClassCode](#)” on page 134. Each buyer-travel booking provider pair can use any standard they choose.

Meal

The common Meal element is defined at “[Meal](#)” on page 135. The Meal element of HotelDetail defines any complimentary meals that are included with the room, such as complimentary continental breakfast.

Rate

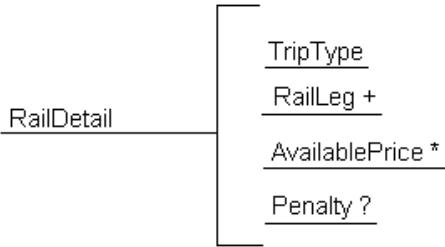
The common Rate element is defined at “[Rate](#)” on page 133. The Rate element of HotelDetail defines one or more rates for the hotel stay. For example, the nightly rate or valet parking rate.

AvailablePrice

The common AvailablePrice element is defined at “[AvailablePrice](#)” on page 132. The AvailablePrice element of HotelDetail defines other available prices that the user did not pick. Available prices can be from the same vendor or another vendor.

RailDetail

The following diagram shows the structure of the RailDetail element.



RailDetail has no attributes.

TripType

TripType is a container for the type attribute, which is required in both AirDetail and RailDetail . The TripType element defines a round trip, one way, or multi-leg trip.

For example, a TripType for a round trip would appear as:

```
<TripType type="round"></TripType>
```

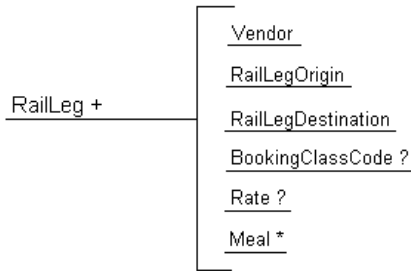
Possible values for the type attribute of TripType:

Value	Description
round	round trip
oneWay	one-way trip
multiLeg	multi-leg or open-jaw trip

RailLeg

One or more RailLeg elements that make up this RailDetail. Each RailDetail must include at least one RailLeg.

The following diagram shows the structure of RailLeg:



RailLeg has the following attributes:

travelSegment	Text information to identify this travel segment. This information is specific to the travel booking provider.
departureTime	Local date and time of departure from the originating location.
arrivalTime	Local date and time of arrival at the destination location.
trainNumber	Train number.
seatNumber	Seat number.
carType	The type of the rail car.

Vendor

The common Vendor element, which provides information about the vendor of a service, is defined at “[Vendor](#)” on page 130. For RailLeg, the Vendor element defines the rail travel provider, such as Amtrak.

RailLegOrigin / RailLegDestination

RailLegOrigin and RailLegDestination have two possible elements, of which exactly one must be included:

Airport

The common Airport element, which contains the three-letter IATA airport code in the airportCode attribute, and an optional Address element, is defined at “[Airport](#)” on page 134.

For information on the International Air Transport Association (IATA) standard, see: <http://www.iata.org/codes/index>.

Address

The physical address of the rail station.

Neither RailLegOrigin nor RailLegDestination has any attributes.

BookingClassCode

The common BookingClassCode element is defined at “[BookingClassCode](#)” on page 134. The BookingClassCode element of a RailLeg element defines the class of travel for the RailLeg according to a rail standard agreed upon by the buyer-travel booking provider pair.

Rate

The common Rate element is defined at “[Rate](#)” on page 133. Rate information for this rail leg. If specified, all the rates in all rail legs must add up to the total at the travel line item level.

Meal

The common Meal element is defined at “[Meal](#)” on page 135. The Meal element of HotelDetail defines any complimentary meals that are included with the room, such as complimentary continental breakfast.

AvailablePrice

The common AvailablePrice element is defined at “AvailablePrice” on page 132. The AvailablePrice element of RailDetail defines other available prices that the user did not pick. Available prices can be from the same vendor or another vendor.

Penalty

The common Penalty element, which describes extra charges assessed by vendors for user changes to travel line items, is defined at “Penalty” on page 132. The Penalty element of RailLeg defines extra charges for changes to, or cancellation of, a rail travel reservation.

FeeDetail

Conveys information about one-time or recurring fees that are not explicitly defined elsewhere in cXML. For example, a one-time fee for furniture rental would not fall into any category defined in TravelDetail or LaborDetail, but could be described in FeeDetail.

FeeDetail has the following attribute:

isRecurring (optional)	Indicates that the fee is recurring.
-----------------------------------	--------------------------------------

UnitRate

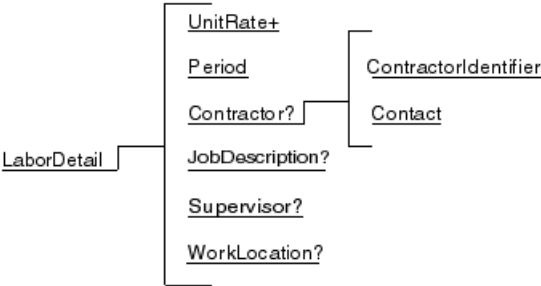
The amount to be paid per unit of time or other measure. In the case of multiple UnitRates, as in a rate schedule, it is customary and expected that each UnitRate will include a TermReference to distinguish it from other UnitRate elements.

Period

Defines the period covered by the FeeDetail.

LaborDetail

LaborDetail contains information about an item related to temporary labor. The following diagram shows the structure of the LaborDetail element:



LaborDetail has the following attribute:

supplierReferenceCode	The supplier's quote or proposal ID, for cross reference.
------------------------------	---

UnitRate

UnitRate represents the amount to be paid per unit of time (or of some other measure). In the case of multiple UnitRates, each UnitRate should include a TermReference to distinguish it from others.

TermReference

TermReference is a generic base element that identifies the definition of the UnitRate in question. TermReference has these attributes:

termName	The name of the ID attribute containing the term.
term	The value of that attribute, that is, the term itself.

Here is a sample UnitRate with a complete TermReference:

```
<UnitRate>
  <Money currency="USD">75</Money>
  <UnitOfMeasure>HUR</UnitOfMeasure>
  <TermReference
    documentURI="https://ABC/MasterAgreement.xml"
    elementType="LaborRate"
```



```
        termName="payCode"
        term="Overtime"
      >
    </UnitRate>
```

In the context of a temporary labor purchase order, where the master agreement is known, and the `elementType` and `termName` are implied by the context, this can be abbreviated as:

```
<UnitRate>
  <Money currency="USD">75</Money>
  <UnitOfMeasure>HUR</UnitOfMeasure>
  <TermReference term="Overtime">
</UnitRate>
```

Period

Period specifies the period of time over which the service occurs.

Contractor

Contractor identifies the contractor being engaged for temporary labor. The contractor is uniquely identified by a `ContractorIdentifier` element, which is exchanged between the buyer and supplier prior to sending orders or timecards. For more information about `TimeCard` transactions, see Chapter 8, “`TimeCard` Transaction.”

ContractorIdentifier

`ContractorIdentifier` uniquely identifies the contractor for both the buyer and supplier. `ContractorIdentifier` has the following attribute:

domain	Required. The domain in which the contractors identity is represented. This attribute allows the buyer and supplier systems to determine who assigned the identification. buyerReferenceID implies that the identification was generated in the buyer system. supplierReferenceID implies that the identification was generated in the supplier system.
--------	---

Contact

Contact contains contact element for the contractor.

JobDescription

`JobDescription` is a text description of the job or work to be performed.

Supervisor

Supervisor specifies contact information for the person who will supervise the contractor.

WorkLocation

WorkLocation is the address of the place where the work is to be performed.

Extrinsic

Extrinsic is supported in SpendDetail, enabling buyer-supplier pairs to convey detailed information on spend that does not fit within TravelDetail, FeeDetail, or LaborDetail.

Extrinsic elements are intended to provide additional machine-readable information. They extend the cXML protocol to support features not required by all implementations. The cXML specification does not define the content of Extrinsic elements. Each buyer-supplier pair must agree on and implement their definitions of Extrinsic elements.

Describes detailed information for any undefined spend category. The name attribute of the Extrinsic element should specify the type of spend category, such as print, market research, or project labor.

It is recommend that all Extrinsic elements in a single SpendDetail element be included under a single Extrinsic with the name attribute used to specify the name of the category. This example shows two Extrinsic elements nested under one heading, within a SpendDetail element:

```
<SpendDetail>
  <Extrinsic name="MarketResearchDetail">
    <Extrinsic name="ResearchObjectives">test objectives</Extrinsic>
    <Extrinsic name="ProjectNumber">PN3434343</Extrinsic>
  </Extrinsic>
</SpendDetail>
```

The Extrinsic element can also appear in the OrderRequestHeader, ItemDetail, and ContractItem elements. These contexts are described further elsewhere in this document. For more information on Extrinsic, see [“Extrinsic”](#) on page 152.

Distribution

Distribution divides the cost of an item among multiple parties. Suppliers return the Distribution element on invoices to facilitate the buyer's reconciliation process.

Accounting

The Accounting element groups AccountingSegments to identify who is charged.

Accounting has the following attribute:

name	The name for this accounting combination. The account from which this charge will be paid.
------	--

AccountingSegment

The AccountingSegment element can contain any relevant accounting code used by a buying organization. Examples of possible values are asset number, billing code, cost center, G/L account, and department. For example:

```
<AccountingSegment id="456">
  <Name xml:lang="en-US">G/L Account</Name>
  <Description xml:lang="en-US">Travel</Description>
</AccountingSegment>
```

AccountingSegment has the following attribute:

id	The unique identifier within this AccountingSegment type. This value might be the actual account code if the type were "Cost Center".
----	---

Name

An identifying name for this AccountingSegment with respect to the others in the Accounting element.

Description

A description of the accounting entity.

Charge

This element specifies the amount to be charged to the entity represented by the Accounting element.

Money

Contains the amount of the Charge at the line item level.

currency	The unique ISO standard three-letter currency code. For example, "USD" = United States Dollar.
----------	--

Response to an OrderRequest

This document is the response part of the synchronous Request-Response transaction. The following example shows a Response to an OrderRequest document:

```
<cXML payloadID="9949494" xml:lang="en"
  timestamp="1999-03-12T18:39:09-08:00">
  <Response>
    <Status code="200" text="OK"/>
  </Response>
</cXML>
```

As shown above, this Response is straightforward. In this case, there is no actual element named "OrderResponse", because the only data that needs to be sent back to the requestor is the Status part of the Response.

The Response tells the requestor its OrderRequest was successfully parsed and acted on by the remote part of HTTP connection. It does not communicate order-level acknowledgement, such as which items can be shipped, or which need to be backordered.

Accepting Order Attachments

Buyers often need to clarify purchase orders with supporting memos, drawings, or faxes. They can attach files of any type to cXML purchase orders by using MIME (Multipurpose Internet Mail Extensions).

cXML contains only references to external MIME parts sent within one multipart MIME envelope (with the cXML document, in an e-mail or faxed together).

Commerce network hubs receive the attachments, and can forward them to suppliers or store them for online retrieval.

For more information about purchase order attachments, see “[Attachments](#)” on page 31.

For more information about the MIME standard, see the following websites:

www.hunnysoft.com/mime

www.rad.com/networks/1995/mime/mime.htm

Chapter 6

Path Routing

In complex relationships between buyers and suppliers, a document might be routed through several intermediary systems before reaching the end supplier. Path Routing enables documents to be routed by and copied to intermediary systems such as marketplaces, and commerce network hubs.

This chapter describes:

- [Overview of Path Routing](#)
- [Nodes](#)
- [Adding Nodes to PunchOutOrderMessage](#)
- [Creating OrderRequests](#)
- [Other Routable Documents](#)
- [CopyRequest](#)

Overview of Path Routing

Path routing is especially useful in direct and indirect marketplaces. In direct marketplaces, suppliers bill buyers directly. In indirect marketplaces, suppliers bill and receive payment from the marketplace host, which in turn bills and receives payment from member buyers.

Direct marketplaces can be PunchOut sites that enable external buyers to access suppliers' PunchOut catalogs. For a marketplace to track transactions originating from it, it must receive copies of all purchase orders as they route to the supplier.

To receive copies of all purchase orders as they route, the marketplace adds itself as a Copy node to the Path of all PunchOutOrderMessage documents sent to the external buyers. This information also allows a marketplace to support edit/inspect PunchOut from procurement applications because it can distinguish which items in the shopping cart come from an external marketplace by inspecting the Path element.

Indirect Marketplaces can receive OrderRequest documents, modify them, split them, and route them to suppliers. Indirect marketplaces are router nodes that create new versions and route OrderRequest documents to suppliers.

To enable path routing in PunchOut:

1. Each system adds itself as a node to the Path element of PunchOutOrderMessage documents sent by suppliers to procurement applications.
2. Procurement applications generate OrderRequest documents by splitting the order based on the Path and SupplierID of each of the ItemIn elements of PunchOutOrderMessage documents. Procurement applications put a Path element at the cXML header level of each OrderRequest document.
3. Subsequent documents, such as OrderRequest, PunchOutSetupRequest, ConfirmationRequest, and ShipNoticeRequest documents are routed and copied by using the Path element at the header level.

Adding a Path element at the item or header level enables copying and routing of cXML documents for marketplaces and commerce network hubs. The Path element records the path taken between the buyer and supplier which documents can later use to find their way back to a supplier.

Nodes

Nodes appear in the Path element of either the header section, or ItemIn and ItemOut elements. Each node in the Path element can be either a router node or a copy node. If the node is of type “copy”, the system simply wants a copy of each document passing through. If the node is of type “route”, the system will modify and re-route each document passing through. Each system in the path must specify which type it is.

Path Element

The Path element contains nodes that are either of type=“copy” or type=“route”. For example, the following contains a copy node and a router node:

```
<Path>
  <Node type="copy">
    <Credential domain="NetworkId">
      <Identity>AN01000000111</Identity>
    </Credential>
  </Node>
```



```
<Node type="route">
  <Credential domain="NetworkId">
    <Identity>AN01000000233</Identity>
  </Credential>
</Node>
</Path>
```

Router Nodes

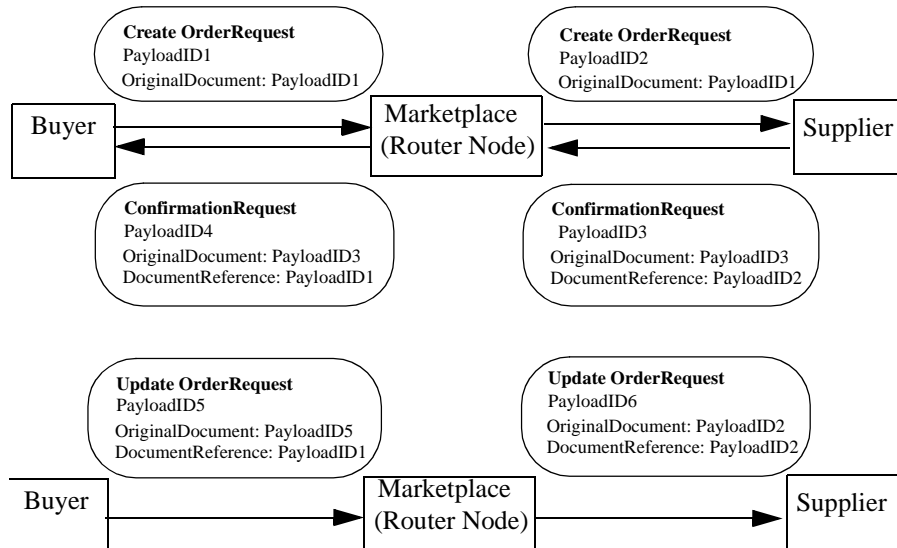
A router node creates a new version of the document it receives and routes it to the next node in the path. The routed document typically changes unit price, bill-to, or ship-to address information.

OriginalDocument Element

The new document must reference the document it is modifying by adding an *OriginalDocument* element, if it is not already present, at the header level that specifies the *payloadID* of the original document. This enables the network hub to keep track of each hop in the *Path* and decide which version of the document to display to the appropriate party.

DocumentReference Element

Each node is responsible for updating any *DocumentReference* elements in the new document it generates. For example, when an *OrderRequest* of type *update* or *delete* is routed to an intermediary node, this node must change the *DocumentReference* in the new version of the updated *OrderRequest* to reference the correct *payloadId* as illustrated in the following diagram:



Copy Nodes

A copy node wants a copy of the document. For example, the following except illustrates a copy node:

```

<Node type="copy">
  <Credential domain="NetworkId">
    <Identity>AN01000000111</Identity>
  </Credential>
</Node>

```

Adding Nodes to PunchOutOrderMessage

PunchOutOrderMessage documents generated by PunchOut sessions can go through intermediary sites on their way back to the buyer. Each intermediary site must add itself as a node to the Path element of the relevant ItemIn elements of the PunchOutOrderMessage.

Node sequence is top to bottom, with the originating buyer at the top. The intermediary node closest to the end supplier must add the supplier of record to the path as well, if the supplier has not already created the path.

The procurement application must include itself as the first router node in the path, which allows other documents such as ConfirmationRequest and ShipmentNoticeRequest documents to be routed back to the originating buyer.

Path Element

The Path element contains nodes that are either of type="copy" or type="route". A Path element is in each ItemIn element of a PunchOutOrderMessage. Each system visited by the PunchOutOrderMessage must add itself as a node to the Path element for each ItemIn element it cares about.

The following PunchOutOrderMessage shows the Path element with two nodes:

```
<ItemIn quantity="1">
  <ItemID>
    <SupplierPartID>1234</SupplierPartID>
  </ItemID>
  <Path>
    <Node type="copy">
      <Credential domain="NetworkId">
        <Identity>AN01000000111</Identity>
      </Credential>
    </Node>
    <Node type="route">
      <Credential domain="NetworkId">
        <Identity>AN01000000233</Identity>
      </Credential>
    </Node>
  </Path>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">10.23</Money>
    </UnitPrice>
    <Description xml:lang="en">Learn ASP in a Week!</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="SPSC">12345</Classification>
    <ManufacturerPartID>ISBN-23455634</ManufacturerPartID>
    <ManufacturerName>O'Reilly</ManufacturerName>
  </ItemDetail>
</ItemIn>
```

Credentials

The From and To elements of the cXML header in a routed document refer to the buyer and supplier of record. Neither of these parties is required to appear in the Path, because they might be visible only to one of the Router nodes.

Creating OrderRequests

When generating purchase orders, procurement applications split requisitions based on the Path and SupplierID of each of the ItemIn elements.

Path Element

Procurement applications put Path elements in the cXML header level of each of the orders. Procurement applications should not include the identical Path element in any of the ItemOut elements in an OrderRequest.

In OrderRequest documents containing PunchOut items, procurement applications must include nodes for both the originating buyer and the supplier of record.

Credentials

Because commerce network hubs are responsible for routing OrderRequest documents to the next node in the path, the Sender credential is always the network hub credential when received by the next node. The preceding node (most recent originator) can always be found by examining the From Credential list or, the Path for the most recent Router node if the Router node doesn't modify the From element. In addition, the type="marketplace" credential must be one of the router nodes in the path. A From credential list with no type="marketplace" credential implies that the identical node is the originating procurement application.

The following example is the header of an OrderRequest sent from a procurement application. Because the From credential has no type="marketplace", the node sending this OrderRequest must be the procurement application. The first node in the path is a marketplace Router node.

```
<Header>
  <From>
    <Credential domain="AribaNetworkUserId">
      <Identity>admin@acme.com</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="NetworkId" type="marketplace">
      <Identity>AN01000000233</Identity>
    </Credential>
    <Credential domain="DUNS">
      <Identity>942888711</Identity>
    </Credential>
  </To>
</Sender>
```

```

    <Credential domain="NetworkId">
      <Identity>AN01000000001</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>
    <UserAgent>Ariba SN</UserAgent>
  </Sender>
  <Path>
    <Node type="route">
      <Credential domain="AribaNetworkUserId">
        <Identity>admin@acme.com</Identity>
      </Credential>
    </Node>
    <Node type="copy">
      <Credential domain="NetworkId">
        <Identity>AN01000000111</Identity>
      </Credential>
    </Node>
    <Node type="route">
      <Credential domain="NetworkId">
        <Identity>AN01000000233</Identity>
      </Credential>
    </Node>
  </Path>
  <OriginalDocument payloadID="pay1"/>
</Header>

```

The following example is an OrderRequest from a marketplace Router node:

```

<Header>
  <From>
    <Credential domain="AribaNetworkUserId">
      <Identity>admin@acme.com</Identity>
    </Credential>
    <Credential domain="NetworkId" type="marketplace">
      <Identity>AN01000000233</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="NetworkId" type="marketplace">
      <Identity>AN01000000233</Identity>
    </Credential>
    <Credential domain="DUNS">
      <Identity>942888711</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkId">
      <Identity>AN01000000001</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>

```

```

        <UserAgent>Ariba SN</UserAgent>
    </Sender>
    <Path>
        <Node type="route">
            <Credential domain="AribaNetworkUserId">
                <Identity>admin@acme.com</Identity>
            </Credential>
        </Node>
        <Node type="copy">
            <Credential domain="NetworkId">
                <Identity>AN01000000111</Identity>
            </Credential>
        </Node>
        <Node type="route">
            <Credential domain="NetworkId">
                <Identity>AN01000000233</Identity>
            </Credential>
        </Node>
    </Path>
    <OriginalDocument payloadID="pay1"/>
</Header>

```

Other Routable Documents

Follow-up documents such as `PunchOutSetupRequest`, `ConfirmationRequest`, and `ShipNoticeRequest` documents also use the `Path` element to route and copy documents.

PunchOutSetupRequest

Procurement applications must include the same path information in the `ItemOut` elements for any subsequent edit or inspect `PunchOut` sessions.

Procurement applications must not perform any item grouping according to the `Path` element during `PunchOut` sessions.

ConfirmationRequest and ShipNoticeRequest

Route `ConfirmationRequest` and `ShipNoticeRequest` documents by using the `Path` element from the cXML header of the `OrderRequest`. The `Path` must be reversed to route the `ConfirmationRequest` or `ShipNoticeRequest` to the originating application.

CopyRequest

Organizations that want to receive copies of purchase orders, but that are not the primary recipients, are called *copy organizations*. They receive copies of purchase orders as cXML documents within CopyRequest attachments sent by commerce network hubs.

Copy organizations must add the CopyRequest transaction to their cXML profile. When the commerce network hub receives a purchase order containing path routing copy information, it first looks up the copy organization's CopyRequest URL in the organization's cXML profile. It then sends the attached document to the copy organization.

Note that the use of CopyRequest attachments differs from previous implementations of CopyRequest, in which cXML documents were contained as internal elements within CopyRequest/cXML. In cXML 1.2.011, the use of the cXML element as a child of CopyRequest is deprecated. Instead, use the cXMLAttachment element to attach another cXML document, whether or not it contains attachments itself. For more information about attachments, see [“Attachments”](#) on page 31.

The following example shows a CopyRequest element forwarding a cXML document that does not itself contain attachments:

```
Content-Type: Multipart/Related; boundary=mime-boundary
[Other headers]

--mime-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/1.2.011/cXML.dtd">
<cXML version="1.0" 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>password</SharedSecret>
  </Credential>
  <UserAgent>Sender Application</UserAgent>
</Sender>
</Header>
<Request deploymentMode="test">
  <CopyRequest>
    <cXMLAttachment>
      <Attachment>
        <URL>cid:222@sendercompany.com</URL>
      </Attachment>
    </cXMLAttachment>
  </CopyRequest>
</Request>
</cXML>

--mime-boundary
Content-Type: text/xml; charset=UTF-8
Content-ID: <222@sendercompany.com>
[Other headers]

[Forwarded cXML]

--mime-boundary--

```

For an example showing a CopyRequest forwarding a cXML document with attachments, see [“Attachment Examples”](#) on page 32.

Chapter 7

Payment

Buying organizations use cXML payment documents to pay suppliers for provided products or services. cXML payment provides immediate access to payment scheduling information, allowing more accurate forecasting and scheduling of payables and receivables. This chapter introduces the cXML transactions that comprise payment.

This chapter describes:

- [Overview of Payment](#)
- [PaymentRemittanceRequest](#)
- [PaymentRemittanceStatusUpdateRequest](#)
- [Payment Document Examples](#)

Overview of Payment

cXML payment automate the payment process by using electronic documents to represent payments and remittance advice documents, and to track and process payments. The cXML payment process includes payment schedules (plans), discounts, creating and sending payments (regardless of where payments are made), and ensuring that payments have been received.

The PaymentRemittance transaction supports payment transaction details for a wide variety of business scenarios, including standard invoices, credit memos, and debit memos.

When a payment is made, the organization making the payment also creates an associated remittance advice document. Remittance advice is a summary statement that provides details about payments that have been made. A typical remittance advice statement includes the payment method used, bank information, discount amount, amount paid, and a list of payables included in the payment.

cXML provides full support for exchanging payment documents electronically, regardless of where actual payments are made.

PaymentRemittance DTD

The cXML standard uses multiple DTDs to optimize the performance of validating parsers. The payment transactions described in this chapter are defined in a separate DTD named `PaymentRemittance.dtd`, available at:

<http://xml.cXML.org/schemas/cXML/<version>/PaymentRemittance.dtd>

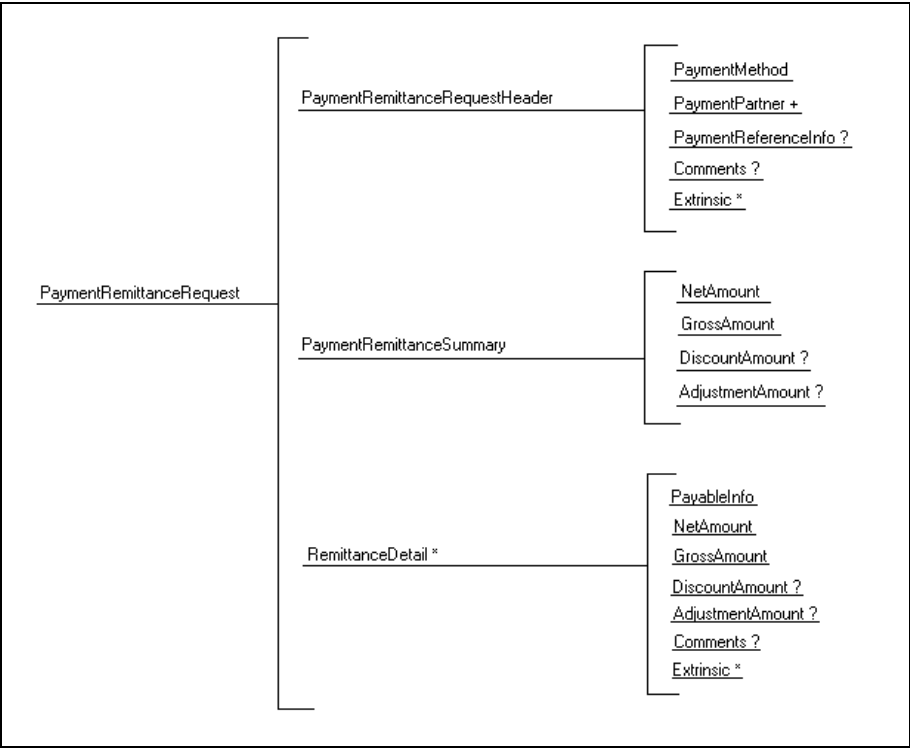
Working with Payment Documents

There are two types of cXML documents used in the payment process. Procurement applications send `PaymentRemittanceRequest` documents, and suppliers respond with generic `Response` documents. When payment transaction status levels are updated, procurement applications send `PaymentRemittanceStatusUpdateRequest` documents. All these documents can pass through a network commerce hub for authentication and routing.

PaymentRemittanceRequest

The `PaymentRemittanceRequest` document is analogous to remittance detail advice for for payment or remittance.

The following diagram shows the structure of the PaymentRemittanceRequest element:



The following example shows the structure of the PaymentRemittanceRequest element:

```
<PaymentRemittanceRequest>
  <PaymentRemittanceRequestHeader>
    header information
  </PaymentRemittanceRequestHeader>
  <PaymentRemittanceSummary>
    summary-level remittance information
  </PaymentRemittanceSummary>
  <RemittanceDetail>
    detail-level remittance information
  </RemittanceDetail>
</PaymentRemittanceRequest>
```

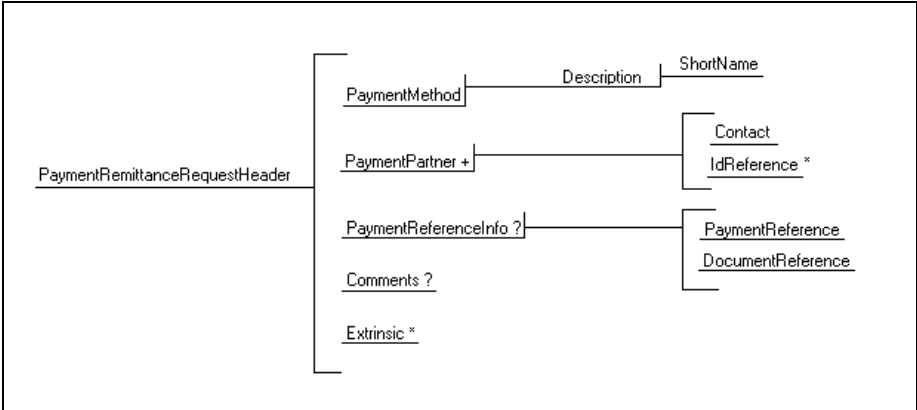
PaymentRemittanceRequest has no attributes.

For an example of a PaymentRemittanceRequest for an invoice, see [“PaymentRemittanceRequest Example”](#) on page 182.

PaymentRemittanceRequestHeader

The PaymentRemittanceRequestHeader element defines header information that applies to the entire payment or remittance.

The following diagram shows the structure of the PaymentRemittanceRequestHeader element:



PaymentRemittanceRequestHeader has the following attributes:

paymentRemittanceID	A unique identifier for this PaymentRemittance, generated by the buying organization's system.
paymentDate	The date and time this Payment or Remittance transaction was created. paymentDate should be earlier than the timestamp of the actual PaymentRemittanceRequest.
isPayment	Indicates whether this request is intended for making payment or is for remittance advice only. If the request is for payment purposes, set the attribute to 'yes'. Remittance advice information can be included in a PaymentRemittanceRequest with isPayment = yes. If isPayment is not specified, the document is assumed to be for remittance advice only.
paymentReferenceNumber	Indicates a payment transaction reference or payment identification number. For example, for check payments, the paymentReferenceNumber is the check number, and for electronic payments, it is an electronic reference or confirmation number.

PaymentMethod

Buying organizations use this element to identify the method for a payment.

PaymentMethod has one attribute:

type	The type of the payment method
-------------	--------------------------------

Possible values of the type attribute of PaymentMethod:

Value	Description
ach	Automated Clearing House
cash	Cash payment
check	Check payment
creditCard	Credit card or PCard payment
debitCard	Debit card payment
draft	A written payment order, directing a second party to pay a third party
wire	Wire transfer
other	Other, not defined in cXML

Description

The description of the payment method. Description is mandatory if the type is set to “other”. The ShortName element in Description must indicate the name of the payment method.

ShortName

A short string describing something in fewer characters than the entire Description. Use the ShortName element when limited space is available. For example, a table of elements might show the ShortName of each. A linked “details” view would show the entire Description (including the ShortName). Without a ShortName, the user interface must default to a truncation of the Description.

This element does not require an xml:lang attribute because it appears only within a Description element. The language of the ShortName must match that of the surrounding Description.

PaymentPartner

Defines the parties involved in the payment. PaymentPartner has no attributes.

Contact

Contact information of the payment partner.

Contact has one attribute:

role	The role of the payment partner
-------------	---------------------------------

Contact elements with role payer and payee are always required. If the payment method indicates a bank transfer, then Contact elements with role originatingBank and receivingBank are required.

Possible values for the role attribute of Contact in a PaymentPartner element:

Value	Description
payer	The payer of this transaction.
payee	The recipient of the payment.
originatingBank	The bank from which the payment will be drawn.
receivingBank	The bank to which the payment will be deposited.
originatingCorrespondentBank (optional)	The bank that will hold the payment and transfer it to the receiving bank or the receiving correspondent bank.
receivingCorrespondentBank (optional)	The bank that will receive the payment and transfer it to the receiving bank.
intermediaryBank (optional)	The intermediary bank.

IdReference

Contains a unique identification reference for the payment partner, including information such as bank account identification, bank identification, and optional bank branch identification.

IdReference is mandatory for all transactions that involve electronic payments. It is optional only for non-electronic payment methods, such as check or cash.

IdReference has the following attributes:

identifier	The unique identifier of the IdReference within the domain
domain	The domain of the IdReference

Possible values of the domain attribute of IdReference are:

Value	Description
bankRoutingID	The routing ID of this payment partner's bank.
accountReceivableID	The ID of the payee's accounts receivable account or department.
bankAccountID	The ID of this payment partner's bank account.
ibanID	The International Bank Account Number for this payment partner, as specified in ISO 13616.
abaRoutingNumber	The American Banking Association 9-digit routing transit number of this payment partner's bank.
bankNationalID	A national clearing code that is specific to a country. This should uniquely identify the bank within the country specified in the Contact.
isoBicID	ISO BIC ID (Bank Identifier Code) as specified in ISO 9362. The Bank Identifier Code (BIC) is a universal method of identifying financial institutions. The BIC consists of 8 or 11 characters, comprising a bank code (4 characters), a country code (2 characters), a location code (2 characters) and an optional branch code (3 characters).
swiftID	SWIFT ID (Society for Worldwide Interbank Financial Telecommunications) identification number.
bankBranchID	The identification number of the bank branch.

The value supplierTaxID is deprecated and will be treated as federalTaxID.

The bank account identification is specified as follows:

identifier	The unique identifier of the IdReference within the domain
domain	The domain of the IdReference

The bank identification is specified as follows:

Value	Description
abaRoutingNumber	ABA (American Banking Association) routing number.
swiftID	SWIFT ID (Society for Worldwide Interbank Financial Telecommunications) identification number.
chipsID	CHIPS ID (Clearing House Interbank Payment System) identification number.
isoBicID	ISO BIC ID (Bank Identifier Code) as specified in ISO 9362. The Bank Identifier Code (BIC) is a universal method of identifying financial institutions. The BIC consists of 8 or 11 characters, comprising a bank code (4 characters), a country code (2 characters), a location code (2 characters) and an optional branch code (3 characters).
bankNationalID	If none of the above bank identification methods is applicable, then use the bankNationalID to indicate national clearing codes that are specific to a country. This should uniquely identify the bank within the country specified in the Contact.

The bank branch identification, if necessary, is specified as follows:

Value	Description
bankBranchID	The bank branch ID

The following table illustrates some valid role - domain value combinations for Contact and IdReference:

Contact@role	IdReference@domain
payer	bankAccountID ibanID
payee	bankAccountID ibanID
originatingBank	abaRoutingNumber bankNationalID isoBicID swiftID bankBranchID (optional)

receivingBank	abaRoutingNumber bankNationalID isoBicID swiftID bankBranchID (optional)
originatingCorrespondentBank	abaRoutingNumber isoBicID swiftID
receivingCorrespondentBank	abaRoutingNumber isoBicID swiftID
intermediaryBank	abaRoutingNumber isoBicID swiftID

Creator

The creator of this IdReference, such as United Parcel Service, Bank of America, or Cisco Systems.

Description

Text description of the IdReference. This is especially useful when the Creator value is not immediately understood by a reader who is investigating a problem.

PaymentReferenceInfo

Defines the ID of an earlier payment made by a buying organization. This ID should uniquely identify the payment made in the buyer system.

PaymentReferenceInfo has no attributes.

PaymentReference

Reference to an earlier PaymentRemittanceRequest. If the earlier payment was made through cXML, this element is required.

PaymentReference has the following attributes:

paymentRemittanceID	The paymentRemittanceID of the request. Note: Do not use the transaction identification number, such as check number.
paymentDate	The payment date.

DocumentReference

The DocumentReference element of a PaymentReference is a container for the payloadID attribute, which refers to a prior PaymentRemittanceRequest.

payloadID	A unique identifier for the prior PaymentRemittanceRequest. The payloadID is copied directly from the cXML element of the PaymentRemittanceRequest.
------------------	---

PaymentIDInfo

The PaymentIDInfo of a PaymentReference refers to the unique identifier for this payment in the buying organization's system. PaymentIDInfo is a container for paymentRemittanceID and paymentDate attributes.

PaymentIDInfo has the following attributes:

paymentRemittanceID	The paymentRemittanceID of the request. Note: Do not use the transaction identification number, such as check number.
paymentDate	The payment date.

Comments

Header-level textual comments about the payment remittance, for the PaymentRemittanceRequestHeader.

Extrinsic

Additional information related to this payment. Information in the Extrinsic element of PaymentRemittanceRequestHeader should not duplicate information in the PaymentRemittanceRequest.

PaymentRemittanceSummary

The PaymentRemittanceSummary element defines summary information of a PaymentRemittanceRequest. Each money amount in a PaymentRemittanceSummary element is expressed as a flat amount with currency.

PaymentRemittanceSummary has no attributes.

NetAmount

The NetAmount element defines the total net payment amount. NetAmount should satisfy the following equation:

$$\text{NetAmount} = \text{GrossAmount} - \text{DiscountAmount} - \text{AdjustmentAmount}$$

GrossAmount

The total gross amount.

DiscountAmount

The total discount amount.

AdjustmentAmount

The total adjustment amount.

RemittanceDetail

The RemittanceDetail element defines the remittance detail of a specific payable that has been paid. Each money amount in a RemittanceDetail element is expressed as a flat amount with currency.

RemittanceDetail has one attribute:

lineNumber	The line number of the associated payable
-------------------	---

PayableInfo

The reference information of the payable, such as an invoice, order, or master agreement. PayableInfo is known to both buyer and supplier. For example, the PayableInfo for an invoice would be the PayableInvoiceInfo.

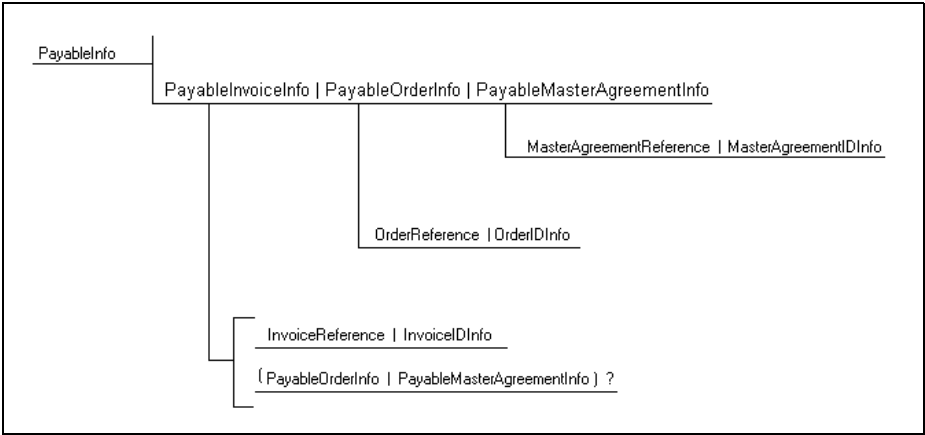
The following example shows the structure of a minimum valid PayableInfo element:

```
<Request>
  <PayableInfo>
    <PayableInvoiceInfo>
      <InvoiceIDReference or InvoiceIDInfo>
        .....
      </InvoiceIDReference or InvoiceIDInfo>
    </PayableInvoiceInfo>
  </PayableInfo>
</Request>
```

The following example shows the structure of a PayableInfo element that includes an optional PayableOrderInfo:

```
<Request>
  <PayableInfo>
    <PayableInvoiceInfo>
      <InvoiceIDReference or InvoiceIDInfo>
        <PayableOrderInfo>
          <OrderIDInfo>
            .....
          </OrderIDInfo>
        </PayableOrderInfo>
      </InvoiceIDReference or InvoiceIDInfo>
    </PayableInvoiceInfo>
  </PayableInfo>
</Request>
```

The following diagram shows the structure of PayableInfo:



PayableInfo has no attributes.

PayableInvoiceInfo

The information about an invoice being paid. PayableInvoiceInfo must contain either InvoiceReference or InvoiceIDInfo, and might contain either PayableOrderInfo or PayableMasterAgreementInfo.

InvoiceReference

Provides a clear reference to a prior InvoiceDetailRequest document. The InvoiceReference is copied from the InvoiceDetailRequest message.

InvoiceIDInfo

Defines the ID of an invoice known to the supplier system. InvoiceIDInfo is a container for two attributes:

invoiceID	The ID of an invoice known to the supplier system
invoiceDate	The invoice date

PayableOrderInfo

Provides supplementary information related to the order. For example, a payment against a consolidated invoice might include associated order information. Defines payable information related to an order that was paid.

PayableOrderInfo has no attributes.

OrderReference

The reference to the order being paid.

OrderIDInfo

The order ID from the procurement application.

PayableMasterAgreementInfo

Provides supplementary information related to the master agreement (contract). For example, a payment against a consolidated invoice might include associated master agreement information. Defines payable information related to the master agreement being paid.

NetAmount

The detail-level net amount for this payable:

$\text{NetAmount} = \text{GrossAmount} - \text{DiscountAmount} - \text{AdjustmentAmount}$

GrossAmount

The detail-level gross payment amount for this payable.

DiscountAmount

Defines the detail-level discount information for this payable.

AdjustmentAmount

The total of various adjustment amounts for this payable, if any. The adjustment amount can be positive, indicating a decrease in payment amount, or negative, indicating an increase in payment amount. For example, a negative AdjustmentAmount might be used to account for late charges or other penalties.

PaymentRemittanceStatusUpdateRequest

The PaymentRemittanceStatusUpdateRequest document provides status information for a payment remittance. Buying organizations send PaymentRemittanceStatusUpdateRequest documents to suppliers to inform suppliers of the status of their payables.

The following example shows the structure of the PaymentRemittanceStatusUpdateRequest element:

```
<Request>
  <PaymentRemittanceStatusUpdateRequest>
    <DocumentReference>
      ....
    </DocumentReference>
    <PaymentRemittanceStatus>
      ....
    </PaymentRemittanceStatus>
  </PaymentRemittanceStatusUpdateRequest>
</Request>
```

DocumentReference

The DocumentReference element is a container for payloadID, which associates a status update with a particular PaymentRemittanceRequest document. DocumentReference repeats a required attribute of the earlier document and adds one optional identifier generated by the supplier. For example:

```
<DocumentReference payloadID="0c300508b7863dccb_14999"/>
```

DocumentReference contains no elements, but has the following attribute:

payloadID	A unique identifier for the document. Copied directly from the cXML element of the previous PaymentRemittanceRequest.
------------------	---

PaymentRemittanceStatus

Defines the status for a payment transaction specified by an existing PaymentRemittanceRequest. PaymentRemittanceStatus has the following attributes:

type	The status type of the payment transaction.
paymentReferenceNumber	Indicates a unique number for a payment. For a check payment, for example, the paymentReferenceNumber would be the check number.

Possible values of the type attribute in PaymentRemittanceStatus are:

Value	Description
paid	The payment transaction was completed successfully.
failed	The payment transaction failed. Under certain conditions, a PaymentRemittance of type "failed" can be resubmitted by the buying organization.
canceled	The payment transaction was canceled.

For a discussion of the common Status element, see [“Status”](#) on page 45.

PaymentRemittanceStatusDetail

Defines status details of the payment transaction specified by an existing PaymentRemittanceStatusDetail. PaymentRemittanceStatusDetail contains a PCDATA string. Typically, this element describes the specifics of a problem.

PaymentRemittanceStatusDetail has the following attributes:

code	Payment transaction status code provided by the payment provider.
description	Textual description of the status code (not specific issue).
xml:lang	The language in which the text attribute and element content are written.

Payment Document Examples

The following examples illustrate payment documents:

- [PaymentRemittanceRequest Example](#)
- [PaymentRemittanceStatusUpdateRequest Example](#)

PaymentRemittanceRequest Example

This example shows a minimum valid PaymentRemittanceRequest.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/
PaymentRemittance.dtd">
<cXML xml:lang="en-US" timestamp="2004-03-10T14:20:53-08:00" payloadID="PR-031004-01">
<Header>
  <From>
    <Credential domain="NetworkId">
      <Identity>AN99123456789</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="NetworkId">
      <Identity>AN99987654321</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkId">
      <Identity>AN99123456789</Identity>
    </Credential>
    <UserAgent>Procurement Application 1.0</UserAgent>
  </Sender>
</Header>
  <Request deploymentMode="production">
    <PaymentRemittanceRequest>
      <PaymentRemittanceRequestHeader paymentDate="2004-10-10T00:00:00-08:00">
```



```

08:00"
paymentReferenceNumber="ACH123456789" paymentRemittanceID=
"PR-031204-01">
<PaymentMethod type="ach"></PaymentMethod>
<PaymentPartner>
  <Contact role="payer">
    <Name xml:lang="en">buyer</Name>
    <PostalAddress>
      <Street>100 1st Street</Street>
      <City>Anywhere</City>
      <State>CA</State>
      <PostalCode>94089</PostalCode>
      <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
  </Contact>
</PaymentPartner>
<PaymentPartner>
  <Contact role="payee">
    <Name xml:lang="en">Supplier</Name>
    <PostalAddress>
      <Street>100 Main Street</Street>
      <City>Anywhere</City>
      <State>CA</State>
      <PostalCode>94089</PostalCode>
      <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
  </Contact>
</PaymentPartner>
<PaymentPartner>
  <Contact role="originatingBank">
    <Name xml:lang="en">Moose Credit Union</Name>
    <PostalAddress>
      <Street>100 Elk Drive</Street>
      <City>Mooseville</City>
      <State>CA</State>
      <PostalCode>94087</PostalCode>
      <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
  </Contact>
  <IdReference domain="abaRoutingNumber" identifier="234567890">
  </IdReference>
</PaymentPartner>
<PaymentPartner>
  <Contact role="receivingBank">
    <Name xml:lang="en">Gold Rush Bank</Name>
    <PostalAddress>
      <Street>100 Bret Harte Road</Street>
      <City>Gold Rush</City>
      <State>CA</State>
      <PostalCode>97123</PostalCode>

```

```

        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Contact>
    <IdReference domain="abaRoutingNumber" identifier="678902345">
    </IdReference>
  </PaymentPartner>
</PaymentRemittanceRequestHeader>
<PaymentRemittanceSummary>
  <NetAmount>
    <Money currency="USD">2.00</Money>
  </NetAmount>
  <GrossAmount>
    <Money currency="USD">2.85</Money>
  </GrossAmount>
  <DiscountAmount>
    <Money currency="USD">0.35</Money>
  </DiscountAmount>
  <AdjustmentAmount>
    <Money currency="USD">0.50</Money>
  </AdjustmentAmount>
</PaymentRemittanceSummary>
<RemittanceDetail lineNumber="1">
  <PayableInfo>
    <PayableInvoiceInfo>
      <InvoiceIDInfo invoiceID="INV-031204-01">
      </InvoiceIDInfo>
      <PayableOrderInfo>
        <OrderIDInfo orderID="P0-031204-01">
        </OrderIDInfo>
      </PayableOrderInfo>
    </PayableInvoiceInfo>
  </PayableInfo>
  <NetAmount>
    <Money currency="USD">2.00</Money>
  </NetAmount>
  <GrossAmount>
    <Money currency="USD">2.85</Money>
  </GrossAmount>
  <DiscountAmount>
    <Money currency="USD">0.35</Money>
  </DiscountAmount>
  <AdjustmentAmount>
    <Money currency="USD">0.50</Money>
  </AdjustmentAmount>
</RemittanceDetail>
</PaymentRemittanceRequest>
</Request>
</cXML>

```

PaymentRemittanceStatusUpdateRequest Example

This example shows a PaymentRemittanceStatusUpdateRequest, sent from a buyer to a supplier:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/
PaymentRemittance.dtd">
<cXML payloadID="1068173501644--6417095366782271471@10.10.13.124" timestamp="2003-04-
20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="NetworkId">
        <Identity>AN99123456789</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkId">
        <Identity>AN99987654321</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkId">
        <Identity>Procurement Application 1.0</Identity>
      </Credential>
    </Sender>
  </Header>
  <Request deploymentMode="production">
    <PaymentRemittanceStatusUpdateRequest>
      <DocumentReference payloadID="1234567890123--
1234567890123456789@10.10.10.100">
        </DocumentReference>
      <PaymentRemittanceStatus type="canceled"
paymentReferenceNumber="PaymentRefNumber">1234</
PaymentRemittanceStatus>
    </PaymentRemittanceStatusUpdateRequest>
  </Request>
</cXML>
```

Chapter 8

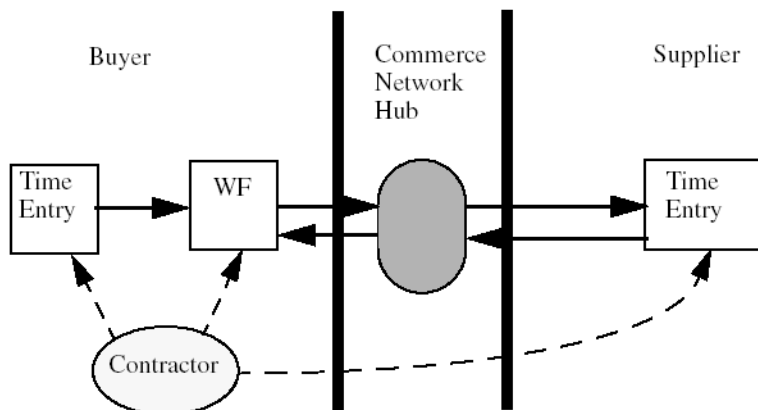
TimeCard Transaction

Timecards are used for placing orders related to temporary labor and contractors. They can be generated and sent by either the buyer or the supplier, depending upon which system captures the timecard information.

TimeCard Requests

Because of the two-way nature of timecards, there are two requests that involve the TimeCard element: TimeCardRequest and TimeCardInfoRequest.

The contractor—that is, the temporary laborer in question—enters timecard information in either the buyer or supplier system, depending upon the situation. Therefore, either buyers or suppliers can send timecard documents, and timecard documents can flow in either direction. In this way, timecards differ from invoices, which are typically sent only by suppliers.



Supplier to Buyer Request

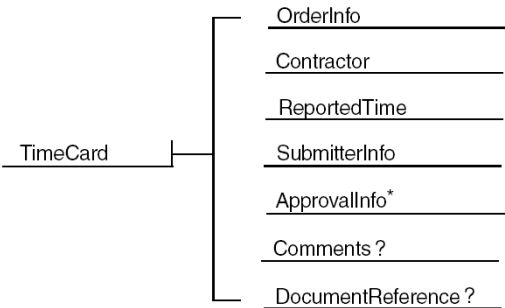
TimeCardRequest describes a timecard document that is sent from a supplier, such as a staffing agency, to a buyer. The from and sender credentials are the supplier's, and the to credential is the buyer's. When the timecard is approved, the buyer sends a StatusUpdateRequest with the DocumentApprovalStatus element indicating whether the timecard was approved or rejected.

Buyer to Supplier Request

TimeCardInfoRequest describes a timecard document that is sent from a buyer to a supplier. The from credential is the buyer's, and to credential is the supplier's.

TimeCard Element

The TimeCard element is used to capture the hours worked by a contractor or other temporary laborer, and has the following structure:



The TimeCard element has the following attributes:.

type	Possible values are new, update, and delete. The value defaults to new, unless the original timecard is updated.
status	Possible values are submitted, approved, denied. The default value is submitted.
timeCardID	Represents the unique identifier for this timecard in the buyer and supplier systems. Required.

OrderInfo

The OrderInfo element is used to reference the order. One timecard can reference only one order.

Contractor

The Contractor element is the definition of a contractor used in the context of temporary labor.

ContractorIdentifier

ContractorIdentifier uniquely identifies the contractor in both the buyer and supplier systems, and is agreed upon by the buyer and the supplier prior to sending out orders or timecards. The ContractorIdentifier element contains the following attribute:

domain	The domain in which the ContractorIdentifier is represented. Possible values are supplierReferenceID or buyerReferenceID, indicating the system in which the ContractorIdentifier originated.
--------	---

Contact

The generic Contact element describes the contractor.

ReportedTime

The ReportedTime element captures the line items for the timecard.

Period

Period denotes the period of time for which the timecard is being submitted.

TimeCardInterval

The TimeCardInterval element represents the time interval being reported on a timecard. It contains the following attributes:

duration	<p>The duration of time being claimed for the line item, represented in the ISO 8601 format PnYn MnDTnH nMnS, where nY represents the number of years, nM the number of months, nD the number of days, T the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds. For example, to indicate a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes, one would write: P1Y2M3DT10H30M.</p> <p>In the event that duration and TimeRange do not agree, duration takes precedence. For example, if duration is 2 hours, and TimeRange is from 4:00 p.m. to 8:00 p.m., then the 2 hour duration takes precedence. However, if duration is not present, then it is computed from the TimeRange.</p>
payCode	<p>The pay code to be used. Recommended pay codes include:</p> <ul style="list-style-type: none"> Regular Overtime Doubletime Mealbreak Triplettime WeeklyRestDay HolidayWorked RegularNightShift OvertimeNightShift DoubletimeNightShift TripletNightShift WeeklyRestDayNightShift RegularMixedShift OvertimeMixedShift DoubletimeMixedShift TriplettimeMixedShift WeeklyRestDayMixedShift
isNonBillable	<p>Implied attribute that designates whether or not the time is billable. The default behavior is billable.</p>

TimeRange

The TimeRange element defines a time range in which the start and end dates can be unbounded.

The TimeRange element contains the following attributes:

startDate	The first date in the billable period. Required.
endDate	The last date in the billable period. Optional, because the difference in time between the endDate and startDate attributes need not agree with the duration attribute in the TimeCardInterval element.

SubmitterInfo

The SubmitterInfo element contains information about the person submitting the timecard.

The SubmitterInfo element contains the following attributes:

Contact

If the Contact element is absent, then it is assumed that the contractor is also the submitter.

submittedDate	The time when the timecard was submitted. Required.
----------------------	---

ApprovalInfo

The ApprovalInfo element includes information about the approver of the timecard. This information is sent by the supplier for informational purposes only, and can include all the approvers in the chain. There can be multiple approvals because many people might need to approve the timecard in question.

The ApprovalInfo element has the following attributes:

approvedDate	The time when the timecard was approved. Required.
---------------------	--

DocumentReference

DocumentReference is used on an update operation to refer to a previous TimeCardRequest or TimeCardInfoRequest.

TimeCard Examples

The following example shows a TimeCardInfoRequest sent upon submission to the supplier:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/Fulfill.dtd">
<cXML xml:lang="en-US"
  payloadID=" tc1@buyer.com"
  timestamp="2003-10-01T23:00:06-08:00">
  <Header>
    <From>
      <Credential domain="NetworkId">
        <Identity>AN0100023456</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkId">
        <Identity> AN0100023457</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkId">
        <Identity> AN0100023456</Identity>
        <SharedSecret>coyote</SharedSecret>
      </Credential>
      <UserAgent>Ariba Buyer</UserAgent>
    </Sender>
  </Header>
  <Request>
    <TimeCardInfoRequest>
      <TimeCard type="new" status="submitted" timeCardID="TC101">
        <OrderInfo>
          <OrderIDInfo orderID="PO12" orderDate="2003-07-22T08:00:00-08:00"/>
        </OrderInfo>
        <Contractor>
          <ContractorIdentifier domain="supplierReferenceID">Doe8610
          </ContractorIdentifier>
          <Contact>
            <Name xml:lang="en">John Doe</Name>
          </Contact>
        </Contractor>
        <ReportedTime>
          <Period startDate="2003-09-22T08:00:00-08:00"
            endDate="2003-09-26T18:00:00-08:00"/>
          <TimeCardTimeInterval duration="PT8H" payCode="Regular">
            <TimeRange startDate="2003-09-22T08:00:00-08:00"
              endDate="2003-09-22T18:00:00-08:00"/>
          </TimeCardTimeInterval>
        </ReportedTime>
      </TimeCard>
    </TimeCardInfoRequest>
  </Request>
</cXML>
```

```

    <TimeCardTimeInterval duration="PT2H"
    payCode="Mealbreak" isNonBillable="yes">
      <TimeRange startDate="2003-09-22T012:00:00-08:00"
      endDate="2003-09-22T14:00:00-08:00"/>
    </TimeCardTimeInterval>
    <TimeCardTimeInterval duration="PT2H" payCode="Overtime" >
      <TimeRange startDate="2003-09-22T18:00:00-08:00"
      endDate="2003-09-22T20:00:00-08:00"/>
    </TimeCardTimeInterval>
    <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
      <TimeRange startDate="2003-09-23T08:00:00-08:00"/>
    </TimeCardTimeInterval>
    <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
      <TimeRange startDate="2003-09-24T08:00:00-08:00"/>
    </TimeCardTimeInterval>
    <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
      <TimeRange startDate="2003-09-25T08:00:00-08:00"/>
    </TimeCardTimeInterval>
    <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
      <TimeRange startDate="2003-09-26T08:00:00-08:00"/>
    </TimeCardTimeInterval>
  </ReportedTime>
  <SubmitterInfo submittedDate="2003-10-01T08:00:00-08:00">
    <Contact>
      <Name xml:lang="en">John Doe</Name>
    </Contact>
  </SubmitterInfo>
</TimeCard>
</TimeCardInfoRequest>
</Request>
</cXML>

```

This example show an update sent upon approval to the supplier.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxm.org/schemas/cXML/1.2.011/Fulfill.dtd">
<cXML xml:lang="en-US"
  payloadID=" tc1-update@buyer.com"
  timestamp="2003-10-01T23:00:06-08:00">
  <Header>
    <From>
      <Credential domain="NetworkId">
        <Identity>AN0100023456</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkId">
        <Identity> AN0100023457</Identity>
      </Credential>
    </To>
  </Header>
  <Body>
    <TimeCardInfoRequest>
      <Request>
        <TimeCard>
          <ReportedTime>
            <TimeCardTimeInterval duration="PT2H"
            payCode="Mealbreak" isNonBillable="yes">
              <TimeRange startDate="2003-09-22T012:00:00-08:00"
              endDate="2003-09-22T14:00:00-08:00"/>
            </TimeCardTimeInterval>
            <TimeCardTimeInterval duration="PT2H" payCode="Overtime" >
              <TimeRange startDate="2003-09-22T18:00:00-08:00"
              endDate="2003-09-22T20:00:00-08:00"/>
            </TimeCardTimeInterval>
            <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
              <TimeRange startDate="2003-09-23T08:00:00-08:00"/>
            </TimeCardTimeInterval>
            <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
              <TimeRange startDate="2003-09-24T08:00:00-08:00"/>
            </TimeCardTimeInterval>
            <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
              <TimeRange startDate="2003-09-25T08:00:00-08:00"/>
            </TimeCardTimeInterval>
            <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
              <TimeRange startDate="2003-09-26T08:00:00-08:00"/>
            </TimeCardTimeInterval>
          </ReportedTime>
          <SubmitterInfo submittedDate="2003-10-01T08:00:00-08:00">
            <Contact>
              <Name xml:lang="en">John Doe</Name>
            </Contact>
          </SubmitterInfo>
        </TimeCard>
      </Request>
    </TimeCardInfoRequest>
  </Body>
</cXML>

```

```

    <Sender>
      <Credential domain="NetworkId">
        <Identity> AN0100023456</Identity>
        <SharedSecret>coyote</SharedSecret>
      </Credential>
      <UserAgent>Suppliers TimeCard Application</UserAgent>
    </Sender>
  </Header>
  <Request>
    <TimeCardInfoRequest>
      <TimeCard type="update" status="approved" timeCardID="TC101">
        <OrderInfo>
          <OrderIDInfo orderID="PO123"
            orderDate="2003-07-22T08:00:00-08:00"/>
        </OrderInfo>
        <Contractor>
          <ContractorIdentifier domain="supplierReferenceID">Doe8610
          </ContractorIdentifier>
          <Contact>
            <Name xml:lang="en">John Doe</Name>
          </Contact>
        </Contractor>
        <ReportedTime>
          <Period startDate="2003-09-22T08:00:00-08:00"
            endDate="2003-09-26T18:00:00-08:00"/>
          <TimeCardTimeInterval duration="PT8H" payCode="Regular">
            <TimeRange startDate="2003-09-22T08:00:00-08:00"
              endDate="2003-09-22T18:00:00-08:00"/>
          </TimeCardTimeInterval>
          <TimeCardTimeInterval duration="PT2H"
            payCode="Mealbreak" isNonBillable="yes">
            <TimeRange startDate="2003-09-22T012:00:00-08:00"
              endDate="2003-09-22T14:00:00-08:00"/>
          </TimeCardTimeInterval>
          <TimeCardTimeInterval duration="PT2H" payCode="Overtime" >
            <TimeRange startDate="2003-09-22T18:00:00-08:00"
              endDate="2003-09-22T20:00:00-08:00"/>
          </TimeCardTimeInterval>
          <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
            <TimeRange startDate="2003-09-23T08:00:00-08:00"/>
          </TimeCardTimeInterval>
          <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
            <TimeRange startDate="2003-09-24T08:00:00-08:00"/>
          </TimeCardTimeInterval>
          <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
            <TimeRange startDate="2003-09-25T08:00:00-08:00"/>
          </TimeCardTimeInterval>
          <TimeCardTimeInterval duration="PT8H" payCode="Regular" >
            <TimeRange startDate="2003-09-26T08:00:00-08:00"/>
          </TimeCardTimeInterval>
        </ReportedTime>
      </TimeCard>
    </TimeCardInfoRequest>
  </Request>
</TimeCardTransaction>

```

```
</ReportedTime>
<SubmitterInfo submittedDate="2003-10-01T08:00:00-08:00">
  <Contact>
    <Name xml:lang="en">John Doe</Name>
  </Contact>
</SubmitterInfo>
<ApprovalInfo approvedDate="2003-10-02T08:00:00-08:00">
  <Contact>
    <Name xml:lang="en">John Doe</Name>
  </Contact>
</ApprovalInfo>
<DocumentReference payloadID="tc1@buyer.com"/>
</TimeCard>
</TimeCardInfoRequest>
</Request>
</cXML>
```

Chapter 9

Master Agreements

cXML supports the transmission of Master Agreement documents, which are contracts between trading partners.

This chapter describes:

- [Overview of Master Agreements](#)
- [MasterAgreementRequest](#)

Overview of Master Agreements

Master Agreements enable buyers to establish a commitment for goods and services with suppliers. They represent a common mechanism for managing supplier and budget commitments, and they enable buyers to negotiate better discounts by basing the discounts on future purchases, while enabling suppliers to more accurately forecast demand.

The Master Agreement transaction enables procurement application to facilitate the negotiation and creation of Master Agreements with suppliers and creation of Release Orders from those Master Agreements. These Agreement documents can be routed from the procurement application to the supplier by a network hub. The execution of an order against a contract is called a release.

MasterAgreementRequest

The MasterAgreementRequest document defines the Master Agreement created by the buying organization. It specifies beginning and end dates, and the committed maximum and minimum values of the agreement. It also lists maximum and minimum values and quantities for individual items.

The following example shows a MasterAgreementRequest document:

```

<MasterAgreementRequest>
  <MasterAgreementRequestHeader
    agreementID="MA123"
    agreementDate="2001-12-01"
    type="value"
    effectiveDate="2002-01-01"
    expirationDate="2002-12-31"
    operation="new">
    <MaxAmount>
      <Money currency="USD">10000</Money>
    </MaxAmount>
    <MaxReleaseAmount>
      <Money currency="USD">10000</Money>
    </MaxReleaseAmount>
    <Contact role="BuyerLocation">
      <Name xml:lang="en">Buyer Company</Name>
      <PostalAddress name="default">
        <DeliverTo>Joe Smith</DeliverTo>
        <DeliverTo>Mailstop M-543</DeliverTo>
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>90489</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Contact>
    <Comments xml:lang="en-US">well formed XML can go here.</Comments>
  </MasterAgreementRequestHeader>
  <AgreementItemOut maxQuantity="100">
    <MaxAmount>
      <Money currency="USD">1000</Money>
    </MaxAmount>
    <MaxReleaseAmount>
      <Money currency="USD">100</Money>
    </MaxReleaseAmount>
    <ItemOut quantity="1">
      <ItemID>
        <SupplierPartID>1233244</SupplierPartID>
      </ItemID>
      <ItemDetail>
        <UnitPrice>
          <Money currency="USD">1.34</Money>
        </UnitPrice>
        <Description xml:lang="en">Blue Ballpoint Pen</Description>
        <UnitOfMeasure>EA</UnitOfMeasure>
        <Classification domain="UNSPSC">12345</Classification>
        <ManufacturerPartID>234</ManufacturerPartID>
        <ManufacturerName>foobar</ManufacturerName>
      </ItemDetail>
    </ItemOut>
  </AgreementItemOut>
</MasterAgreementRequest>

```



```

        <URL>www.foo.com</URL>
    </ItemDetail>
    <Shipping trackingDomain="FedEx" trackingId="1234567890">
        <Money currency="USD">2.5</Money>
        <Description xml:lang="en-us">FedEx 2-day</Description>
    </Shipping>
    <Comments xml:lang="en-US">Any well formed XML</Comments>
</ItemOut>
</AgreementItemOut>
</MasterAgreementRequest>

```

MasterAgreementRequestHeader Element

The MasterAgreementRequestHeader contains information about the Master Agreement common to all contained items.

MasterAgreementHeader has the following attributes:

agreementDate	The date and time the agreement request was created. This is different from the effective and expiration date of the agreement.
type	Specifies whether the agreement refers to a value or quantity.
effectiveDate	Specifies the date the agreement is available for ordering or releases.
expirationDate	Specifies the date the agreement is no longer available
operation	Specifies the type of the agreement request. Can be "new", "update" or "delete". Defaults to "new". The "delete" operation is used to cancel an existing agreement. The delete request should be an exact replica of the original request.
parentAgreementPayloadID	PayloadID for the corresponding parent document from which this agreement is derived. Optional.
agreementID	The procurement system agreementID for this request.

MasterAgreementHeader can contain the following optional child elements:

MaxAmount	(Optional) Contains the maximum amount for all line items in the Master Agreement.
MinAmount	(Optional) Contains the committed amount for all line items on the Master Agreement.
MaxReleaseAmount	(Optional) The contractual maximum amount per Release of this Master Agreement.

MinReleaseAmount	(Optional) The contractual minimum amount per Release of this Master Agreement.
Contact	(Optional) Use "Contact" element to supply any additional Address or Location information.
Comments	(Optional) Can contain additional information about the status of the overall Master Agreement.
Extrinsic	(Optional) Can be used to insert additional data about the MasterAgreement for application consumption.

AgreementItemOut Element

The AgreementItemOut element specifies the requirements of a particular line item that is part of the Master Agreement contract.

AgreementItemOut has the following attributes:

maxQuantity	(Optional) Specifies the maximum quantity for this particular line Item.
minQuantity	(Optional) Specifies the minimum quantity for this particular line Item.
maxReleaseQuantity	(Optional) Specifies the maximum quantity per release for this particular line Item.
minReleaseQuantity	(Optional) Specifies the minimum quantity per release for this particular line Item.

AgreementItemOut can contain the following child elements:

MaxAmount	(Optional) Contains the maximum amount for this particular line Item.
MinAmount	(Optional) Contains the minimum amount for this particular line Item.
MaxReleaseAmount	(Optional) Indicates the item level maximum amount per release.
MinReleaseAmount	(Optional) Indicates the item level minimum amount per release.
ItemOut	A line item that is part of the master agreement. Required. The lineNumber attribute in the ItemOut specifies the corresponding lineNumber on the Master Agreement in the Procurement Application. The quantity attribute in the ItemOut should be set to "one" and ignored at the Master Agreement implementation processing stage.

Chapter 10

Later Status Changes

cXML allows entities to set the status of purchase orders and line items within them.

This chapter describes:

- [Overview of Status](#)
- [StatusUpdateRequest](#)
- [ConfirmationRequest](#)
- [ShipNoticeRequest](#)

Overview of Status

After the `OrderRequest` transaction has completed, suppliers and intermediate servers might need to communicate additional information back to the buying organization. In addition, after a buying organization receives an invoice, it might need to communicate back to the supplier about invoice status. The transactions described in this chapter are used for that purpose. These transactions share some common semantics and elements.

Like the response to an `OrderRequest` (see [“Response to an OrderRequest”](#) on page 154), none of these transactions includes a specific `Response` element. Instead, the returned document contains a nearly empty `Response` (only a `Status`). Each returned document has the form:

```
<cXML payloadID="9949494@supplier.com"
    timestamp="2000-01-12T18:39:09-08:00" xml:lang="en-US">
  <Response>
    <Status code="200" text="OK"/>
  </Response>
</cXML>
```

The returned code is “200” only if the operation completed successfully.

StatusUpdateRequest

This transaction informs an earlier node about changes in the processing status of an order or an invoice.

One change is of particular significance: when an intermediate hub successfully transmits an OrderRequest onward, it can inform the original sender or a previous hub about that success. Transitions through various queues and processing steps at a supplier or hub might also be significant to the buying organization.

Order-processing partners (such as fax or EDI service providers) send StatusUpdateRequest transaction messages to network commerce hubs to set purchase order status. It affects the order status indicator on the hub, which is visible to both buyers and suppliers. Additionally, suppliers can send this transaction to allow buying organizations to see the status of document processing within the supplier's organization.

Buying organizations use StatusUpdateRequest to update the status of invoices on network commerce hubs, which can in turn forward them to suppliers.

This request updates the processing status of a single OrderRequest document. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML xml:lang="en-US"
      payloadID="0c30050@supplierorg.com"
      timestamp="2000-01-08T23:00:06-08:00">
  <Header>
    <From>
      <Credential domain="NetworkId">
        <Identity>AN00000123</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkId">
        <Identity>AN00000456</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkId">
        <Identity>AN00000123</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Supplier's Super Order Processor</UserAgent>
    </Sender>
  </Header>
```

```
<Request>
  <StatusUpdateRequest>
    <DocumentReference
      payloadID="0c300508b7863dcc1b_14999"/>
    <Status code="200" text="OK" xml:lang="en-US">Forwarded
      to supplier</Status>
    </StatusUpdateRequest>
  </Request>
</cXML>
```

This request contains only an `DocumentReference` and a `Status` element. Both are required. The `Status` can communicate a later transport error encountered by an intermediate hub. The semantics of this element are identical to a `Status` that might have been returned in the initial HTTP response to an `OrderRequest` document.

The 200/OK code is especially important when documents are stored and forwarded. This code indicates that a supplier has begun processing the `OrderRequest` or a hub has forwarded the document. The recipient should expect no further `StatusUpdateRequest` documents after 200/OK arrives.

Suppliers and hubs utilizing the `StatusUpdate` transaction must return code 201/Accepted when an `OrderRequest` is queued for later processing. After it sends 200/OK (in the immediate Response to an `OrderRequest` or a later `StatusUpdateRequest`), the server should send no further `StatusUpdate` transactions for that order. Errors later in processing might lead to exceptions to this rule.

DocumentReference Element

The `DocumentReference` element associates a status update with a particular `OrderRequest` document and is used by the `StatusUpdateRequest` transaction. It contains enough information to associate the update request with a particular document. It repeats a required attribute of the earlier document and adds one optional identifier generated by the supplier. For example:

```
<DocumentReference
  payloadID="0c300508b7863dcclb_14999"/>
```

DocumentReference contains no elements, but has the following attribute:

payloadID	A unique number with respect to space and time that is used for logging purposes to identify documents. This value should not change in the case of retry attempts. The recommended implementation is: datetime.process id.random number@hostname Taken directly from the cXML element of the OrderRequest document.
------------------	---

PaymentStatus Element

The PaymentStatus element contains the status of a PCard transaction. The status update includes information such as the success of the transaction, transaction ID, authorization ID, order ID, total, tax, shipping information, and the time stamp of the original submission.

A StatusUpdateRequest document is sent to a supplier in response to a ConfirmationRequest with type="RequestToPay" to a network hub. This ConfirmationRequest invokes a payment service where the network hub requests a payment service provider, such as VeriSign, to perform a point of sale transaction against the PCard listed in the purchase order and return the status of the transaction. The network hub then sends the transaction status back to the supplier in a StatusUpdateRequest document. For example:

```
<StatusUpdateRequest>
  <DocumentReference payloadID="0c300508b7863dcclb_14999"/>
  <Status code="0" text="Approved">Approved</Status>
  <PaymentStatus orderID="PC100" transactionTimestamp="2000-01-08T10:00:06-08:00" type="Sale" transactionID="V20000212000" authorizationID="PN123">
    <PCard number="1234567890123456" expiration="2003-03-31"/>
    <Total>
      <Money currency="USD">500.00</Money>
    </Total>
    <Shipping>
      <Money currency="USD">20.00</Money>
      <Description xml:lang="en">shipping charge</Description>
    </Shipping>
    <Tax>
      <Money currency="USD">40.00</Money>
      <Description xml:lang="en">CA Sales Tax</Description>
    </Tax>
  </PaymentStatus>
</StatusUpdateRequest>
```

The PaymentStatus element contains the required PCard and Total element, and optionally Shipping, Tax, and Extrinsic elements.

The PCard element contains two attributes that specify the number of the PCard and its expiration date.

PaymentStatus has the following attributes

orderID	Identifies the referenced order. It is copied from the ConfirmationRequest or the OrderRequest.
transactionTimeStamp	Specifies the time when the payment transaction was submitted.
type required	Specifies the type of PCard transaction. The value must be one of the following: "Authorization" Authorizes the PCard. No charge is made. There is one authorization per order. "Settlement" Transfers the funds secured by a previous authorization transaction. "Sale" Initiates a charge to the PCard. "Credit" Initiates a credit against the original charge. Compensates for an order that did not meet buyer expectations, to make adjustments to an account that was overcharged, or to credit an account for items returned by a buyer.
transactionID	Assigned to the transaction by the payment processing gateway.
authorizationID	The authorization code for the transaction provided by the bank.

SourcingStatus Element

The SourcingStatus element provides update information for a RFQ sourcing transaction, PunchOutSetupRequest with operation="source".

```
<StatusUpdateRequest>
  <DocumentReference payloadID="123345678.RFQID:1234456787" />
  <Status code="200" text="OK">Approve Request</Status>
  <SourcingStatus action="approve" xml:lang="en"/>
</StatusUpdateRequest>
```

The action attribute identifies the update type for the transaction. Can be “approve”, “cancel”, or “deny”. The body of the SourcingStatus element can contain human-readable information about the new state of the RFQ.

InvoiceStatus Element

When using StatusUpdateRequest for invoices, include the InvoiceStatus element.

The InvoiceStatus type attribute refers to the action taken by the buying organization on the invoice. It can have the following values:

processing	The invoice was received by the buying organization and is being processed.
reconciled	The invoice sucessfully reconciled. The amounts in the invoice have not yet been paid.
rejected	The invoice failed reconciliation. The buying organization is rejecting the invoice. The Comments element should contain free text explaing why the invoice was rejected, and the actions the supplier should take. The supplier can then resubmit a corrected invoice (a new invoice document with a new invoice number).
paying	The invoice has been approved for payment and is in the payment process.
paid	The invoice amounts have been paid by the buying organization.

The PartialAmount element allows buying organizations to specify different amounts paid than the amounts specified in invoices. If invoices are paid in full, do not include PartialAmount. The existence of PartialAmount alerts the supplier to read the Comments elements which should contain more explanations on the differences.

ConfirmationRequest

This transaction provides detailed status updates on a specific Order Request. It extends the simple acknowledgment of an order, provided by StatusUpdateRequest, to a more detailed item level confirmation and ship notification.

Note: The DTD for this transaction is contained in Fulfill.dtd rather than cXML.dtd.

No specific Response document is required for this transaction. Servers must respond to a ConfirmationRequest with a generic Response document.

A document is one of six types, specified by the type attribute of the ConfirmationHeader element: “accept”, “allDetail”, “reject”, “except”, “detail”, and “requestToPay”. With a type of “detail”, you can update portions of an Order Request, such as prices, quantities, and delivery dates, reject portions, and add tax and shipping information. Only the line items mentioned are changed. With a type of “allDetail”, you can update all information of specified line items without rejecting or accepting the order. You can apply the confirmation to the entire order request using the types “accept”, “reject”, and “except”. “allDetail” and “detail” update individual lines, they do not accept or reject the entire order in one stroke.

A ConfirmationRequest with type=“requestToPay” invokes a payment service where the network hub requests a payment service provider, such as VeriSign, to perform a point of sale transaction against the PCard listed in the purchase order and return the status of the transaction. The network hub then sends the transaction status back to the supplier in a StatusUpdateRequest document.

The following example shows a ConfirmationRequest element that is of type “accept”.

```
<ConfirmationRequest>
  <!-- Without the confirmID, it remains possible to update this
  confirmation. An update would refer (in the OrderReference element) to the same
  OrderRequest document, would describe the status of the same items, and would
  point to this document through its DocumentReference element. However, the
  confirmID makes the update much more explicit.-->
  <ConfirmationHeader type="accept" noticeDate="2000-10-12T18:39:09-08:00"
    confirmID="C999-234" invoiceID="I1010-10-12">
    <Shipping>
      <Money currency="USD">2.5</Money>
      <Description xml:lang="en-CA">FedEx 2-day</Description>
    </Shipping>
    <Tax>
      <Money currency="USD">0.19</Money>
      <Description xml:lang="en-CA">CA Sales Tax</Description>
    </Tax>
    <Contact role="shipFrom">
      <Name xml:lang="en-CA">Workchairs, Vancouver</Name>
      <PostalAddress>
        <Street>432 Lake Drive</Street>
        <City>Vancouver</City>
        <State>BC</State>
        <PostalCode>B3C 2G4</PostalCode>
        <Country isoCountryCode="CA">Canada</Country>
      </PostalAddress>
      <Phone>
        <TelephoneNumber>
          <CountryCode isoCountryCode="CA">1</CountryCode>
          <AreaOrCityCode>201</AreaOrCityCode>
          <Number>921-1132</Number>
        </TelephoneNumber>
      </Phone>
    </Contact>
  </ConfirmationHeader>
</ConfirmationRequest>
```

```
        </TelephoneNumber>
      </Phone>
    </Contact>
    <Comments xml:lang="en-CA">Look's great</Comments>
  </ConfirmationHeader>
  <!-- The orderID and orderDate attributes are not required in the
       OrderReference element. -->
  <OrderReference orderID="DO1234">
    <DocumentReference payloadID="32232995@ariba.acme.com" />
  </OrderReference>
</ConfirmationRequest>
```

Multiple "detail" type Confirmation Requests can refer to a single Order Request, but they must not refer to common line items.

To perform a substitution, include a ConfirmationItem element to specify the item to be replaced, then provide an ItemIn element for the replacement. Only use ItemIn elements for substitutions. You should then wait for a corresponding change order from the buyer before shipping.

The ConfirmationRequest element is a request to add confirmation information to the data known about an order at the receiving server. It can contain three elements: ConfirmationHeader, OrderReference, and an optional ConfirmationItem. If the ConfirmationRequest type specified in the ConfirmationHeader is either "detail" or "except", you can include ConfirmationItem elements to update specific line items from an Order Request.

While multiple confirmations can be sent for one order, each confirmation must mention a line item only once. In addition, a line item must not be mentioned in more than one confirmation request. Multiple confirmations are allowed, and sensible, only when the Request type specified is "allDetail" or "detail". Only one confirmation per order is allowed when the Request type is "accept", "except", or "reject". When a confirmation with one of these types arrives, the receiving system must discard any and all previous confirmations for the order.

ConfirmationItem elements can appear in any order within the ConfirmationRequest. However, listing the lineNumber elements in ascending order is easily readable and preferred. Again, no line item can appear more than once within a ConfirmationRequest element.

OrderReference Element

The OrderReference element provides a clear reference to a prior OrderRequest document. While the contained DocumentReference provides an unambiguous reference, the additional attributes of the OrderReference allow the ConfirmationRequest and ShipNoticeRequest to be viewed independently. Contains a DocumentReference element (see page 222) and two attributes: orderID and orderDate.

orderID Attribute

Specifies the buyer system orderID for the confirmation, that is, the PO number. When used, it must be copied directly from the referenced OrderRequest document's OrderRequestHeader element.

orderDate Attribute

Specifies the date and time the OrderRequest was created. If present, it must be copied directly from the referenced OrderRequest document's OrderRequestHeader.

ConfirmationHeader Element

The ConfirmationHeader element contains information that is common to all items contained in the Confirmation Request. It has the following attributes:

- type
- noticeDate
- invoiceID
- operation
- ConfirmID
- incoTerms

The ConfirmationHeader element can contain the following elements:

- DocumentReference
- Tax
- Shipping
- Total
- Contact
- Hazard
- Comments
- Extrinsic

If the ConfirmationRequest type specified in the ConfirmationHeader (see page 211) is either, "allDetail", "detail" or "except", you can include ConfirmationItem elements to update specific line items from an Order Request.

The following example shows a Confirmation Request of type "except":

```
<ConfirmationRequest>
  <!-- Without the confirmID, it remains possible to update the original confirmation.
  This update refers (in the OrderReference element) to the same OrderRequest
  document, describes the status of the same items and refers to the original
  confirmation document in the DocumentReference element. However, the confirmID
  makes the update much more explicit.
  Note: The noticeDate changes to match the time of the update and not the original
  confirmation time.-->

  <ConfirmationHeader type="except" noticeDate="2000-10-13T18:39:09-08:00"
    confirmID="C999-234" operation="update"
    invoiceID="I1102-10-13">
    <DocumentReference payloadID="1233444-2001 @premier.workchairs.com" />
    <Total>
      <Money currency="USD">190.60</Money>
    </Total>
    <Shipping>
      <Money currency="USD">2.5</Money>
      <Description xml:lang="en-CA">FedEx 2-day</Description>
    </Shipping>
    <Tax>
      <Money currency="USD">0.19</Money>
      <Description xml:lang="en-CA">CA Sales Tax</Description>
    </Tax>
    <Contact role="shipFrom">
      <Name xml:lang="en-CA">Workchairs, Vancouver</Name>
      <PostalAddress>
        <Street>432 Lake Drive</Street>
        <City>Vancouver</City>
        <State>BC</State>
        <PostalCode>B3C 2G4</PostalCode>
        <Country isoCountryCode="CA">Canada</Country>
      </PostalAddress>
      <Phone>
        <TelephoneNumber>
          <CountryCode isoCountryCode="CA">1</CountryCode>
          <AreaOrCityCode>201</AreaOrCityCode>
          <Number>921-1132</Number>
        </TelephoneNumber>
      </Phone>
    </Contact>
    <Comments xml:lang="en-CA">Look's great, but for the price.</Comments>
  </ConfirmationHeader>
  <!-- The orderID and orderDate attributes are not required in the OrderReference
```

```

element. -->
<OrderReference orderID="DO1234">
  <DocumentReference payloadID="32232995@ariba.acme.com" />
</OrderReference>
<ConfirmationItem lineNumber="1" quantity="10">
  <UnitOfMeasure>EA</UnitOfMeasure>
  <ConfirmationStatus quantity="10" type="detail" shipmentDate="2000-10-14"
    deliveryDate="2000-10-19">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice>
      <Money currency="USD">1.64</Money>
    </UnitPrice>
    <Comments xml:lang="en-CA">Very sorry. There's been a slight
      (30 cents) price increase for that colour and it will be one day late.
    </Comments>
  </ConfirmationStatus>
</ConfirmationItem>
</ConfirmationRequest>

```

type Attribute

This required attribute specifies the type of confirmation.

"accept"	Accept the entire order as described in the referenced OrderRequest document A document of this type can contain ConfirmationItem elements. If ConfirmationItem elements are present in the ConfirmationRequest document, they must contain only ConfirmationStatus elements of type="accept".
"allDetail"	Update only those line items described in the included ConfirmationItem elements. Line items not mentioned specifically in this document retain their current status. Unlike the "detail" type, this type of confirmation includes all information known by the supplier, whether or not it differs from the data provided in the original OrderRequest document. An "allDetail" confirmation is compatible with current EDI and order entry tools, which commonly send buyers a snapshot of an order in supplier's systems. Due to the reconciliation issues caused by confirmations of this type, it is recommended that this type be considered as a "bridge" strategy for the short term. An "allDetail" confirmation must contain ConfirmationItem elements. The contained ConfirmationStatus elements must have types "allDetail", "reject", or "unknown". "accept" or "detail" ConfirmationStatus types are not allowed because they conflict with the requirements of this document type.
"reject"	Reject the entire order. Specify a reason for the rejection in the Comments element. A document of this type must not contain ConfirmationItem elements.

"detail"	<p>Update individual line items as specified within the contained ConfirmationItem elements. Line items not mentioned specifically retain their current states. This document type should include only information that differs from the information in the original OrderRequest document.</p> <p>Do not include the variations described in an earlier ConfirmationRequest in later ConfirmationRequest documents that restore information provided in the OrderRequest. For example, the Tax element might appear in the ConfirmationStatus of one ConfirmationRequest but not in an update to that confirmation. This signifies that the original OrderRequest actually contained the correct charge.</p> <p>This document type must contain ConfirmationItem elements. The contained ConfirmationStatus elements can have any type except "allDetail".</p>
"except"	<p>Accept the entire order as described in the referenced OrderRequest document with exceptions listed within the ConfirmationItem elements. Accept line items not mentioned without change.</p> <p>This document type must contain ConfirmationItem elements. The contained ConfirmationStatus elements can have any type except "allDetail".</p>
"requestToPay"	<p>This type of confirmation is for the supplier to request the initiation of payment transactions for either the whole order or some line items.</p> <p>If the request contains no ConfirmationItem elements, the intent will be to initiate payment transaction on the total amount of all line items in the order, except those being rejected.</p> <p>If the request contains ConfirmationItem elements, payment transaction will be initiated against the specified items and quantities.</p> <p>A ConfirmationItem in this type of request does not have to describe the complete line item. It should only contain "requestToPay" ConfirmationStatus elements for new payment transactions.</p>
"backordred"	<p>Sets the entire order to backordered status. The supplier does not have the items in stock, but will ship them when they are available.</p>

noticeDate Attribute

Specifies the date and time the confirmation document was created.

invoiceID Attribute

The invoiceID attribute is an optional supplier-generated identifier for an invoice associated with the items described in this confirmation. It is identical to the Invoice Number that appears at the top of a physical invoice.

confirmID Attribute

A supplier-specified optional identifier for the document assigned by the supplier. The attribute is user-visible and secondary to the document’s PayloadID.

This value does not vary as a particular confirmation is updated. That is, documents with operation="update" describing the status of the same items in the same order share a confirmID with the original ConfirmationRequest with operation="new".

When the confirmID does not appear in an operation="new" ConfirmationRequest, it must not appear in a corresponding operation="update" document. The DocumentReference element contained in the update’s ConfirmationHeader and the payloadID attribute of the original or previous update link the two documents.

operation Attribute

This optional attribute specifies whether the confirmation is new, or an update to a previous confirmation.

"new"	Default value. No previous confirmation request has been sent.
"update"	Updates a previous confirmation request. The confirmID must match a previous request’s confirmID.

An "update" confirmation allows a supplier to correct an error in a confirmation or to add additional information learned later. In either case, an "update" document must be complete: all data from the original confirmation or a previous update should be discarded by the recipient.

The requirement that the confirmation be complete enables ConfirmationRequest information to expand. There are no restrictions on new items not yet referenced in an "update" ConfirmationRequest. New items must not have already been mentioned in another ConfirmationRequest unless all of the items from the other confirmation are now described in the consolidated document. This protocol does not support splitting confirmations (sending an "update" ConfirmationRequest document describing a subset of items in an earlier version), or partial consolidations of confirmations (sending an "update" ConfirmationRequest document that contains a subset of information from another confirmation).

An "update" ConfirmationRequest must contain the same confirmID, if any, as the previous version of the confirmation. This attribute is an unambiguous and a direct connection between all versions of the confirmation.

An "update" ConfirmationRequest must also include a DocumentReference element in the ConfirmationHeader. See [“DocumentReference Element”](#) on page 214 for more information on this element. This element sequences multiple versions of a confirmation and is the only link between those versions. See [“confirmID Attribute”](#) on page 213 for more of the implications of leaving out the attribute. Other confirmations discarded through consolidations as previously described are not explicitly referenced by the new, larger ConfirmationRequest document.

A confirmation can not be deleted; the protocol does not include a delete option for this request. Suppliers must replace incorrect or invalid confirmations with correct information. A type="unknown" ConfirmationStatus will reset such information to its original state. This covers the case of an error in accepting or rejecting an item that has not been researched.

incoTerms Attribute

The incoTerms attribute specifies optional shipping terms defined by the *International Chamber of Commerce*. These terms inform the buyer which portion of the shipping charges are their responsibility. Allowed values include:

"cfr"	Cost and freight
"cif"	Cost, insurance, and freight
"cip"	Carriage and insurance paid to
"cpt"	Carriage paid to
"daf"	Delivered at frontier
"ddp"	Delivered duty paid
"ddu"	Delivered duty unpaid
"deq"	Delivered ex quay (duty paid)
"des"	Delivered ex ship
"exw"	Ex works
"fas"	Free alongside ship
"fca"	Free carrier
"fob"	Free on board vessel

DocumentReference Element

The DocumentReference element should appear only when operation is "update" (see page 213). It should reference the most recent ConfirmationRequest document for this particular confirmation, usually indicated by a common confirmID. For example, when a confirmation is created, updated, and then updated again, the final document

should contain a DocumentReference referring to the previous ConfirmationRequest with operation="update". That document, in turn, refers to the original operation="new" ConfirmationRequest document (see page 213).

Tax and Shipping Elements

Tax and Shipping amounts can be updated and included in the confirmation with new values without any corresponding line item information.

Total Element

The Total value should match the OrderRequest document value unless a ConfirmationItem describes a new UnitPrice or quantity. It is not necessary to copy this information from the OrderRequest document: although permissible, Total, Tax, and Shipping information should not be included if they match those amounts in the original order.

Contact Element

The Contact element should be used primarily to add new information about an order. It is not necessary to copy this information from the OrderRequest document.

Contact role values include:

"technicalSupport"	Technical support
"customerService"	Customer service
"sales"	Sales
"shipFrom"	Starting point for shipments related to this order
"shipTo"	Copies the ShipTo element from the OrderRequest document
"payTo"	Where payment for this order should be sent
"billTo"	Copies the BillTo element from the OrderRequest document
"supplierCorporate"	Supplier at corporate

Elements in the Contact list can appear in any order. A contact role must not appear more than once within a ConfirmationHeader element.

Hazard Element

Elements in the Hazard list can appear in any order. The same hazard should not be listed more than once in a ConfirmationHeader element. Each hazard listed at this level should apply to the entire order or all items mentioned in the confirmation. A

ConfirmationRequest that updates the status of a single line item should not include Hazard elements in the ConfirmationItem element. See [“Hazard Element”](#) on page 231 for more information.

Comments Element

The Comments element can contain additional information about the status of the overall order, or the portion described in this confirmation, such as payment terms, additional details on shipping terms and clarification of the status. For status information, terms such as “backordered”, “shipped”, and “invalid” might be appropriate. All such data are intended for human use.

Extrinsic Element

The Extrinsic element list can be used to insert additional data about the order for application consumption. These elements can include pre-defined keywords and values affecting workflow in the receiving system.

Elements in the Extrinsic list can appear in any order. An extrinsic type must not appear more than once within a ConfirmationHeader element. A type must not be mentioned both in this list and in a particular ConfirmationStatus element. The ConfirmationHeader must not contain a default extrinsic value overridden at the lower level.

ConfirmationItem Element

The ConfirmationItem element completely describes the status of a specific line item. The ConfirmationItem element can contain the following elements: UnitOfMeasure, ConfirmationStatus, Contact, and Hazard. ConfirmationStatus can occur more than once, and only Contact is optional.

ConfirmationItem has the following attributes:

quantity	Specifies how many items were ordered. Expressed in units given in the UnitOfMeasure element. Matches the quantity value for the line item's ItemOut element in the corresponding OrderReference element. Required.
lineNumber	Position, counting from 1, of the item in an order. Matches the corresponding line item, ItemOut, in the document referenced by the OrderReference element. Required.

You can use more than one ConfirmationRequest document to update the status of an entire order, but only mention a particular line item in one document and in only one ConfirmationItem within that document.

Contact Element

Use Contact elements in the ConfirmationItem to describe contacts specific to the item. The elements can be in any order. If you specify a particular Contact role, specify it in the ConfirmationItem or ConfirmationHeader but not both. Do not specify the role more than once within a ConfirmationItem.

List elements in the Contact list in any order. Do not add a Contact role attribute more than once within a ConfirmationItem element.

Hazard Element

List elements in the Hazard list in any order. Do not list the same hazard more than once in a ConfirmationItem. Each hazard listed at this level, in a ConfirmationItem element, must apply to this specific line item. A ConfirmationRequest that updates the status of a single line item should not include Hazard elements in the ConfirmationItem element.

ConfirmationStatus Element

The ConfirmationStatus element provides the status of a specific line item or portion thereof. Quantities at this level must sum to the quantity in the containing ConfirmationItem. Use a consistent UnitOfMeasure in the ConfirmationItem element and its contained ConfirmationStatus element. In a substitution, you can use a different UnitOfMeasure in the ItemDetail contained within the ItemIn element.

When accepting or rejecting an item, include only a UnitOfMeasure element in the ConfirmationStatus element.

Use an ItemIn element only to recommend a substitution. With a substitution, you must match the quantity of the ItemIn element to that of the containing ConfirmationStatus, unless the UnitOfMeasure has changed. This requires an ItemDetail element within the ItemIn element.

You can update UnitPrice, Tax and Shipping amounts in the ConfirmationStatus element without a complete part substitution. It is not necessary to copy this information from the OrderRequest document. Do not include UnitPrice, Tax, and Shipping if they match those in the original ItemOut element.

When the type is "accept", "allDetail", or "detail", you can add tax or shipping amounts not mentioned in the original order. Use the "accept" type when these additions are the only changes to the order. Use the "detail" type to indicate a substitution if there is an ItemIn

element, a price change if there is a UnitPrice element, or a delayed shipment if there is a deliveryDate attribute. The "allDetail" type requires reconciliation software to determine what has changed since the original order.

Use the Comments element to add information about the status of this portion of the item. Terms such as "backordered", "shipped", and "invalid" might be sensible. All such data is intended for human use.

Alternately, use the Extrinsic element list to insert additional data about this particular item portion for application consumption. These elements can include pre-defined keywords and values affecting workflow in the receiving system.

Elements in the Extrinsic list can appear in any order. An extrinsic attribute value must not appear more than once within a ConfirmationStatus element. A type must not be mentioned both in this list and in the overall ConfirmationHeader element. The ConfirmationHeader must not contain a default extrinsic value overridden at this lower level.

quantity Attribute

Specifies how many items have this status. Expressed in the units specified in the UnitOfMeasure element.

type Attribute

Specifies the status of this portion of the order.

"accept"	Accept this portion as described in the referenced ItemOut element.
"allDetail"	<p>Accept this portion of the line item as detailed in the contents of this ConfirmationStatus element. These contents completely describe what will be shipped. Unlike the "detail" type, this confirmation type includes all information known by the supplier, whether or not it differs from the data provided in the original OrderRequest document.</p> <p>This type is provided for compatibility with current EDI and order entry tools, which commonly send the buyer a snapshot of an order in the supplier's systems. Due to the reconciliation issues caused by confirmations of this type, it is recommended that you use this type as a "bridge" strategy suitable only for the short term.</p> <p>Allowed only in documents whose ConfirmationHeader type is "allDetail".</p>
"detail"	Accept this portion with the changes detailed in the ConfirmationStatus element. At least one of the UnitPrice, Shipping, Tax, or ItemIn elements, or the deliveryDate attribute must be present. This is a substitution if there is an ItemIn element, a price change if there is a UnitPrice element, or a delayed shipment if there is a deliveryDate attribute.
"reject"	Reject this portion of the line item.
"unknown"	<p>The status of this portion of the line item is not known at the time of this confirmation. This line item status provides a placeholder while the supplier does further research. Update confirmations can also reset the status of a line item portion to "unknown" when an earlier confirmation incorrectly accepted or rejected that portion.</p> <p>Allowed only in documents whose ConfirmationHeader type is "allDetail", "detail", or "except".</p>
"backordered"	Sets this portion of the line item to backordered status. The supplier does not have the items in stock, but will ship them when they are available.

shipmentDate Attribute

Specifies the date and time this shipment is expected to leave the supplier. Use the ConfirmationStatus element to include this information if the type is "accept", "allDetail", or "detail".

deliveryDate Attribute

Specifies the new date and time this shipment is expected to arrive. Do not include if the value matches the requestedDeliveryDate attribute, if any, in the corresponding OrderRequest document. Otherwise, use the ConfirmationStatus element to include this information if its type is "accept", "allDetail", or "detail".

ShipNoticeRequest

Suppliers use the ShipNoticeRequest document to send shipment information about orders. This transaction describes a single shipment and can contain portions of multiple orders as well as hazard information for the entire shipment or individual line items.

Note: The DTD for this transaction is contained in Fulfill.dtd rather than cXML.dtd.

ShipNoticeRequest can contain the following elements:

- ShipNoticeHeader
- ShipControl
- ShipNoticePortion

ShipNoticeRequest documents do not provide updates to tax and shipping amounts. This information should be transmitted with ConfirmationRequest documents. If necessary, you can send a ConfirmationRequest with operation="update" with this information after the shipment has been delivered.

ConfirmationRequest and ShipNoticeRequest documents with operation="update" must include all relevant information from the original OrderRequest document.

The following example shows a ShipNoticeRequest element:

```
<ShipNoticeRequest>
  <ShipNoticeHeader shipmentID="S2-123" noticeDate="2000-10-14T18:39:09-08:00"
    shipmentDate="2000-10-14T08:30:19-08:00"
    deliveryDate="2000-10-18T09:00:00-08:00">
    <Contact role="shipFrom">
      <Name xml:lang="en-CA">Workchairs, Vancouver</Name>
      <PostalAddress>
        <Street>432 Lake Drive</Street>
        <City>Vancouver</City>
        <State>BC</State>
        <PostalCode>B3C 2G4</PostalCode>
        <Country isoCountryCode="CA">Canada</Country>
      </PostalAddress>
      <Phone>
        <TelephoneNumber>
          <CountryCode isoCountryCode="CA">1</CountryCode>
          <AreaOrCityCode>201</AreaOrCityCode>
          <Number>921-1132</Number>
        </TelephoneNumber>
      </Phone>
    </Contact>
  </ShipNoticeHeader>
</ShipNoticeRequest>
```

```

        </Phone>
      </Contact>
      <Comments xml:lang="en-CA">Got it all into one shipment.</Comments>
    </ShipNoticeHeader>
    <ShipControl>
      <CarrierIdentifier domain="SCAC">FDE</CarrierIdentifier>
      <CarrierIdentifier domain="companyName">Federal Express</CarrierIdentifier>
      <ShipmentIdentifier>8202 8261 1194</ShipmentIdentifier>
    </ShipControl>
    <ShipNoticePortion>
      <!-- The orderID and orderDate attributes are not required in the OrderReference
           element. -->
      <OrderReference orderID="DO1234">
        <DocumentReference payloadID="32232995@ariba.acme.com" />
      </OrderReference>
    </ShipNoticePortion>
  </ShipNoticeRequest>

```

The ShipNoticeRequest element contains information about a ship notice common to all contained items. It is not necessary to copy this information from the OrderRequest document. The Contact element should be used primarily to add new information about an order.

The ShipNoticeRequest element contains three elements: ShipNoticeHeader, ShipControl, and ShipNoticePortion. All are required, and both ShipNoticePortion and ShipControl can occur more than once.

Shipments with multiple responsible carriers are described in one of two ways:

1. A single carrier or third-party logistics provider creates a tracking identifier that can be used to retrieve information about the entire trip. Suppliers send such information in a single ShipControl element.
2. Each segment requires a separate tracking number. Suppliers send such information with one ShipControl element per segment.

ShipControl elements must appear in the order the shipment will travel. The first such element must not have an explicit starting date, the ShipControl startDate attribute must not be present, and that carrier's control must begin at the shipment's origination time specified by the ShipNoticeHeader shipmentDate attribute value. All later ShipControl elements must have increasing, or later, starting dates specified by the ShipControl startDate attribute value.

ShipNoticePortion elements can appear in any order. A particular order, with ShipNoticePortion, OrderReference, or DocumentReference payloadID attribute value, must not appear more than once in a ShipNoticeRequest element.

Note: Many elements and attributes in the ShipNoticeRequest and ShipNoticeHeader elements are optional only for the operation="delete" case. For other operations, one or more ShipControl and ShipNoticePortion elements must appear in a ShipNoticeHeader element.

ShipNoticeHeader Element

The ShipNoticeHeader element contains information about a ship notice common to all contained items. The ShipNoticeHeader element can contain the following elements: ServiceLevel, DocumentReference, Contact, Hazard, Comments, and Extrinsic, all of which are optional.

ServiceLevel Element

One or more ServiceLevel elements must appear in all ShipNoticeRequest documents, except when operation="delete" is specified. Each ServiceLevel must contain a single string corresponding to the level of service, such as "overnight", provided by the carrier for this shipment. When multiple ServiceLevel elements appear, all must describe the same level of service in different languages or locales. No two ServiceLevel elements can have the same xml:lang attribute. Elements in such a list can appear in any order.

DocumentReference Element

The contained DocumentReference element appears only when the operation is "update" or "delete". In that case, the DocumentReference element references the most recent ShipNoticeRequest document for this particular ship notice, usually indicated by a common shipmentID. For example, when a ship notice is created, updated, and then updated again, the final document should contain a DocumentReference referring to the previous ShipNoticeRequest with operation="update". That document, in turn, refers to the original operation="new" ShipNoticeRequest document.

Contact Element

Contact roles can include: technicalSupport, customerService, sales, shipFrom (starting point for this shipment), shipTo (should echo the ShipTo element from the OrderRequest documents), buyerCorporate (details the supplier has about the buying organization), and supplierCorporate. Generally, it is not necessary to copy information from the various OrderRequest documents: the Contact element should be used primarily to add information to that known about an order.

Elements in the Contact list can appear in any order. A Contact role attribute value must not appear more than once within a ShipNoticeHeader element.

Hazard Element

Elements in the Hazard list can appear in any order. The same hazard should not be listed more than once in a ShipNoticeHeader. Each hazard listed at this level, in a ShipNoticeHeader element, should apply to the entire shipment, or to all items contained in this shipment. A ShipNoticeRequest for a single line item should not include Hazard elements in the ShipNoticeItem element.

Comments Element

Use the Comments element to include additional information about the shipment. In the ShipNoticeHeader element, that information must be common to all contained items and routes. All such data must be intended for human use.

Extrinsic Element

Alternately, use the Extrinsic element list to insert additional data about the shipment for application consumption. These elements can include pre-defined keywords and values affecting workflow in the receiving system.

Elements in the Extrinsic list can appear in any order. An extrinsic type, Extrinsic name attribute value, must not appear more than once within a ShipNoticeHeader element. A type must not be mentioned both in this list and in a particular ShipControl or ShipNoticePortion element. The ShipNoticeHeader must not contain a default extrinsic value overridden at either lower level.

shipmentID Attribute

A supplier-specified optional identifier for the document. The attribute is user-visible and secondary to the document's PayloadID. It is required.

This value does not vary as a particular ship notice is updated. That is, "update" or "delete" documents describing the same shipment share a shipmentID with the original "new" ShipNoticeRequest.

operation Attribute

This optional attribute specifies whether the ShipNoticeRequest document is new or an update to a previous ship notice.

"new"	Default value. No previous ship notice has been sent.
"update"	Updates a previous ship notice request. Allows a supplier to correct an error in a ship notice or to add additional information learned later. In either case, an "update" document must be complete: all data from the original should be discarded by the recipient. The shipmentID must match a previous request's shipmentID.
"delete"	Removes the changes described in the previous new or updated ShipNoticeRequest from the state of the shipment. Only use when the supplier discards a planned shipment or incorrectly sends a ShipNoticeRequest about an order that will not take place. The shipmentID must match a previous request's shipmentID.

If the operation is not "new", explicitly or by default, you must also include in the ShipNoticeRequest a DocumentReference element in the ShipNoticeHeader element. See [“DocumentReference Element”](#) on page 222 for more information on this element. This effectively sequences multiple versions of a ship notice.

noticeDate Attribute

Specifies the date and time the ShipNoticeRequest document was created. Required.

shipmentDate Attribute

The date and time the shipment left the supplier. You must specify this attribute in all ShipNoticeRequest documents except when the operation is "delete".

deliveryDate Attribute

Specifies the date and time this shipment is expected to arrive. While this value can default to the requestedDeliveryDate of a single order, that attribute is optional in an OrderRequest document, and the ShipNoticeRequest can refer to multiple OrderRequest documents. You must include this attribute in all ShipNoticeRequest documents except when the operation is "delete".

ServiceLevel Element

Specifies a language-specific string for the service level code. Each ServiceLevel must contain a string in the specified language that corresponds to the level of service, such as “overnight”, provided by the carrier for this shipment. It has the required attribute xml:lang (see page 88).

ShipControl Element

Specifies the carrier responsible for some portion of the shipment. A ShipControl element contains the CarrierIdentifier, ShipmentIdentifier, PackageIdentification, Route, Contact, Comments, and Extrinsic elements.

The shipment is tracked using the identifiers provided at this level. Those identifiers should be valid from the startDate of one ShipControl element or the shipment’s shipmentDate until the startDate of the next.

CarrierIdentifier

The CarrierIdentifier list can include multiple identifiers for the same carrier. Elements in this list can appear in any order. A particular identification domain (CarrierIdentifier@domain attribute value) must not appear more than once in a ShipControl element. The identification provided by all elements of the CarrierIdentifier list must correspond to the same company.

Route Element

If present, Route elements must be in the order the shipment will travel.

Contact Element

The most common Contact roles in this element are:

"carrierCorporate"	Details the contact information the supplier has about the carrier organization.
"shipFrom"	A Contact element with role "shipFrom" must appear in all ShipControl elements after the first. This role must not appear in the first ShipControl element because it would duplicate that role in the overall ShipNoticeHeader element.

Do not use a “shipTo” role in this element because a Contact with role "shipTo" would always duplicate information in the following ShipControl element or that role in the ShipNoticeHeader. Control passes from one carrier to another at a particular location and estimated time.

List the elements in Contact in any order. A Contact role attribute value must not appear more than once within a ShipControl element.

Comments Element

The Comments element can contain additional information about the shipment while under the control of this carrier. In the context of the ShipControl element, that information must be common to all contained routes or made clear which Route is affected. All such data must be intended for human use.

Extrinsic Element

Alternately, the Extrinsic element list can be used to insert additional data about this carrier or their period of responsibility for application consumption. These elements can include pre-defined keywords and values affecting workflow in the receiving system.

Elements in the Extrinsic list can appear in any order. An Extrinsic name attribute value must not appear more than once within a ShipControl element. The same type must not be mentioned both in this list and in the overall ShipNoticeHeader element. The ShipNoticeHeader must not contain a default extrinsic value overridden at this lower level.

startDate Attribute

Specifies the date and time this shipment started this part of the route. Required for all ShipControl elements after the first. This attribute must not appear in the first ShipControl element because it would duplicate the ShipNoticeHeader's shipmentDate attribute.

Route Element

Specifies how the shipment will travel on this segment. If two ShipmentIdentifier values are present, the second defines the end of a contiguous and inclusive range of numbers that appear on the shipment. Route can contain a Contact element.

The only Contact role should be "carrierCorporate", which details the contact information the supplier has about the carrier organization, "shipFrom", and "shipTo".

Each carrier within a segment controlled by a third-party logistics provider provides tracking information to that provider externally. The ShipNoticeRequest includes tracking information at the ShipControl level only.

A Route element can describe only a single mode of travel. If described at all, each mode of a multi-modal route must be described by a separate Route element. It is not necessary to describe every leg of the journey to the buyer's ShipTo location.

The "carrierCorporate" role is relevant at this level only when a third party is providing tracking information across multiple carriers. A Contact element with role "shipFrom" must appear in all Route elements after the first. Route elements are not required to describe the entire travel under a specific carrier's control. They can describe a discontinuous stream of events, starting and ending at different times and locations.

Elements in the Contact list can appear in any order. A Contact role attribute value must not appear more than once within a Route element.

method Attribute

Identifies the transportation type code.

"air"	Transportation by flight
"motor"	Transportation by land motor craft, common carrier
"rail"	Transportation by rail
"ship"	Transportation by boat; ocean

Because shipments can travel through multiple segments with different methods, this attribute has no default.

startDate Attribute

Specifies the date and time this shipment started this part of the trip. Required in all Route elements after the first.

endDate Attribute

Specifies the date and time this shipment ended this part of the trip. Must come after startDate. If any Route elements follow, the startDate of that element must not precede this value.

CarrierIdentifier Element

Identifies the carrier that will transport this shipment. There is one attribute, called domain.

domain Attribute

Specifies the domain in which CarrierIdentifier value has meaning. For example, "SCAC" for Standard Carrier Alpha Code, or the legal company name.

Recognized domains include the following:

<i>company name</i>	The legal name for this company. In some cases, this can also be provided in a Contact element with role "carrierCorporate". Using a Contact element should be reserved for cases in which additional detail about the carrier must be conveyed.
"SCAC"	Standard Carrier Alpha Code. www.nmfta.org
"IATA"	International Air Transport Association. www.iata.org
"AAR"	Association of American Railroads. www.aar.org
"UIC"	International Union of Railways. www.uic.asso.fr
"EAN"	European Article Numbering. www.ean-ucc.org
"DUNS"	Dun and Bradstreet's Data Universal Numbering System. www.dnb.com/dnbhome.htm

ShipmentIdentifier Element

An identifier defined by the carrier that appears on the shipment that can be used to obtain additional detail about the shipment. Has meaning in the domain described by the CarrierIdentifier values in the containing Route element.

Essentially, this is a tracking number. Different carriers have different names for shipment identifiers. This is commonly called a way bill number, a pro number, and also a bill of lading. They all represent tracking numbers.

Packageldentification Element

Specifies the identifiers that appear on the containers, skids, boxes, or packages that constitute the shipment. The range of numbers described is inclusive at both extremes.

rangeBegin Attribute

Specifies the earliest number that appears on the separate elements in this shipment.

rangeEnd Attribute

Specifies the highest number that appears on the separate elements in this shipment. Must be greater than or equal to rangeBegin.

ShipNoticePortion Element

Contains purchase order and item information. Specifies what will be in the shipment. It contains three elements, OrderReference, ShipNoticeItem, Contact, Comments, and Extrinsic. All but OrderReference are optional. It contains two attributes: quantity and lineNumber.

OrderReference Element

A particular OrderRequest specified in the OrderReference element must be mentioned in at most one ShipNoticePortion element. While multiple shipments can be sent for one order, a ship notice must mention each order only once.

If a ShipNoticePortion element contains no ShipNoticeItem elements, the entire referenced order is included in the shipment. This simplifying option prevents inclusion of hazard and packaging information.

Contact Element

Any Contact elements provided at this level describe contacts specific to this portion of the order. The ShipNoticeHeader description mentions roles appropriate at this level as well, though shipFrom, shipTo, buyerCorporate, and supplierCorporate information should not vary at this level. A particular Contact role must not appear in both the ShipNoticePortion and ShipNoticeHeader elements. Therefore, roles such as “technicalSupport”, “customerService”, and “sales” are most appropriate within the ShipNoticePortion.

Elements in the Contact list can appear in any order. A Contact role attribute value must not appear more than once within a ShipNoticePortion element.

Comments Element

The Comments element can contain additional information about the order in this shipment. In this context (the ShipNoticePortion element), that information must be common to all contained items or make it clear which ShipNoticeItem is affected. All such data must be intended for human use.

Extrinsic Element

Alternately, the Extrinsic element list can be used to insert additional data about this order for application consumption. These elements can include pre-defined keywords and values affecting workflow in the receiving system.

Elements in the Extrinsic list can appear in any order. An Extrinsic name attribute value must not appear more than once within a ShipNoticePortion element. A type must not be mentioned both in this list and in the overall ShipNoticeHeader element. The ShipNoticeHeader must not contain a default extrinsic value overridden at this lower level.

ShipNoticeItem Element

The portion of a specific line item that is part of this shipment. Each line item from an order must be mentioned in at most one ShipNoticeItem element. ShipNoticeItem contains three elements: UnitOfMeasure (see page 55), Packaging, and Hazard.

Elements in the Hazard list can appear in any order. The same Hazard should not be listed more than once in a ShipNoticeItem. Each Hazard listed at this level (in a ShipNoticeItem element) must apply to this specific line item. A ShipNoticeRequest for a single line item should not include Hazard elements in the ShipNoticeItem element.

quantity Attribute

Quantity specifies how many items were shipped. Expressed in units given in the UnitOfMeasure element.

lineNumber Attribute

Position, counting from 1, of the item in an order. Matches the corresponding line item, ItemOut, in the document referenced by the OrderReference element.

Packaging Element

Details about the packaging of this line item. The dimensions mentioned in the Dimension element list can appear in any order. The Packaging element contains one or more PackagingCode elements and optional Dimension element (see page 231). A particular Dimension type attribute value must not appear more than once in a Packaging element.

PackagingCode Element

Specifies one language-specific code for the packaging of the item. Values such as "pallet", "skid" and "truck load" might be appropriate for an English-based locale. The xml:lang attribute specifies the language or locale in which the PackagingCode content is written.

Dimension Element

Specifies a single dimension for the packaging of the item.

- quantity attribute
Specifies the size in this dimension. Expressed in the units given in the UnitOfMeasure element.
- type attribute
Type of dimension. Supported values include:

"length"	The length of the packaging.
"width"	The width of the packaging.
"height"	The height of the packaging.
"weight"	The weight of the packaging.
"volume"	The volume of the packaging

Hazard Element

The Hazard element provides a textual description and optional codes about hazards inherent in both an item and an overall shipment. A hazard for an entire shipment can be due to either identical hazards for all items or to hazards inherent in shipping the various products together. It can also include detailed handling requirements. There are two elements: Description, and Classification. Classification is optional and can occur more than once.

The Description element list, if provided, should include detailed handling requirements. Elements in this list can appear in any order. A description locale specified by the xml:lang attribute must not appear more than once. When more than one Description element is present, each must contain translations of a common description.

Classification elements can appear in any order. A Classification domain attribute must not appear more than once in a Hazard element.

All listed Classification elements and the Description, if provided, must relate to a single hazard. Additional hazards must use separate Hazard elements.

The following Classification domain values are expected in this context:

"UNDG "	United Nations Dangerous Goods
"IMDG"	International Marine Organization Dangerous Goods
"NAHG"	North American Hazardous Goods

OrderReference Element

The OrderReference element refers to a prior OrderRequest document. It contains a DocumentReference element.

orderID Attribute

Specifies the buyer system orderID for the ship notice, that is, the PO number. When used, it must be copied directly from the referenced OrderRequest document's OrderRequestHeader element.

orderDate Attribute

Specifies the date and time the OrderRequest was created. The date format is yyyy-mm-dd per international ISO standard 8601.

Chapter 11

Invoices

The cXML InvoiceDetail transaction enables suppliers to send invoices to buying organizations or marketplaces. This transaction supports invoice details for a wide variety of business scenarios, including standard invoices, credit memos, debit memos, and receipts.

This chapter describes:

- [Overview of Invoices](#)
- [InvoiceDetailRequest](#)
- [Response](#)
- [Invoice Status Update](#)
- [Example Invoices](#)

Overview of Invoices

Suppliers use cXML invoices to bill buying organizations or marketplaces for provided products or services. Invoices can be generated against any portion of any line items from single or multiple purchase orders. The InvoiceDetail transaction supports cancel invoices, credit memos, debit memos, and receipts.

Invoices describe purchase orders, line items, partners involved, accounting distribution, payment terms, discounts, shipping and special handling, taxes, deposit and prepayment, and remittance information.

Suppliers send invoices to commerce network hubs. Commerce network hubs route invoices to the buying organization by either querying the buying organization's ProfileResponse or by looking up routing information in the buying organization's network account.

The cXML InvoiceDetailRequest document represents an invoice. After a receiving system accepts an invoice document, it responds with a generic cXML Response.

After buying organizations begin processing invoices, they send StatusUpdateRequest documents to notify the commerce network hub about their reconciliation progress. The commerce network hub can forward these documents to suppliers.

Early InvoiceRequest Document

Previously, cXML support for invoicing was provided by the InvoiceRequest document, which contained less detail than InvoiceDetailRequest and did not support line item or summary invoices.

InvoiceRequest is deprecated in cXML 1.2.011. All cXML invoice projects should implement InvoiceDetailRequest.

Debit and Credit Amounts

In invoices, positive amounts are debits the buying organization owes the supplier; negative amounts are credits issued by the supplier to the buying organization. For example, the supplier can specify a SubtotalAmount of -50 USD to issue a credit of fifty US dollars to the buying organization. Debit can be used in both standard invoices and debit memos. Credit can be used in both standard invoices and credit memos.

For PCard-enabled purchase orders, suppliers should request payment by using the request-to-pay functionality provided by ConfirmationRequest documents (for more information, see [“ConfirmationRequest”](#) on page 206.) Suppliers should not use invoice documents in this case to request payment, but they can use them as information-only receipts.

Shipping Information

Invoices can include shipping information such as shipping charges, dates, from/to addresses, and carrier IDs. One of the reasons invoices support shipping information is because it can affect the final prices and taxes for orders shipped internationally.

The shipping information in invoices is not meant to be a substitute for sending ShipNoticeRequest documents.

Types of Invoices

InvoiceDetailRequest has the features and flexibility to support most business scenarios.

Individual and Summary Invoices

cXML supports both *individual* and *summary* invoices:

Invoice Category	Description
Individual Invoice	Applies against a single purchase order.
Summary Invoice	Applies against multiple purchase orders.

Invoice Level

cXML supports both header and detailed invoices:

Invoice Level	Description
Header Invoice	Applies against the entirety of one or more purchase orders, without describing their line items. Specify isHeaderInvoice="yes" and use InvoiceDetailHeaderOrder elements, which do not contain line-item information.
Detailed Invoice	(Line-item level invoice) Applies against specific line items from one or more purchase orders. Leave out isHeaderInvoice and use InvoiceDetailOrder elements, which contain line-item information.

Invoice Purpose

Use the InvoiceDetailRequestHeader attributes to specify the purpose of the invoice.

Invoice Purpose	Description
Standard Invoice	Request for payment after providing products or services. Specify purpose="standard" and operation="new".
Credit Memo	Specifies credit to a buying organization. Specify purpose="creditMemo" and operation="new". Must be a header invoice. Amounts must be negative.
Debit Memo	Specifies debit to a buying organization. Specify purpose="debitMemo" and operation="new". Must be a header invoice. Amounts must be positive.
Information Only	Provides a record of charges, similar to a receipt. No action is expected. Specify isInformationOnly="yes" and operation="new".
Cancel Invoice	Cancels a previously sent invoice. Specify operation="delete".

Invoice DTD

The cXML standard uses multiple DTDs to optimize the performance of validating parsers. The InvoiceDetail transaction is defined in a separate DTD named InvoiceDetail.dtd, available at:

<http://xml.cXML.org/schemas/cXML/<version>/InvoiceDetail.dtd>

InvoiceDetailRequest

InvoiceDetailRequest documents represent invoices.

The structure of the InvoiceDetailRequest document is:

```
<Request>
  <InvoiceDetailRequest>
    <InvoiceDetailRequestHeader>
      header information
    </InvoiceDetailRequestHeader>
    <InvoiceDetailHeaderOrder>
      order-level invoice information
    </InvoiceDetailHeaderOrder>
    . . .
  or
    <InvoiceDetailOrder>
      detailed line-item information
    </InvoiceDetailOrder>
    . . .
    <InvoiceDetailSummary>
      invoice summary
    </InvoiceDetailSummary>
  </InvoiceDetailRequest>
</Request>
```

InvoiceDetailOrder elements are for detailed (line-item level) invoices and InvoiceDetailHeaderOrder elements are for header invoices. Invoices must not contain both types of elements. Both types of elements contain *invoice lines*.

All invoice line level amounts must add up to the total specified in InvoiceDetailSummary.

InvoiceDetailRequestHeader

Defines header information that applies to the entire invoice.

InvoiceDetailHeader has the following attributes:

invoiceID	A supplier-generated identifier for the Invoice. Identical to the Invoice Number that appears at the top of a physical Invoice.
isInformationOnly	Indicates whether the buying organization needs to take action: yes — Invoice is for the buying organization's information only (no action needs to be taken by the buying organization). Not specified — (default) Invoice is functional. The buying organization needs to take action upon receiving this document (submit payment or accept credit).
purpose	Purpose of the invoice: standard — (default) A standard billing statement from the supplier to the buying organization. creditMemo — A credit memo for issuing credit to the buying organization. isHeaderInvoice must be yes. Also, the element InvoiceDetailSummary/DueAmount must be a negative amount. debitMemo — A debit memo for billing a balance owed by the buying organization. isHeaderInvoice must be yes. Also, the element InvoiceDetailSummary/DueAmount must be a positive amount.
operation	How this document is acting on the invoice: new — (default) Creates a new invoice. delete — Cancels an existing invoice. The PayloadID of the existing invoice must be specified in a DocumentReference.
invoiceDate	Date and time Invoice was created (should be earlier than the cXML timestamp).

InvoiceDetailHeaderIndicator

Defines indicators that describe overall attributes of the invoice. By default, all indicators are false.

InvoiceDetailHeaderIndicator has the following attributes:

isHeaderInvoice	Category of the invoice: yes — Header invoice. Invoice uses InvoiceDetailHeaderOrder, which contains header level invoice information without item details Not specified — Detail invoice. Invoice uses InvoiceDetailOrder, which contains item details.
isVatRecoverable	yes — The entire invoice is VAT (Value Added Tax)-recoverable.

InvoiceDetailLineIndicator

Indicates the presence of invoicing details at invoice line level (in InvoiceDetailItem, InvoiceDetailServiceItem, or InvoiceDetailOrderSummary). By default, all indicators are false.

If this element indicates that invoicing details exist at invoice line level, invoice lines that do not provide such information are assumed to have values of zero, or “not available” for that information.

InvoiceDetailLineIndicator has the following attributes:

isTaxInLine	yes — Tax (Tax) is provided at invoice line level.
isSpecialHandlingInLine	yes — Special handling (InvoiceDetailLineSpecialHandling) is provided at invoice line level.
isShippingInLine	yes — Shipping (InvoiceDetailLineShipping) is provided at invoice line level.
isDiscountInLine	yes — Discount (InvoiceDetailDiscount) is provided at invoice line level.
isAccountingInLine	yes — Accounting distribution (Distribution) is provided at invoice line level. If isHeaderInvoice is true, this indicator must not be specified, because Distribution is available only at item level.

InvoicePartner

Defines a party involved in invoicing, including the issuer of the invoice and the person sold to.

Invoices support InvoicePartner because the Contact element alone does not support the wide variety of reference identifiers involved in invoicing.

Do not use this element to specify ship from or ship to; instead, use InvoiceDetailShipping.

Contact

Contact information of the invoice partner. Allowed contact roles are issuerOfInvoice, soldTo, billTo, remitTo.

The from role has been deprecated.

IdReference

Defines an ID reference. The identifier/domain pair should be unique within each trading partner relationship (a buying organization and a supplier).

IdReference has the following attributes:

identifier	The unique identifier of the IdReference within the domain.		
domain	The domain of the IdReference. One of the following values: <div><div>accountID</div><div>federalTaxID</div><div>provincialTaxID</div><div>taxExemptionIS</div></div> <div><div>bankRoutingID</div><div>stateTaxID</div><div>vatID</div></div> <div><div>accountPayableID</div><div>accountReceivableID</div><div>gstID</div></div>		
	Values can be application-specific, such as 1099ID or courtRegisterID.		
	supplierTaxID has been deprecated and is treated as federalTaxID.		

Creator

The creator of the IdReference (for example, the name of the bank, shipper, or other organization).

Description

Textual description of the IdReference for human readability.

DocumentReference

Identifies an earlier InvoiceDetailRequest document. If operation="delete", DocumentReference is required and it must reference the original InvoiceDetailRequest document (with operation="new"). In all other situations, DocumentReference is optional.

DocumentReference has the following attribute:

payloadID	The payloadID attribute of another cXML document.
------------------	---

InvoiceDetailShipping

The shipping details of the invoice.

InvoiceDetailShipping has the following attribute:

shippingDate	The date and time this shipment leaves the supplier.
---------------------	--

Contact

The ship from and ship to addresses. Both ship from and ship to must be specified.

Contact has the following attributes:

role	The role name of the contact. Allowed roles are shipFrom, shipTo, and carrierCorporate.
addressID	Unique identifier for the contact's address.

Name

Name of the contact

PostalAddress

Postal address of the contact

Email

Email address of the contact

Email has the following attributes:

preferredLang	The language in which the recipient prefers to receive emails. This attribute is optional. See “xmlLangCode” on page 54 for more information.
----------------------	---

Phone

Telephone number of the contact

Fax

Fax number of the contact

URL

Web address of the contact

CarrierIdentifier

This list can include multiple identifiers for the same carrier. Elements in this list can appear in any order. An identification domain (CarrierIdentifier domain) must not appear more than once in an InvoiceDetailShipping element. All identification provided by elements of one CarrierIdentifier list must correspond to the same company.

CarrierIdentifier has the following attribute:

domain	Domain for this value. Recognized domains include: companyName - The legal name for this company. In some cases, this could also be provided in a Contact element with role "carrierCorporate". That option should be reserved for cases in which additional detail about the carrier appears in this element. SCAC - Standard Carrier Alpha Code (www.nmfta.org/scac2.htm) IATA - International Air Transport Association (www.iata.org) AAR - Association of American Railroads (www.aar.org) UIC - International Union of Railways (www.uic.asso.fr) EAN - European Article Numbering (www.ean-ucc.org) DUNS - D&B's Data Universal Numbering System (www.dnb.com)
---------------	---

ShipmentIdentifier

The tracking number of this shipment.

DocumentReference

Identifies an earlier ShipNoticeRequest.

For more information, see “[DocumentReference](#)” on page 240.

InvoiceDetailPaymentTerm (deprecated)

InvoicedetailPaymentTerm is deprecated in cXML 1.2.011, in favor of [PaymentTerm](#).

PaymentTerm

Defines a payment term in an invoice or order. PaymentTerm defines either the net term (without discount) or the discount term (with discount).

PaymentTerm has the following attributes:

payInNumberOfDays	The number of days after invoice date to pay in full.
--------------------------	---

Discount

The percentage or amount of the discount term. The discount rate applies if the invoice total is paid within the time specified by payInNumberOfDays. Positive rates denote discounts and negative rates denote penalties. Do not use a percentage sign (%) or divide by 100; for example “2” means 2%.

Do not use the Discount element if the PaymentTerm is a net term.

Period

The period over which the services were rendered.

Period has the following attributes:

startDate	The starting date of the service.
endDate	The ending date of the service.

InvoiceDetailOrder

Defines the invoice information of an order with item details, used only when isHeaderInvoice is false (not specified).

An invoice line is an InvoiceDetailItem or an InvoiceDetailServiceItem and its invoice line number is specified by the invoiceLineNumber attribute.

InvoiceDetailOrderInfo

Defines information related to the corresponding purchase order, including order reference and related master agreement reference, if any. Applications use this information to match the invoice with the corresponding purchase order or master agreement. The more definitive the reference, the more likely applications can successfully perform document matching.

InvoiceDetailOrderInfo can contain several possible elements for referring to documents. OrderReference is strongly recommended, but if that information is not available, use MasterAgreementReference, MasterAgreementIDInfo, OrderIDInfo, or SupplierOrderInfo, in that order.

OrderReference

The reference to the purchase order being invoiced.

MasterAgreementReference

Defines a reference to an earlier MasterAgreementRequest document. This element identifies the master agreement of the release order to be invoiced.

MasterAgreementReference has the following attributes:

agreementID	The ID number of a master agreement known to the buying organization's system.
agreementDate	The date and time the master agreement request was created.

MasterAgreementIDInfo

Defines the buying organization's ID number of the corresponding master agreement if the order being invoiced is a release. This element identifies the master agreement of the contract or release order to be invoiced.

MasterAgreementIDInfo has the following attributes:

agreementID	The ID number of a master agreement known to the buying organization's system.
agreementDate	The date and time the master agreement request was created.

OrderIDInfo

Identifies a purchase order known to the buying organization.

OrderIDInfo has the following attributes:

orderID	The ID of a purchase order (purchase order number) known to the buying organization.
orderDate	The date and time the purchase order was created.

SupplierOrderInfo

Defines supplier sales order information related to a purchase order.

SupplierOrderInfo has the following attribute:

orderID	Supplier sales order ID of the purchase order.
----------------	--

InvoiceDetailItem

Defines an invoice line item.

The buying organization might require information provided here to match the information provided in the purchase order. For example, the buying organization might require there to be no change in the UnitOfMeasure value.

InvoiceDetailItem has the following attributes:

invoiceLineNumber	Supplier defined ID for the current invoice line. Should be unique across all invoice lines within an invoice.
quantity	The quantity being invoiced for the line item.
inspectionDate	The date when the transfer of goods or the delivery of services occurs according to legal tax definitions. The usage of this attribute is optional in most cases, and must be defined by the trading partners involved in the transaction.

UnitOfMeasure

The line item's unit of measure.

UnitPrice

The unit price.

InvoiceDetailItemReference

Defines all references related to an invoice line item.

InvoiceDetailItemReference has the following attributes:

lineNumber	The purchase order line number of current line item, copied from the OrderRequest.
serialNumber (deprecated)	The product serial number for the current line item. This attribute was deprecated in cXML 1.2.009. Use SerialNumber elements, instead.

ItemID

The supplier part number of current line item, from the OrderRequest.

Description

The line item description, from the OrderRequest.

SerialNumber

A serial number that uniquely identifies an accountable item that is being invoiced.

You can include multiple SerialNumber elements; the number of SerialNumber elements should match the invoice item quantity.

Use SerialNumber elements instead of the InvoiceDetailItemReference serialNumber attribute, which was deprecated in cXML 1.2.009.

ManufacturerPartID

The manufacturer part number.

ManufacturerName

The name of the manufacturer.

Country

The country of origin of the product listed in the line item.

SubtotalAmount

The invoice subtotal of the current line item: UnitPrice times quantity.

Tax

The tax for the line item. Ignored if isTaxInLine is false (not specified).

Money

The amount owed for tax.

Description

Textual description of the tax.

TaxDetail

Detailed information about the tax.

TaxDetail has the following attributes:

purpose	The purpose of the tax. For example, "tax" or "custom duty".
category	The category of the tax. For example, "sales", "usage", "vat", or "gst".
percentageRate	The tax rate percentage. Do not include a percent symbol (%).
isVatRecoverable	Set to "true" if the tax amount is recoverable.
taxPointDate	The date on which VAT becomes due
paymentDate	The date on which payment must be made (used only for transactions in France).
isTriangularTransaction	Set to "true" to indicate that the transaction occurred between three parties in three different countries, but the movement of goods did not follow the invoicing route. Add a Contract element with role="subsequentBuyer" to identify the subsequent buying organization in triangular transactions.

TaxableAmount

The amount that is taxable.

TaxAmount

The amount of tax.

TaxLocation

The locale in which the tax applies.

Description

Textual description of the tax.

TriangularTransactionLawReference

Reference to the relevant EU law covering the VAT for triangular transactions. For example, “VAT - EC Article 28 Simplification Invoice”.

InvoiceDetailLineSpecialHandling

The special handling information for the line item. Ignored if `isSpecialHandlingInLine` is false (not specified).

InvoiceDetailLineShipping

The shipping information for the line item. Ignored if `isShippingInLine` is false (not specified).

GrossAmount

The `SubtotalAmount` plus taxes, shipping, and special handling charges for the line item.

InvoiceDetailDiscount

The discount for the line item. Ignored if `isDiscountInLine` is false (not specified).

NetAmount

The `GrossAmount` minus discounts for the line item.

Distribution

Accounting information generated by the buying organization, such as cost center or general ledger category. This information should be copied from the `OrderRequest`. Ignored if `isAccountingInLine` is false (not specified).

Comments

Textual comments for the line item.

Extrinsic

Additional information related to the line item. Should not duplicate anything in InvoiceDetailItem or InvoiceDetailOrder.

InvoiceDetailServiceItem

Specifies a service being invoiced.

InvoiceDetailServiceItem has the following attributes:

invoiceLineNumber	Supplier defined ID for the current invoice line. Should be unique across all invoice lines within an invoice.
quantity	The quantity being invoiced for the line item. For service items, quantity represents the number of units of service rendered. For example, 2 hours of service, where UnitOfMeasure is "HUR"
referenceDate	The reference date for the service item. This can indicate the date at which the service line item is being invoiced.
inspectionDate	The date when the transfer of goods or the delivery of services occurs according to legal tax definitions. The usage of this attribute is optional in most cases, and must be defined by the trading partners involved in the transaction.

InvoiceDetailServiceItemReference

InvoiceDetailServiceItemReference has the following attribute:

lineNumber	The line number of current line item on the master agreement. This value is required if the item being invoiced is part of a detailed master agreement that specifies detailed pricing terms at the line item or commodity level. It is optional if the item being invoiced is part of a master supplier agreement or blanket purchase order which do not contain detailed line item pricing information.
-------------------	---

Classification

Commodity classification of the service.

ItemID

The Supplier's part number for the service

Description

Description of the service.

SubtotalAmount

The subtotal amount of the service item. If unit price and invoiced quantity are specified, then subtotal should be the product of them.

Period

The period during which the service was rendered.

For more information, see [“Period”](#) on page 242.

UnitRate

The rate at which the service item is charged. In cXML version 2.1.011 or later, use the `UnitRate` element rather than `UnitOfMeasure` and `UnitPrice`, because `UnitRate` includes the rate code. For some services, such as temporary labor, `UnitRate` is required.

`UnitRate` represents the amount to be paid per unit of time (or of some other measure). In the case of multiple `UnitRates`, each `UnitRate` should include a `TermReference` to distinguish it from others. `TermReference` is a generic base element that identifies the definition of the `UnitRate` in question. See [“UnitRate”](#) on page 150.

UnitOfMeasure (deprecated)

`UnitOfMeasure` is deprecated in cXML 1.2.011, and should not be used in new cXML documents. Use `UnitRate` instead. `UnitOfMeasure` is the unit of measure for the service. For example, HUR for per hour or MON for per month.

UnitPrice (deprecated)

`UnitPrice` is deprecated in cXML 1.2.011, and should not be used in new cXML documents. Use `UnitRate` instead. `UnitPrice` is the price, per unit of measure.

Tax

The tax for the line item. Ignored if `isTaxInLine` is false (not specified).

For more information, see [“Tax”](#) on page 246

GrossAmount

The SubtotalAmount plus taxes, shipping, and special handling charges for the line item.

InvoiceDetailDiscount

The discount for the line item. Ignored if isDiscountInLine is false (not specified).

NetAmount

The GrossAmount minus discounts for the line item.

Distribution

Accounting information generated by the buying organization, such as cost center or general ledger category. This information should be copied from the OrderRequest. Ignored if isAccountingInLine is false (not specified).

Comments

Textual comments for the line item.

InvoiceLaborDetail

Contains information about an item related to temporary labor. I

Contractor

The contractor whose work is being invoiced.

JobDescription

A text description of the job being performed.

Supervisor

Specifies contact information for the person who supervises the contractor.

WorkLocation

The address of the place where the work is performed.

InvoiceTimeCardDetail

Invoice details about a temporary labor service. The pay code for this invoice line item is in the UnitRate of the containing InvoiceDetailServiceItem.

TimeCardReference

Provides a clear reference to a prior TimeCard cXML document.
TimeCardReference has the following attribute:

timeCardID	Unique ID for the timecard as sent on the TimeCard document during TimeCardInfoRequest or TimeCardRequest. For more information, see Chapter 8, "TimeCard Transaction."
------------	---

TimeCardIDInfo

Defines the unique ID of the timecard known to the buyer and supplier systems. TimeCardIDInfo has the following attribute:

timeCardID	Unique ID for the timecard as sent on the TimeCard document during TimeCardInfoRequest or TimeCardRequest. For more information, see Chapter 8, "TimeCard Transaction."
------------	---

Extrinsic

Additional information related to this line item. Do not duplicate information in InvoiceDetailItem or InvoiceDetailOrder.

Use Extrinsic elements to specify line item related attributes such as service location, overtime/regular, and union/non- union.

For simple attributes such as overtime/regular, use a simple name, value pair, for example:

```
<Extrinsic name="serviceType">Temporary</Extrinsic>.
```

For structured attributes such as service location, use a structured element, for example:

```
<Extrinsic name="serviceLocation">
  <Contact role="serviceLocation">
    <Name>XYZ Inc</Name>
    <PostalAddress>
      <Street>123 Easy St</Street>
      <City>Sunnyvale</City>
      <State>California</State>
      <PostalCode>94089</PostalCode>
      <Country isoCountryCode="US">USA</Country>
    </PostalAddress>
  </Contact>
</Extrinsic>
```

InvoiceDetailHeaderOrder

Defines the header invoice information of a purchase order, without item details, used only when isHeaderInvoice="yes".

In this case, an invoice line is an InvoiceDetailHeaderOrder and its invoice line number is specified by the invoiceLineNumber attribute.

InvoiceDetailOrderInfo

Defines information related to the corresponding purchase order.

For more information, see “InvoiceDetailOrderInfo” on page 243

InvoiceDetailOrderSummary

Defines header level summary info of an order in an invoice line.

InvoiceDetailOrderSummary has the following attribute:

invoiceLineNumber	Supplier defined ID for the current invoice line. It should be unique across all invoice lines of the same InvoiceDetailRequest.
inspectionDate	The date when the transfer of goods or the delivery of services occurs according to legal tax definitions. The usage of this attribute is optional in most cases, and must be defined by the trading partners involved in the transaction.

SubtotalAmount

The invoice subtotal of the this order.

Period

The period over which the services were rendered.

For more information, see “Period” on page 242.

Tax

The tax for this order. Ignored if isTaxInLine is false (not specified).

For more information, see “Tax” on page 246.

InvoiceDetailLineSpecialHandling

The special handling information for this order. Ignored if isSpecialHandlingInLine is false (not specified).

InvoiceDetailLineShipping

The shipping information for this invoice line. Ignored if isShippingInLine is false (not specified).

InvoiceDetailShipping

The shipping details. For more information, see “[InvoiceDetailShipping](#)” on page 240.

Money

The shipping amount.

GrossAmount

The SubtotalAmount plus taxes, shipping, and special handling charges.

InvoiceDetailDiscount

The discount or penalty for this invoice line. Ignored if isDiscountInLine is false (not specified).

InvoiceDetailDiscount has the following attribute:

percentageRate	Discount or penalty rate percentage. Positive rates denote discounts and negative rates denote penalties. Do not include a percentage sign (%) or divide by 100; for example “2” means 2%.
-----------------------	--

NetAmount

The GrossAmount minus discount amount.

Comments

Textual comments for the line item.

Extrinsic

Additional information related to the line item. Should not duplicate anything in InvoiceDetailOrderSummary or InvoiceDetailHeaderOrder.

InvoiceDetailSummary

Defines the summary information of an invoice.

SubtotalAmount

Sum of line item quantities multiplied by unit price.

Tax

Total tax information.

For more information, see [“Tax”](#) on page 246.

SpecialHandlingAmount

Total special handling charge.

ShippingAmount

Total shipping charge.

GrossAmount

Sum of subtotal, taxes, special handling charges, and shipping charges, before discounts.

InvoiceDetailDiscount

The total discount or penalty applied in the invoice.

For more information, see [“InvoiceDetailDiscount”](#) on page 253.

NetAmount

Total GrossAmount minus discounts.

DepositAmount

Total deposit or prepayment amount.

DueAmount

Total amount due and payable: NetAmount minus DepositAmount. If purpose="creditMemo", this amount must be negative. If purpose="debitMemo", this amount must be positive.

Response

Immediately after receiving an invoice, the receiving system should respond with a generic cXML Response document, for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/
InvoiceDetail.dtd">
<cXML timestamp="2001-10-31T23:07:22-08:00" payloadID="1004598442900-
8367273815197467070@10.10.13.100">
  <Response>
    <Status code="201" text="Accepted">Acknowledged</Status>
  </Response>
</cXML>
```

For a list of possible status codes, see “[Status](#)” on page 45.

Invoice Status Update

After buying organizations receive invoices, they can perform reconciliation to match the charges within them to amounts within purchase orders or master agreements. They can then set invoice status to indicate whether charges reconciled successfully.

Buying organizations update the status of invoices by sending StatusUpdateRequest documents to commerce network hubs, which can forward them to suppliers.

StatusUpdateRequest documents for invoices contain InvoiceStatus elements. Invoice status can be processing, reconciled, rejected, paying, or paid, which refers to the action taken by the buying organization on the invoice:

processing	The invoice was received by the buying organization and is being processed.
------------	---

reconciled	The invoice reconciled. The amounts in the invoice have not yet been paid.
rejected	The invoice failed to reconcile. The buying organization is rejecting the invoice. The Comments element should contain free text explaining why the invoice was rejected, and the actions the supplier should take. The supplier can resubmit a corrected invoice (a new invoice document with a new invoice number).
paying	The invoice is in the payment process or has been partially paid.
paid	The invoice amounts have been paid by the buying organization.

The PartialAmount element enables buying organizations to specify different amounts paid than the amounts specified in invoices. PartialAmount should not appear for invoices that are paid in full. The existence of PartialAmount alerts the supplier to read the Comments elements, which should explain the differences.

The DocumentReference within the StatusUpdateRequest must refer to the InvoiceDetailRequest document. The Status element should have status code 200.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML timestamp="2001-09-05T16:34:28-07:00" payloadID="999732868377--
681956365911302107@10.11.128.161">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>jill@buyerorg.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>jack@supplierorg.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>jill@buyerorg.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Procurement Application V1.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <StatusUpdateRequest>
      <DocumentReference payloadID="Inv123"></DocumentReference>
      <Status code="200" text=""></Status>
    </StatusUpdateRequest>
  </Request>
</cXML>
```

```

    <InvoiceStatus type="paid">
      <PartialAmount>
        <Money currency="USD">10.99</Money>
      </PartialAmount>
      <Comments>This charge is paid, minus $2.00 due to missing items.</Comments>
    </InvoiceStatus>
  </StatusUpdateRequest>
</Request>
</cXML>

```

For more information about the StatusUpdate transaction, see [“StatusUpdateRequest”](#) on page 202.

Example Invoices

The following examples illustrate several types of invoices.

- [Standard Header Invoice](#)
- [Standard Detail Invoice](#)
- [Service Invoice](#)
- [Marketplace Invoice](#)

Standard Header Invoice

This example shows a header invoice against a single purchase order.

```

<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE cXML SYSTEM "http://
xml.cXML.org/schemas/cXML/1.2.011/InvoiceDetail.dtd">
<cXML timestamp="2001-10-10T16:23:01-07:00" payloadID="Oct102001_0447pm">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserID">
        <Identity>jack@supplierorg.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserID">
        <Identity>jill@buyerorg.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserID">
        <Identity>jack@supplierorg.com</Identity>

```

```

    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Supplier's Super Invoice Generator</UserAgent>
</Sender>
</Header>
<Request>
  <InvoiceDetailRequest>
    <InvoiceDetailRequestHeader invoiceDate="2001-10-09T00:00:00-07:00"
      invoiceID="Oct102001_0447pm" purpose="standard" operation="new">
      <InvoiceDetailHeaderIndicator isHeaderInvoice="yes" />
      <InvoiceDetailLineIndicator isTaxInLine="yes" isShippingInLine="yes"
        isSpecialHandlingInLine="yes" isDiscountInLine="yes" />
      <InvoicePartner>
        <Contact role="billTo">
          <Name xml:lang="en-US">Buyer Headquarters</Name>
          <PostalAddress>
            <Street>111 Main Street</Street>
            <City>Anytown</City>
            <State>CA</State>
            <PostalCode>94089</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
        </Contact>
      </InvoicePartner>
      <InvoicePartner>
        <Contact role="remitTo">
          <Name xml:lang="en-US">Supplier Accts. Receivable</Name>
          <PostalAddress>
            <Street>One Bank Avenue</Street>
            <City>Any City</City>
            <State>CA</State>
            <PostalCode>94087</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Contact>
            <IdReference identifier="123456789" domain="bankRoutingID" />
            <IdReference identifier="3456" domain="accountID" />
          </Contact>
        </InvoicePartner>
        <Comments xml:lang="en-US">This is an invoice for DO789</Comments>
      </InvoiceDetailRequestHeader>
      <InvoiceDetailHeaderOrder>
        <InvoiceDetailOrderInfo>
          <OrderReference>
            <DocumentReference payloadID="99576652.982.090.136" />
          </OrderReference>
        </InvoiceDetailOrderInfo>
        <InvoiceDetailOrderSummary invoiceLineNumber="1">
          <SubtotalAmount>
            <Money currency="USD">5000.00</Money>
          </SubtotalAmount>
        </InvoiceDetailOrderSummary>
      </InvoiceDetailHeaderOrder>
    </InvoiceDetailRequest>
  </Request>
</Invoice>

```

```

<Tax>
  <Money currency="USD">500.00</Money>
  <Description xml:lang="en-US">State Tax</Description>
</Tax>
<InvoiceDetailLineSpecialHandling>
  <Money currency="USD">110.00</Money>
</InvoiceDetailLineSpecialHandling>
<InvoiceDetailLineShipping>
  <InvoiceDetailShipping>
    <Contact role="shipFrom" addressID="1000487">
      <Name xml:lang="en">Main Shipping Dock</Name>
      <PostalAddress name="default">
        <Street>15 Oak Road</Street>
        <City>Bigtown</City>
        <State>CA</State>
        <PostalCode>95032</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
      <Email name="default">shipper@supplierorg.com</Email>
      <Phone name="work">
        <TelephoneNumber>
          <CountryCode isoCountryCode="US">1</CountryCode>
          <AreaOrCityCode>888</AreaOrCityCode>
          <Number>1234567</Number>
        </TelephoneNumber>
      </Phone>
    </Contact>
    <Contact role="shipTo" addressID="1000487">
      <Name xml:lang="en">Main Receiving</Name>
      <PostalAddress name="default">
        <DeliverTo>Jason Lynch</DeliverTo>
        <Street>77 Nowhere Street</Street>
        <City>Industrial Town</City>
        <State>CA</State>
        <PostalCode>95035</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
      <Email name="default">jlynch@buyerorg.com</Email>
      <Phone name="work">
        <TelephoneNumber>
          <CountryCode isoCountryCode="US">1</CountryCode>
          <AreaOrCityCode>999</AreaOrCityCode>
          <Number>3582000</Number>
        </TelephoneNumber>
      </Phone>
    </Contact>
  </InvoiceDetailShipping>
  <Money currency="USD">200.00</Money>
</InvoiceDetailLineShipping>
<GrossAmount>

```

```

        <Money currency="USD">5810.00</Money>
      </GrossAmount>
      <InvoiceDetailDiscount percentageRate="10">
        <Money currency="USD">581.00</Money>
      </InvoiceDetailDiscount>
      <NetAmount>
        <Money currency="USD">5229.00</Money>
      </NetAmount>
      <Comments>This a Standard Header Level Invoice</Comments>
    </InvoiceDetailOrderSummary>
  </InvoiceDetailHeaderOrder>
</InvoiceDetailSummary>
<SubtotalAmount>
  <Money currency="USD">5000.00</Money>
</SubtotalAmount>
<Tax>
  <Money currency="USD">500.00</Money>
  <Description xml:lang="en-US">State Tax</Description>
</Tax>
<SpecialHandlingAmount>
  <Money currency="USD">110.00</Money>
</SpecialHandlingAmount>
<ShippingAmount>
  <Money currency="USD">200.00</Money>
</ShippingAmount>
<GrossAmount>
  <Money currency="USD">5810.00</Money>
</GrossAmount>
<InvoiceDetailDiscount percentageRate="10">
  <Money currency="USD">581.00</Money>
</InvoiceDetailDiscount>
<NetAmount>
  <Money currency="USD">5229.00</Money>
</NetAmount>
<DepositAmount>
  <Money currency="USD">1000.00</Money>
</DepositAmount>
<DueAmount>
  <Money currency="USD">4229.00</Money>
</DueAmount>
</InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>

```

Standard Detail Invoice

This example shows a detail invoice for two line items in a single purchase order. It contains payment terms that define discounts for early payment and penalties for late payment. It also contains the buying organization's accounting information copied from the purchase order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/
InvoiceDetail.dtd">
<cXML payloadID="Oct102001_1204pm" timestamp="2001-04-20T23:59:45-07:00">
  <Header>
    From, To, and Sender credentials
  </Header>
  <Request>
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader invoiceID="Oct102001_1204pm"
        purpose="standard" operation="new"
        invoiceDate="2001-04-20T23:59:20-07:00">
        <InvoiceDetailHeaderIndicator/>
        <InvoiceDetailLineIndicator isTaxInLine="yes" isShippingInLine="yes"
          isAccountingInLine="yes"/>
        <InvoicePartner>
          Sell To contact information
        </InvoicePartner>
        <InvoicePartner>
          Remit To contact information
        </InvoicePartner>
        <PaymentTerm payInNumberOfDays="10">
          <Discount>10</Discount>
        </PaymentTerm>
        <PaymentTerm payInNumberOfDays="20">
          <Discount>5</Discount>
        </PaymentTerm>
        <PaymentTerm payInNumberOfDays="30">
          <Discount>0</Discount>
        </PaymentTerm>
        <PaymentTerm payInNumberOfDays="40">
          <Discount>-5</Discount>
        </PaymentTerm>
        <PaymentTerm payInNumberOfDays="50">
          <Discount>-9</Discount>
        </PaymentTerm>
      </InvoiceDetailRequestHeader>
      <InvoiceDetailOrder>
        <InvoiceDetailOrderInfo>
          <OrderReference>
            <DocumentReference payloadID="99576652.982.090.136"/>
          </OrderReference>

```

```

<MasterAgreementReference>
  <DocumentReference payloadID="99576652.980.000.423"/>
</MasterAgreementReference>
<SupplierOrderInfo orderID="DO1234"></SupplierOrderInfo>
</InvoiceDetailOrderInfo>
<InvoiceDetailItem invoiceLineNumber="1" quantity="1">
  <UnitOfMeasure>EA</UnitOfMeasure>
  <UnitPrice><Money currency="USD">15.40</Money></UnitPrice>
  <InvoiceDetailItemReference lineNumber="1">
    <ItemID>
      <SupplierPartID>TEX08134</SupplierPartID>
    </ItemID>
    <Description xml:lang="en">
      Texas Instruments Superview Calculator - 12-Digit Print/Display
    </Description>
    <SerialNumber>45993823469876</SerialNumber>
  </InvoiceDetailItemReference>
  <SubtotalAmount>
    <Money currency="USD">15.40</Money>
  </SubtotalAmount>
  <Tax>
    <Money currency="USD">1.54</Money>
    <Description xml:lang="en">total item tax</Description>
    <TaxDetail purpose="tax" category="sales" percentageRate="8">
      <TaxableAmount>
        <Money currency="USD">15.40</Money>
      </TaxableAmount>
      <TaxAmount>
        <Money currency="USD">1.23</Money>
      </TaxAmount>
      <TaxLocation xml:lang="en">CA</TaxLocation>
    </TaxDetail>
    <TaxDetail purpose="tax" category="sales" percentageRate="2">
      <TaxableAmount>
        <Money currency="USD">15.40</Money>
      </TaxableAmount>
      <TaxAmount>
        <Money currency="USD">0.31</Money>
      </TaxAmount>
      <TaxLocation xml:lang="en">US</TaxLocation>
    </TaxDetail>
  </Tax>
</InvoiceDetailLineShipping>
  <InvoiceDetailShipping>
    Ship From and Ship To contact information
  </InvoiceDetailShipping>
  <Money currency="USD">2.00</Money>
</InvoiceDetailLineShipping>
<GrossAmount>
  <Money currency="USD">18.94</Money>

```



```

</GrossAmount>
<NetAmount>
  <Money currency="USD">18.94</Money>
</NetAmount>
<Distribution>
  <Accounting name="Buyer assigned accounting code 15">
    <AccountingSegment id="ABC123456789">
      <Name xml:lang="en">Purchase</Name>
      <Description xml:lang="en">Production Control</Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">18.94</Money>
  </Charge>
</Distribution>
<Distribution>
  <Accounting name="Buyer assigned accounting code 16">
    <AccountingSegment id="ABC000000001">
      <Name xml:lang="en">Trade</Name>
      <Description xml:lang="en">Misc (Expensed)</Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">18.94</Money>
  </Charge>
</Distribution>
</InvoiceDetailItem>
<InvoiceDetailItem invoiceLineNumber="2" quantity="1">
  <UnitOfMeasure>PK</UnitOfMeasure>
  <UnitPrice><Money currency="USD">4.95</Money></UnitPrice>
  <InvoiceDetailItemReference lineNumber="2">
    <ItemID>
      <SupplierPartID>PENCIL123</SupplierPartID>
    </ItemID>
    <Description xml:lang="en">
      One dozen wood #2 pencils with eraser
    </Description>
  </InvoiceDetailItemReference>
  <SubtotalAmount>
    <Money currency="USD">4.95</Money>
  </SubtotalAmount>
  <Tax>
    <Money currency="USD">0.50</Money>
    <Description xml:lang="en">total item tax</Description>
    <TaxDetail purpose="tax" category="sales" percentageRate="8">
      <TaxableAmount>
        <Money currency="USD">0.40</Money>
      </TaxableAmount>
      <TaxAmount>
        <Money currency="USD">4.95</Money>
      </TaxAmount>
    </TaxDetail>
  </Tax>
</InvoiceDetailItem>

```

```

        </TaxAmount>
        <TaxLocation xml:lang="en">CA</TaxLocation>
    </TaxDetail>
    <TaxDetail purpose="tax" category="sales" percentageRate="2">
        <TaxLocation xml:lang="en">US</TaxLocation>
        <TaxableAmount>
            <Money currency="USD">4.95</Money>
        </TaxableAmount>
        <TaxAmount>
            <Money currency="USD">0.10</Money>
        </TaxAmount>
    </TaxDetail>
</Tax>
<InvoiceDetailLineShipping>
    <InvoiceDetailShipping>
        Ship From and Ship To contact information
    </InvoiceDetailShipping>
    <Money currency="USD">1.00</Money>
</InvoiceDetailLineShipping>
<GrossAmount>
    <Money currency="USD">6.45</Money>
</GrossAmount>
<NetAmount>
    <Money currency="USD">6.45</Money>
</NetAmount>
</InvoiceDetailItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>
    <SubtotalAmount>
        <Money currency="USD">20.35</Money>
    </SubtotalAmount>
    <Tax>
        <Money currency="USD">2.04</Money>
        <Description xml:lang="en">total tax</Description>
        <TaxDetail purpose="tax" category="sales" percentageRate="8">
            <TaxableAmount>
                <Money currency="USD">20.35</Money>
            </TaxableAmount>
            <TaxAmount>
                <Money currency="USD">1.63</Money>
            </TaxAmount>
            <TaxLocation xml:lang="en">CA</TaxLocation>
        </TaxDetail>
        <TaxDetail purpose="tax" category="sales" percentageRate="2">
            <TaxableAmount>
                <Money currency="USD">20.35</Money>
            </TaxableAmount>
            <TaxAmount>
                <Money currency="USD">0.41</Money>
            </TaxAmount>
        </TaxDetail>
    </Tax>

```

```

        <TaxLocation xml:lang="en">US</TaxLocation>
      </TaxDetail>
    </Tax>
    <ShippingAmount>
      <Money currency="USD">3.00</Money>
    </ShippingAmount>
    <GrossAmount>
      <Money currency="USD">25.39</Money>
    </GrossAmount>
    <NetAmount>
      <Money currency="USD">25.39</Money>
    </NetAmount>
    <DueAmount>
      <Money currency="USD">25.39</Money>
    </DueAmount>
  </InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>

```

Service Invoice

The following invoice is for both regular items and service items.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/
InvoiceDetail.dtd">
<cXML payloadID="combo-1@supplier.com" timestamp="2001-04-20T23:59:45-07:00">
  <Header>
    From, To, and Sender credentials
  </Header>
  <Request deploymentMode="test">
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader
        invoiceID="123456"
        purpose="standard"
        operation="new"
        invoiceDate="2001-04-20T23:59:20-07:00">
      <InvoiceDetailHeaderIndicator/>
      <InvoiceDetailLineIndicator
        isTaxInLine="yes"
        isShippingInLine="yes"
        isAccountingInLine="yes"/>
      <InvoicePartner>
        <Contact role="soldTo" addressID="B2.4.319">
          <Name xml:lang="en">Mike Smith</Name>
          Postal address, email address, phone, and fax information
        </Contact>
      </InvoicePartner>
    </InvoiceDetailRequest>
  </Request>
</cXML>

```

```

<InvoicePartner>
  <Contact role="remitTo" addressID="Billing">
    <Name xml:lang="en">Lisa King</Name>
    Postal address, email address, phone, and fax information
  </Contact>
  <IdReference identifier="00000-11111" domain="accountReceivableID">
    <Creator xml:lang="en">Supplier ERP</Creator>
  </IdReference>
  <IdReference identifier="123456789" domain="bankRoutingID">
    <Creator xml:lang="en">Supplier Bank</Creator>
  </IdReference>
</InvoicePartner>
<PaymentTerm payInNumberOfDays="10">
  <Discount>10</Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays="20">
  <Discount>5</Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays="30">
  <Discount>0</Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays="40">
  <Discount>-5</Discount>
</PaymentTerm>
</InvoiceDetailRequestHeader>
<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <MasterAgreementIDInfo agreementID="MA-1234"/>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem invoiceLineNumber="1" quantity="100">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice>
      <Money currency="USD">57.13</Money>
    </UnitPrice>
    <InvoiceDetailItemReference lineNumber="2">
      <ItemID>
        <SupplierPartID>TOW08134</SupplierPartID>
      </ItemID>
      <Description xml:lang="en">Roll Towel Series 2000</Description>
    </InvoiceDetailItemReference>
    <SubtotalAmount>
      <Money currency="USD">5713</Money>
    </SubtotalAmount>
    <Tax>
      <Money currency="USD">287</Money>
      <Description xml:lang="en">total item tax</Description>
      <TaxDetail purpose="tax" category="State sales tax" percentageRate="8">
        <TaxableAmount>
          <Money currency="USD">5713</Money>
        </TaxableAmount>
      </TaxDetail>
    </Tax>
  </InvoiceDetailItem>
</InvoiceDetailOrder>

```

```

        <TaxAmount>
          <Money currency="USD">200</Money>
        </TaxAmount>
        <TaxLocation xml:lang="en">CA</TaxLocation>
      </TaxDetail>
    </Tax>
    <GrossAmount>
      <Money currency="USD">6000</Money>
    </GrossAmount>
    <NetAmount>
      <Money currency="USD">6000</Money>
    </NetAmount>
  </InvoiceDetailItem>
  <InvoiceDetailServiceItem
    invoiceLineNumber="2"
    quantity="100">
    <InvoiceDetailServiceItemReference lineNumber="1">
      <Classification domain="UNSPC">76111501</Classification>
      <Description xml:lang="en">
        Window cleaning services at $30/hour
      </Description>
    </InvoiceDetailServiceItemReference>
    <SubtotalAmount>
      <Money currency="USD">3000.00</Money>
    </SubtotalAmount>
    <Period startDate="2001-02-01T12:00:00-00:00"
      endDate="2001-03-30T12:00:00-00:00"/>
    <UnitOfMeasure>HUR</UnitOfMeasure>
    <UnitPrice>
      <Money currency="USD">30</Money>
    </UnitPrice>
    <Distribution>
      <Accounting name="Buyer assigned accounting code 1">
        <AccountingSegment id="ABC123456789">
          <Name xml:lang="en">Facilities</Name>
          <Description xml:lang="en">Facilities</Description>
        </AccountingSegment>
      </Accounting>
      <Charge>
        <Money currency="USD">3000</Money>
      </Charge>
    </Distribution>
    <Extrinsic name="serviceLocation">
      <Contact role="serviceLocation">
        <Name xml:lang="en">Jerry Seinfeld : NEW YORK</Name>
        <PostalAddress>
          <Street>2345 S. SAN PEDRO</Street>
          <City>New York</City>
          <State>NY</State>
          <PostalCode>10002</PostalCode>

```

```

        <Country isoCountryCode="US">USA</Country>
      </PostalAddress>
    </Contact>
  </Extrinsic>
</InvoiceDetailServiceItem>
  <!-- timecard invoice service line item -->
<InvoiceDetailServiceItem
  invoiceLineNumber = "3" quantity = "12">
  <InvoiceDetailServiceItemReference lineNumber = "1">
    <Classification domain = "UNSPC">80111604</Classification>
    <Description xml:lang = "en">Assistant AA101</Description>
  </InvoiceDetailServiceItemReference>
  <SubtotalAmount>
    <Money currency = "USD">1200</Money>
  </SubtotalAmount>
  <Period startDate = "2001-04-01T12:00:00-00:00"
    endDate = "2001-04-30T12:00:00-00:00"/>
  <UnitRate>
    <Money currency = "USD">100.00</Money>
    <UnitOfMeasure>HUR</UnitOfMeasure>
    <TermReference termName="payCode" term="regular"/>
  </UnitRate>
  <GrossAmount>
    <Money currency = "USD">1200</Money>
  </GrossAmount>
  <NetAmount>
    <Money currency = "USD">1200</Money>
  </NetAmount>
  <InvoiceLaborDetail>
    <Contractor>
      <ContractorIdentifier domain="ContractorId">
        Contr1234
      </ContractorIdentifier>
      <Contact>
        <Name>John Doe</Name>
      </Contact>
    </Contractor>
    <JobDescription>
      Assistant left-handed broom closet monitor.
    </JobDescription>
    <Supervisor>
      <Contact>
        <Name>Jill Hill</Name>
      </Contact>
    </Supervisor>
  </InvoiceTimeCardDetail>
  <TimeCardIDInfo timeCardID="TC123">
  </InvoiceTimeCardDetail>
</InvoiceLaborDetail>
</InvoiceDetailServiceItem>

```

```

</InvoiceDetailOrder>
<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <MasterAgreementIDInfo agreementID="MA-1235"/>
  </InvoiceDetailOrderInfo>
  <!-- milestone invoicing -->
  <InvoiceDetailServiceItem invoiceLineNumber="4">
    <InvoiceDetailServiceItemReference lineNumber="1">
      <Classification domain="UNSPC">78102694</Classification>
      <Description xml:lang="en">
        Market Research preliminary analysis
      </Description>
    </InvoiceDetailServiceItemReference>
    <SubtotalAmount>
      <Money currency="USD">5000</Money>
    </SubtotalAmount>
  </InvoiceDetailServiceItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>
  <SubtotalAmount>
    <Money currency="USD">13713</Money>
  </SubtotalAmount>
  <Tax>
    <Money currency="USD">287</Money>
    <Description xml:lang="en">total tax</Description>
    <TaxDetail purpose="tax"
      category="State sales tax"
      percentageRate="8">
      <TaxableAmount>
        <Money currency="USD">5713</Money>
      </TaxableAmount>
      <TaxAmount>
        <Money currency="USD">200</Money>
      </TaxAmount>
      <TaxLocation xml:lang="en">CA</TaxLocation>
    </TaxDetail>
    <TaxDetail purpose="tax"
      category="Federal sales tax"
      percentageRate="2">
      <TaxableAmount>
        <Money currency="USD">5713</Money>
      </TaxableAmount>
      <TaxAmount>
        <Money currency="USD">87</Money>
      </TaxAmount>
    </TaxDetail>
  </Tax>
  <GrossAmount>
    <Money currency="USD">14000.00</Money>
  </GrossAmount>

```

```

        <NetAmount>
          <Money currency="USD">14000.00</Money>
        </NetAmount>
        <DueAmount>
          <Money currency="USD">14000.00</Money>
        </DueAmount>
      </InvoiceDetailSummary>
    </InvoiceDetailRequest>
  </Request>
</cXML>

```

Marketplace Invoice

This example shows the header of an invoice sent to a marketplace. It illustrates how to generate correct credentials for a marketplace.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/
InvoiceDetail.dtd">
<cXML payloadID="123344-2001@.supplierorg.com"
timestamp="2001-04-20T23:59:45-07:00">
  <Header>
    <From>
      <!-- Supplier -->
      <Credential domain="AribaNetworkUserId">
        <Identity>chef@supplierorg.com</Identity>
      </Credential>
    </From>
    <To>
      <!-- Marketplace -->
      <Credential domain="AribaNetworkUserId" type="marketplace">
        <Identity>bigadmin@marketplace.org</Identity>
      </Credential>
      <!-- Marketplace Member Organization -->
      <Credential domain="AribaNetworkUserId">
        <Identity>admin@acme.com</Identity>
      </Credential>
    </To>
    <Sender>
      <!-- Supplier -->
      <Credential domain="AribaNetworkUserId">
        <Identity>chef@supplierorg.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Nifty Invoice Generator V1.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <InvoiceDetailRequest>

```

Chapter 12

Catalogs

Catalogs are files that convey product and service content to buying organizations. Suppliers use them to describe the products and services they offer and their prices, and they are the main communication channel from suppliers to their customers.

This chapter describes:

- [Catalog Definitions](#)
- [Type Definitions](#)
- [Subscription Management Definitions](#)
- [Catalog Upload Transaction](#)

Catalog Definitions

The cXML catalog definitions consist of two main elements: Supplier and Index. All three elements describe data intended for persistent or cached use within a hub or a buying organization's procurement system.

- **Supplier**—Contains basic data about the supplier, such as address, contact, and ordering information.
- **Index**—Describes data about the supplier's inventory of goods and services, such as description, part numbers, and classification codes.

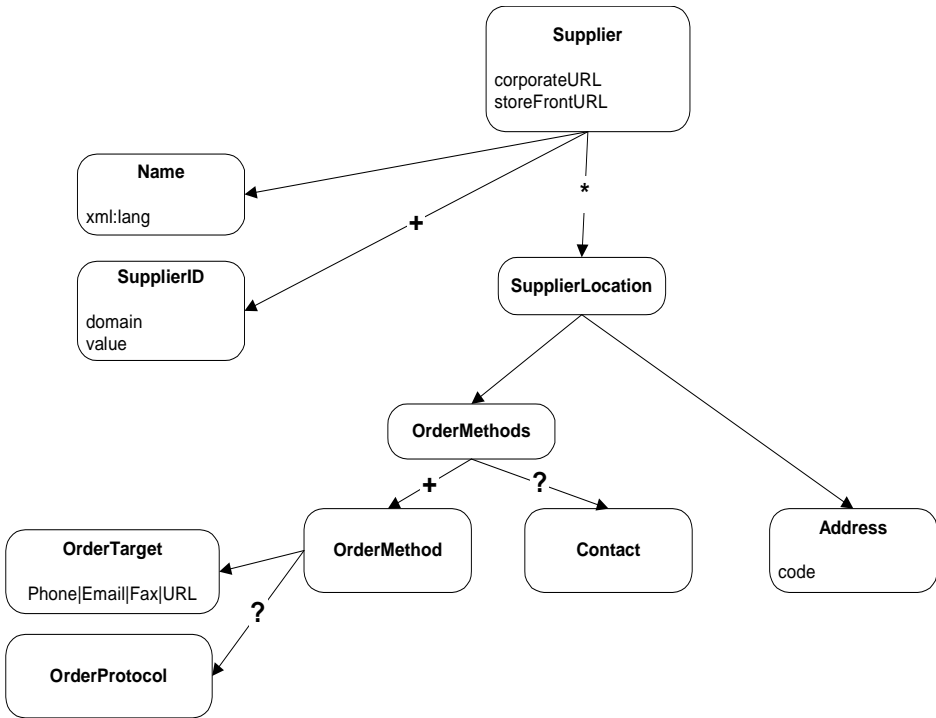
Contract was a catalog element that was deprecated in cXML 1.2.008.

Note that Index uses several sub-elements to describe line items in suppliers' inventories. Suppliers can send either price information for caching within buyers' systems or PunchOut information to enable buyers to punch out to remote websites for pricing and other information.

These elements are unusual in cXML because they commonly appear as the top level element in a compliant XML document. In fact, Index rarely appears elsewhere in a cXML document.

Supplier

The Supplier element encapsulates a named supplier of goods or services. It must have a Name element and a SupplierID element. Additionally, it describes optional address and ordering information for the supplier:



Supplier has the following attributes:

corporateURL (optional)	URL for supplier's website.
storeFrontURL (optional)	URL for website for shopping or browsing.

The following example shows an outline of the Supplier element:

```

<Supplier>
  <Name xml:lang="en-US">Workchairs </Name>
  <SupplierID domain="InternalSupplierID">29</SupplierID>
  <SupplierID domain="DUNS">76554545</SupplierID>
  <SupplierLocation>
    <Address>
      <Name xml:lang="en-US">Main Office</Name>
      <PostalAddress>
        ...
      </PostalAddress>
      <Email>bobw@workchairs.com</Email>
      <Phone name="Office">
        ...
      </Phone>
      <Fax name="Order">
        ...
      </Fax>
      <URL>http://www.workchairs.com/Support.htm</URL>
    </Address>
    <OrderMethods>
      <OrderMethod>
        <OrderTarget>
          <URL>http://www.workchairs.com/cxmlorders</URL>
        </OrderTarget>
      </OrderMethod>
    <Contact>
      <Name xml:lang="en-US">Mr. Smart E. Pants</Name>
      <Email>sepants@workchairs.com</Email>
      <Phone name="Office">
        ...
      </Phone>
    </Contact>
  </OrderMethods>
</SupplierLocation>
</Supplier>

```

SupplierLocation

Some suppliers conduct business from more than one location. A `SupplierLocation` element can be used for each location. This element also encapsulates how that location does business or the ways that it can accept orders. A `SupplierLocation` element contains an `Address` and a set of `OrderMethods`.

OrderMethods and OrderMethod

The OrderMethods element is a grouping of one or more OrderMethod elements for the given SupplierLocation element. The position of OrderMethods in the list is significant—the first element is the preferred ordering method, the second element is the next priority, and so on in decreasing order of preference.

OrderMethod encapsulates ordering information in the form of an order target (such as phone, fax, or URL) and an optional protocol to further clarify the ordering expectations at the given target; for example, “cxml” for a URL target.

Index

This element is the root element for updating catalogs within buying organizations’ procurement systems.

An Index element is associated with a single supplier. The Index element allows for a list of supplier IDs, where each ID is considered a synonym for that supplier.

The Index contains one or more IndexItem elements. The IndexItem element contains elements that add or delete from the buying organization’s cached catalog. The following example shows an outline of an Index element:

```
<Index loadmode="Incremental">
  <SupplierID> ... </SupplierID>
  ...
  <IndexItem>
    <IndexItemAdd>
      <ItemID>
        ...
      </ItemID>
      <ItemDetail>
        ...
      </ItemDetail>
      <IndexItemDetail>
        <SearchGroupData>
          ...
        </SearchGroupData>
        ...
      </IndexItemDetail>
    </IndexItemAdd>
  </IndexItem>

  <IndexItem>
    <IndexItemDelete>
      <ItemID>
        ...
      </ItemID>
    </IndexItemDelete>
  </IndexItem>
</Index>
```

```

        </ItemID>
      </IndexItemDelete>
    </IndexItem>
  <IndexItem>
    <IndexItemPunchout>
      <ItemID>
        ...
      </ItemID>
      <PunchOutDetail>
        <SearchGroupData>
          ...
        </SearchGroupData>
        ...
      </PunchOutDetail>
    </IndexItemPunchout>
  </IndexItem>
</Index>
```

Index has the following attribute:

loadmode (optional)	The mode in which the target application should load the Index:	
	"Full"	Completely replaces a previously loaded index.
	"Incremental"	Imports the index on top of the existing index, replacing or deleting existing items and adding new items. The recommended application default is incremental.

IndexItem, IndexItemAdd, IndexItemDelete, and IndexItemPunchout

The IndexItem element is a container for the list of items in an index. It contains three types of elements:

- IndexItemAdd—Inserts a new item or updates an existing item in the index. It contains an ItemID element, an ItemDetail element, and an IndexItemDetail element.
- IndexItemDelete—Removes an item from the index. It contains an ItemID element identifying the item.
- IndexItemPunchout—Inserts an item for initiating puchout to the supplier's website. It contains a PunchoutDetail element and an ItemID element. It is similar to an IndexItemAdd element except that it does not require price information. Buyers acquire item details in real-time from the supplier's website.

ItemID

The ItemID element uniquely identifies a supplier's items. It contains a SupplierPartID element and an optional SupplierPartAuxiliaryID element.

If SupplierPartID does not uniquely identify the item, the supplier should use SupplierPartAuxiliaryID to specify an "auxiliary" key that identifies the part uniquely when combined with the SupplierID and SupplierPartID. For example, a supplier might use the same SupplierPartID for an item, but have a different price for units of "EA" and "BOX". In this case, a reasonable SupplierPartAuxiliaryID for the two items might be "EA" and "BOX."

SupplierPartAuxiliaryID could also be used as a supplier cookie, enabling the supplier to refer to complex configuration or part data. It could contain all the data necessary for the supplier to reconstruct what the item in question is in their computer system (a basket or cookie of data that makes sense only to the supplier). For more information, see ["Buyer and Supplier Cookies"](#) on page 96.

ItemDetail

ItemDetail contains detailed information about an item, or all the data that a user might want to see about an item beyond the essentials represented in the ItemID. It must contain a UnitPrice, a UnitOfMeasure, one or more Description elements, and a Classification, and it can optionally contain a ManufacturerPartID, a ManufacturerName, a URL, a LeadTime, and any number of Extrinsic elements. For more information, see ["ItemDetail"](#) on page 108.

The optional LeadTime element describes the number of days needed for the buyer to receive the product. For example:

```
<LeadTime>14</LeadTime>
```

Note that in an IndexItemAdd element, duplicate LeadTime information might come from both ItemDetail, where it is optional, and IndexItemDetail, where it is mandatory. If the LeadTime elements are defined in both cases, then they should be identical.

In the context of an IndexItemAdd, Extrinsic elements extend information about a particular item. These extensions should not be transmitted to a supplier within an OrderRequest, because the supplier can retrieve the same data using the unique ItemID.

IndexItemDetail

The IndexItemDetail element contains index-specific elements that define additional aspects of an item, such as LeadTime, ExpirationDate, EffectiveDate, SearchGroupData, or TerritoryAvailable.

PunchoutDetail

PunchoutDetail is similar to ItemDetail, except it requires only one or more Description elements and a Classification. It can also contain URL, ManufacturerName, ManufacturerPartID, ExpirationDate, EffectiveDate, SearchGroupData, TerritoryAvailable, and Extrinsic elements. It does not contain pricing, lead time, or unit of measure information.

Type Definitions

Types allow type providers such as content aggregators, suppliers, and marketplaces to extend root catalog item definitions and to define named groupings of commodity-specific attributes such as parametric types.

Types are named collections of named attributes. Each attribute is further defined in terms of a type, that is, types can contain other types. Types can also derive from or extend other types.

Type definitions describe supplemental catalog attributes and parametric data types. They provide a rich framework for defining parametric types, and they allow the definition and standardization of parametric types from *type provider* organizations independent of index data.

Use the SearchGroupData and SearchDataElement elements to specify the actual parametric data for a given catalog item. SearchGroupData must reference a defined type, and SearchDataElement specifies data for each type attribute within that type.

A TypeDefinition document contains a TypeProvider element and either Type or PrimitiveType elements.

TypeProvider

TypeProvider specifies the provider of the types being defined, identified by a name and one or more IDs (for example, NetworkId or DUNS).

TypeProvider has the following attribute:

name (required)	The canonical name used to reference the type provider when fully qualifying the name of a type (for example, in a SearchGroupData element reference).
---------------------------	--

Name

The Name element is for localized display purposes, allowing different names to be provided per locale.

OrganizationID

Unique identifier for the type provider organization.

Type

Type elements are named elements containing one or more TypeAttribute elements. Types can extend (or derive from) other types, thus inheriting their parents' TypeAttribute elements.

There is one important distinction between type inheritance and standard object-oriented inheritance models: child TypeAttributes cannot override parent TypeAttributes.

It is illegal to define a TypeAttribute of the same name as a parent TypeAttribute.

Type has the following attributes:

name (required)	Canonical name of the type.
extends (optional)	Name of the type that is being extended.

Name

Type names are always scoped by TypeProvider names, allowing for the existence of multiple type taxonomies. Applications should respect the following notation for a fully-qualified type name outside a defined TypeProvider scope:

Type Provider Name: Type Name

For example, if an organization named Acme provides a type definition named Pipes, that type would be referenced as "Acme:Pipes" in SearchGroupData names.

Description

You can provide names in multiple locales through the optional `Description` element list. The `ShortName` element within that `Description` should be used to provide an alternative locale specific name for the type. The required name attribute should be used within the `SearchGroupData` element to reference a given type.

TypeAttribute

`TypeAttribute` elements define attributes within a type. The name attribute is required and is the name used in the `SearchDataElement` element. Optional `Name` elements provide locale-specific alternative names for this attribute.

`TypeAttribute` elements themselves are of a named type, as indicated by the "type" attribute. The name can be another `Type`, or a `PrimitiveType`, defined below.

name (required)	Specifies the canonical name of this attribute.
type (required)	Specifies the data type of this attribute. Can be one of: "integer" A whole number, with no fraction. "string" A group of characters with words that can be individually indexed for free text searching. "literal" A group of characters with words that cannot be individually indexed for free text searching. "double" A floating point number. "date" A date of the form yyyy-mm-dd; for example, 2002-01-25 "boolean" A Boolean value; yes, no, 1, 0, true, false, t, or f.
shortTag	Alias for this attribute.
mappedFrom	Specifies the name of another object in the system that implicitly defines this attribute.
isRequired	Indicates whether this attribute requires a (non-empty) value.
isRequiredForOrdering	Indicates that the value for an attribute must be provided (usually by the requisitioner) before the item can be included in an order for the supplier. Typically used for ad-hoc or partially specified catalog items.
isRefinable	Indicates whether this attribute is refinable in search queries.
isSearchable	Indicates whether this attribute is searchable in search queries.
isCollection	Indicates whether this attribute allows repeating values.
isCaseSensitive	Indicates whether this attribute preserves letter case. This property applies only to attributes of type string or literal. It has no effect on numeric, boolean, or date attributes, nor does it apply to attributes of complex type.

isInKey	Indicates whether this attribute is part of the unique key for the type.
isInFreeTextSearch	Indicates whether this attribute should be indexed to be a candidate in a free-text (All) query.
isHidden	Indicates whether this attribute is displayed to users.
isSortable	Indicates whether this attribute can be sorted.
isReadOnly	Indicates whether values assigned to this attribute are frozen and cannot be changed by the receiving application.
unit	Specifies the unit of this attribute, if applicable. For example, if the TypeAttribute is of a PrimitiveType with a scalar type of "integer", this unit might be "IN" to indicate inches.

Name

Localized name of the TypeAttribute.

Description

Localized description of the TypeAttribute.

EnumerationValue

EnumerationValue allows you to optionally specify a set of one or more valid data values for the TypeAttribute.

For example:

```
<TypeAttribute name="COLOR"
  type="Name"
  isRefinable="yes">
  <Name xml:lang="en">Color</Name>
  <EnumerationValue>Red</EnumerationValue>
  <EnumerationValue>Yellow</EnumerationValue>
  <EnumerationValue>Black</EnumerationValue>
</TypeAttribute>
```

Range

Range allows you to optionally specify a range of valid data values for the TypeAttribute. It contains RangeBegin, RangeEnd, or both.

For example:

```
<TypeAttribute name="WEIGHT"
  type="Number"
```

```
isRefinable="yes">
<Name xml:lang="en">Weight</Name>
<Range>
  <RangeBegin>12</RangeBegin>
  <RangeEnd inclusive="no">100</RangeEnd>
</Range>
</TypeAttribute>
```

Both RangeBegin and RangeEnd can optionally specify the attribute inclusive="no", which excludes the specified beginning or ending value as legal values.

PrimitiveType

PrimitiveType is a named scalar type, where the list of recognized scalar types is given above. These types are building blocks for defining simple TypeAttributes. For example a PrimitiveType could define a TypeAttribute that is a string of length 255.

PrimitiveType has the following optional attributes:

min	The minimum length for a TypeAttribute of scalarType "string" or "literal".
max	The maximum length for a TypeAttribute of scalarType "string" or "literal".
maxPrecision	The maximum precision for a TypeAttribute of scalarType "double".
maxScale	The maximum scale for a TypeAttribute of scalarType "double".

Subscription Management Definitions

Intermediaries such as network commerce hubs can manage supplier information and catalogs used by procurement systems.

This section describes request-response elements for managing supplier data and catalogs. In all cases, the requests are initiated by the procurement system.

This section discusses:

- [Supplier Data](#)
- [Catalog Subscriptions](#)

Supplier Data

Supplier data management uses three types of transactions:

- **SupplierList** – Returns the names of suppliers with which the buyer has relationships.
- **SupplierData** – Returns supplier details.
- **SupplierChange** – Returns the names of suppliers whose information has changed.

SupplierListRequest

SupplierListRequest requests a list of the suppliers with whom the buyer has established trading relationships.

```
<Request>
  <SupplierListRequest/>
</Request>
```

SupplierListResponse

SupplierListResponse lists the suppliers with whom the buyer has established trading relationships.

```
<Response>
  <Status code="200" text="OK"/>
  <SupplierListResponse>
    <Supplier corporateURL=http://www.workchairs.com
      storeFrontURL="http://www.workchairs.com">
      <Name xml:lang="en-US">Workchairs, Inc.</Name>
      <Comments xml:lang="en-US">this is a cool company</Comments>
      <SupplierID domain="DUNS">123456</SupplierID>
    </Supplier>
    <Supplier corporateURL=http://www.computersRus.com
      storeFrontURL="http://www.computersRus.com">
      <Name xml:lang="en-US">Computers R us</Name>
      <Comments xml:lang="en-US">another cool company</Comments>
      <SupplierID domain="DUNS">123456789</SupplierID>
    </Supplier>
  </SupplierListResponse>
</Response>
```

SupplierDataRequest

SupplierDataRequest requests data about a supplier.

```
<Request>
```

```

    <SupplierDataRequest>
      <SupplierID domain="DUNS">123456789</SupplierID>
    </SupplierDataRequest>
  </Request>

```

SupplierDataResponse

SupplierDataResponse contains data about a supplier.

```

<Response>
  <Status code="200" text="OK"/>
  <SupplierDataResponse>
    <Supplier corporateURL=http://www.workchairs.com
      storeFrontURL="http://www.workchairs.com">
      <Name xml:lang="en-US">Workchairs, Inc.</Name>
      <Comments xml:lang="en-US">this is a cool company</Comments>
      <SupplierID domain="DUNS">123456</SupplierID>
      <SupplierLocation>
        <Address>
          <Name xml:lang="en-US">Main Office</Name>
          <PostalAddress>
            <DeliverTo>Bob A. Worker</DeliverTo>
            <Street>123 Front Street</Street>
            <City>Toosunny</City>
            <State>CA</State>
            <PostalCode>95000</PostalCode>
            <Country isoCountryCode="US">USA</Country>
          </PostalAddress>
          <Email>bobw@workchairs.com</Email>
          <Phone name="Office">
            <TelephoneNumber>
              <CountryCode
                isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>800</AreaOrCityCode>
              <Number>5551212</Number>
            </TelephoneNumber>
          </Phone>
          <Fax name="Order">
            <TelephoneNumber>
              <CountryCode
                isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>408</AreaOrCityCode>
              <Number>5551234</Number>
            </TelephoneNumber>
          </Fax>
          <URL>http://www.workchairs.com/Support.htm</URL>
        </Address>
        <OrderMethods>
          <OrderMethod>

```

```

        <OrderTarget>
          <URL>http://www.workchairs.com/cxmlorder</URL>
        </OrderTarget>
        <OrderProtocol>cXML</OrderProtocol>
      </OrderMethod>
    </OrderMethods>
  </SupplierLocation>
</Supplier>
</SupplierDataResponse>
</Response>

```

For information about the `Supplier` element, see [“Supplier”](#) on page 272.

SupplierChangeMessage

This element is for notification of changes to supplier data.

This message relies on the `GetPending` transaction. The buying organization sends a `GetPendingRequest` to query for waiting messages. If the network commerce hub has a message waiting, it includes it within the `GetPendingResponse`. For more information, see Chapter 13, [“GetPending Transaction.”](#)

```

<Message>
  <SupplierChangeMessage type="new">
    <Supplier corporateURL=http://www.workchairs.com
      storeFrontURL="http://www.workchairs.com">
      <Name xml:lang="en-US">Workchairs, Inc.</Name>
      <Comments xml:lang="en-US">this is a cool company</Comments>
      <SupplierID domain="DUNS">123456</SupplierID>
      <SupplierLocation>
        <Address>
          <Name xml:lang="en-US">Main Office</Name>
          <PostalAddress>
            <DeliverTo>Bob A. Worker</DeliverTo>
            <Street>123 Front Street</Street>
            <City>Toosunny</City>
            <State>CA</State>
            <PostalCode>95000</PostalCode>
            <Country isoCountryCode="US">USA</Country>
          </PostalAddress>
          <Email>bobw@workchairs.com</Email>
          <Phone name="Office">
            <TelephoneNumber>
              <CountryCode
                isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>800</AreaOrCityCode>
              <Number>5551212</Number>
            </TelephoneNumber>
          </Phone>
        </Address>
      </SupplierLocation>
    </Supplier>
  </SupplierChangeMessage>
</Message>

```

```

    <Fax name="Order">
      <TelephoneNumber>
        <CountryCode
          isoCountryCode="US">1</CountryCode>
        <AreaOrCityCode>408</AreaOrCityCode>
        <Number>5551234</Number>
      </TelephoneNumber>
    </Fax>
    <URL>http://www.workchairs.com/Support.htm</URL>
  </Address>
  <OrderMethods>
    <OrderMethod>
      <OrderTarget>
        <URL>http://www.workchairs.com/cxmlorder</URL>
      </OrderTarget>
      <OrderProtocol>cXML</OrderProtocol>
    </OrderMethod>
  </OrderMethods>
</SupplierLocation>
</Supplier>
</SupplierChangeMessage>
</Message>

```

Catalog Subscriptions

Catalog-subscription management uses three types of transactions:

- SubscriptionList – Returns the names of catalogs to which the buyer has subscribed.
- SubscriptionContent – Returns catalog contents.
- SubscriptionChange – Returns the names of catalogs that have changed.

Subscription

The catalog subscription transactions all use the Subscription element to describe metadata about a catalog subscription.

For example:

```

<Subscription>
  <InternalID>1234</InternalID>
  <Name xml:lang="en-US">Q2 Prices</Name>
  <Changetime>2002-03-12T18:39:09-08:00</Changetime>
  <SupplierID domain="DUNS">123456789</SupplierID>
  <Format version="2.1">CIF</Format>
  <Description xml:lang="en-US">The best prices for software</Description>

```

</Subscription>

The elements within Subscription include:

InternalID	A unique ID internal to the intermediary. Contains an optional domain attribute.
Name	The name of the subscription.
ChangeTime	The date and time when any aspect of the subscription last changed.
SupplierID	The ID of the supplier.
Format	The format of the catalog.
Description	A description of the catalog.

SubscriptionListRequest

This element requests the buyer's current list of catalog subscriptions.

```
<Request>
  <SubscriptionListRequest/>
</Request>
```

SubscriptionListResponse

This element lists the buyer's current list of catalog subscriptions.

```
<Response>
  <Status code="200" text="OK"/>
  <SubscriptionListResponse>
    <Subscription>
      <InternalID>1234</InternalID>
      <Name xml:lang="en-US">Q2 Software Prices</Name>
      <Changetime>1999-03-12T18:39:09-08:00</Changetime>
      <SupplierID domain="DUNS">123456789</SupplierID>
      <Format version="2.1">CIF</Format>
      <Description xml:lang="en-US">The best prices for software</Description>
    </Subscription>
    <Subscription>
      <InternalID>1235</InternalID>
      <Name xml:lang="en-US">Q2 Hardware Prices</Name>
      <Changetime>1999-03-12T18:15:00-08:00</Changetime>
      <SupplierID domain="DUNS">555555555</SupplierID>
      <Format version="2.1">CIF</Format>
      <Description xml:lang="en-US">The best prices for hardware</Description>
    </Subscription>
  </SubscriptionListResponse>
</Response>
```



```

        </Subscription>
    </SubscriptionListResponse>
</Response>

```

SubscriptionContentRequest

This element requests the contents of a subscribed catalog. The request includes the InternalID and SupplierID for the catalog.

```

<Request>
  <SubscriptionContentRequest>
    <InternalID>1234</InternalID>
    <SupplierID domain="DUNS">123456789</SupplierID>
  </SubscriptionContentRequest>
</Request>

```

SubscriptionContentResponse

This element contains the contents of a catalog. The catalog format can be either CIF (Catalog Interchange Format) or cXML. If it is CIF, it is base64 encoded and included as the content of a CIFContent element. If it is cXML, the Index element is directly included.

```

<Response>
  <Status code="200" text="OK"/>
  <SubscriptionContentResponse>
    <Subscription>
      <InternalID>1234</InternalID>
      <Name xml:lang="en-US">Q2 Software Prices</Name>
      <Changetime>1999-03-12T18:39:09-08:00</Changetime>
      <SupplierID domain="DUNS">123456789</SupplierID>
      <Format version="3.0">CIF</Format>
      <Description xml:lang="en-US">The best prices for software</Description>
    </Subscription>
    <SubscriptionContent filename="april_prices.cif">
      <CIFContent>
        <!-- base64 encoded data -->
        ABCDBBDBDBDBDB
        . . .
      </CIFContent>
    </SubscriptionContent>
  </SubscriptionContentResponse>
</Response>

```

SubscriptionChangeMessage

This element signals the buying organization's procurement system that a subscribed catalog has changed.

This message relies on the GetPending transaction. The buying organization sends a GetPendingRequest to query for waiting messages. If the network commerce hub has a message waiting, it includes it within the GetPendingResponse. For more information, see Chapter 13, "[GetPending Transaction](#)."

```
<Message>
  <SubscriptionChangeMessage type="new">
    <Subscription>
      <InternalID>1234</InternalID>
      <Name xml:lang="en-US">Q2 Software Prices</Name>
      <Changetime>1999-03-12T18:39:09-08:00</Changetime>
      <SupplierID domain="DUNS">123456789</SupplierID>
      <Format version="2.1">CIF</Format>
    </Subscription>
  </SubscriptionChangeMessage>
</Message>
```

The type attribute describes the type of change: new, delete, or update.

Catalog Upload Transaction

The cXML Catalog Upload transaction enables suppliers to programmatically upload and publish catalogs on network commerce hubs.

The Catalog Upload transaction gives you an alternative to logging on to network hubs to interactively upload and publish catalogs. You can use it to automatically distribute updated catalogs whenever you change pricing or availability of your products or services.

The Catalog Upload transaction supports both CIF and cXML catalogs.

The Catalog Upload transaction consists of two cXML documents:

CatalogUploadRequest

Sent by suppliers to upload a catalog. It contains the catalog as an attachment and specifies whether the catalog is new or an update, and whether to automatically publish it after upload.

Response Sent by the network commerce hub to acknowledge the receipt of a CatalogUploadRequest.

CatalogUploadRequest

The following example shows a CatalogUploadRequest:

MIME header

ID of network hub

ID of MIME attachment

--kdfkajfdksadjfklasdjfkljdfdsfdkf

Content-type: text/xml; charset=UTF-8

Content-ID: <part1.PCO28.975529413484@saturn.workchairs.com>

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

<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">

<cXML timestamp="2000-12-28T16:56:03-08:00" payloadID="155556789@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>

<UserAgent>My Homemade Catalog Manager V2.0</UserAgent>

</Sender>

</Header>

<Request>

<CatalogUploadRequest operation="update">

<CatalogName xml:lang="en">Winter Prices</CatalogName>

<Description xml:lang="en">This catalog contains our premiere-level prices for office chairs and other durable furniture.</Description>

<Attachment>

<URL>cid:part2.PCO28.975529413154@saturn.workchairs.com</URL>

</Attachment>

<Commodities>

<CommodityCode>52</CommodityCode>

</Commodities>

<AutoPublish enabled="true"/>

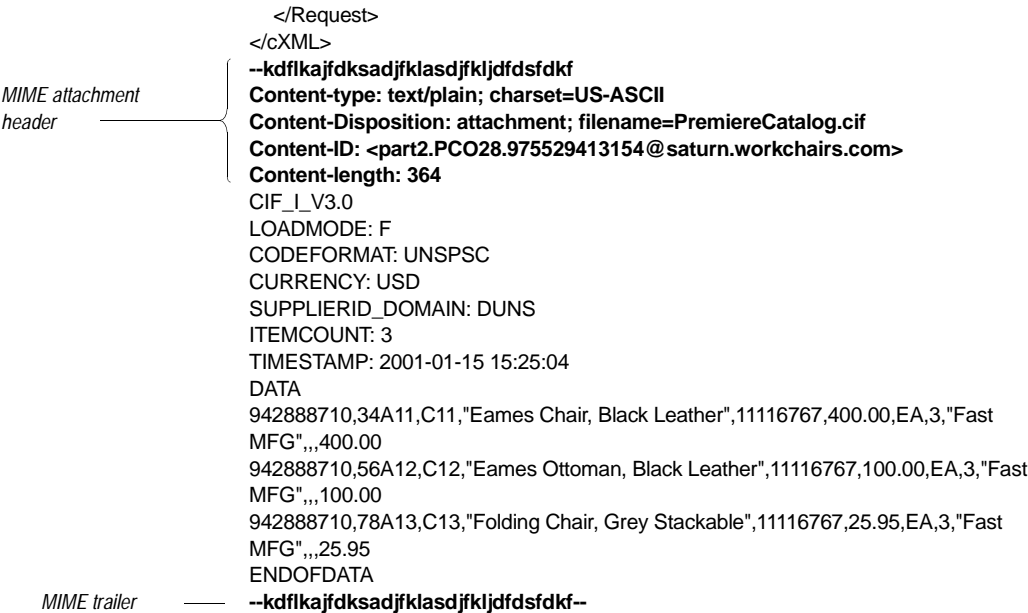
<Notification>

<Email>judy@workchairs.com</Email>

<URLPost enabled="true"/>

</Notification>

</CatalogUploadRequest>



CatalogUploadRequest Element

CatalogUploadRequest has the following attribute:

operation	Specifies the type of upload to perform:
“new”	Uploads a new catalog. A catalog with the same name must not exist.
“update”	Overwrites an exiting catalog. A catalog with the same name must exist.

CatalogUploadRequest contains the following elements.

CatalogName

CatalogName specifies the name of the uploaded catalog. This value is the user-visible name, not the file name of the catalog.

CatalogName has the following attribute:

xml:lang	<p>Specifies the language used for the catalog name.</p> <p>Language codes are defined in the XML 1.0 Specification (at www.w3.org/TR/1998/REC-xml-19980210.html). In the most common case, this includes an ISO 639 Language Code and, optionally, an ISO 3166 Country Code separated by a hyphen.</p> <p>The recommended cXML language code format is xx[-YY[-zzz]*] where xx is an ISO 639 Language code, YY is an ISO 3166 Country Code, and zzz is an IANA or private subcode for the language in question. Again, use of the Country Code is always recommended. By convention, the language code is lowercase and the country code is uppercase. This is not required for correct matching of the codes.</p>
----------	---

Description

Description briefly describes the catalog contents. Buying organizations can search and view this information.

Description has the following attribute:

xml:lang	<p>Specifies the language used for the catalog name.</p> <p>For more information, see the description of xml:lang for CatalogName, above.</p>
----------	---

Attachment

Attachment specifies the URL of the attached catalog.

The Attachment element contains one URL element with the scheme “cid:”.

For more information about attachments, see “Attaching Your Catalog” on page 293.

Commodities

Commodities specifies the top-level commodity codes for the items in your catalog. Buying organizations use these codes to search for new catalogs.

The Commodities element contains one or more CommodityCode elements.

Use two-digit UNSPSC (Universal Standard Products and Services Classification) segment codes.

For a list of UNSPSC segment codes go to the UNSPSC website at www.unspsc.org.

AutoPublish

AutoPublish automatically publishes the catalog to buyers after upload.

You can automatically publish only if both of the following requirements are met:

- 1. A previous version of the catalog exists in your account and you are performing an update operation.
- 2. The previous version is in the “published” state. It must have been published private (with a list of buyers) or public.

AutoPublish has the following attribute:

Enabled	Specifies whether to automatically publish the catalog: “true” Publishes the catalog. It must be an update to a previously published catalog. “false” Does not publish the catalog. You can log on to your account and manually publish the catalog.
----------------	--

Notification

Notification sends catalog-status notifications through e-mail or cXML POST. For examples of these messages, see “[Receiving Later Catalog Status](#)” on page 294.

Notification contains either one Email element or one URLPost element, or both elements.

Email specifies the mailbox to the newtork commerce hub e-mails status messages. You can use only one Email element, and it can contain only one e-mail address.

URLPost specifies whether the newtork commerce hub sends catalog status messages as cXML StatusUpdateRequest documents.

The URL destination of the StatusUpdateRequest is determined by your website’s response to the ProfileRequest transaction. For more information see Chapter 3, “[Profile Transaction](#).”

URLPost has the following attribute

Enabled	Specifies whether the network sends catalog-status notifications through StatusUpdateRequest:	
	"true"	Enables this feature.
	"false"	Disables this feature.

Attaching Your Catalog

Send your catalog attached to the CatalogUploadRequest document. Large catalogs must be zipped to compress them before uploading.

Using a MIME envelope

Include the catalog file in the CatalogUpdateRequest as a MIME (Multipurpose Internet Mail Extensions) attachment. cXML contains only references to external MIME parts sent within one multipart MIME envelope.

The referenced catalog file must reside within a multipart MIME envelope with the cXML document. A cXML requirement for this envelope (over the basics described in RFC 2046 "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types") is the inclusion of Content-ID headers with the attached file.

Note: The cXML specification allows attachments to reside outside of the MIME envelope, but the Catalog Upload transaction does not support that attachment method.

The Attachment element contains only a reference to the external MIME part of the attachment. Attachment contains a single URL with the scheme "cid:".

For more information about attachments in cXML, see the discussion of the "Attachment" on page 123.

Catalog files can be zipped to compress them.

Response

After you send a CatalogUploadRequest, the network commerce hub replies with a standard cXML Response document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML payloadID="980306507433-6714998277961341012@10.10.83.39"
timestamp="2001-01-23T19:21:47-08:00">
```

```

<Response>
  <Status code="201" text="Accepted">The catalog upload request is
processing</Status>
</Response>
</cXML>

```

The following table lists possible status codes:

Status	Text	Meaning
200	Success	The catalog-upload request succeeded.
201	Accepted	The catalog-upload request is processing.
461	Bad Commodity Code	The commodity code you assigned to the catalog is invalid.
462	Notification Error	No notification method (e-mail or URL) provided.
463	Bad Catalog Format	The zip file is invalid.
464	Bad Catalog	No catalog is attached, or more than one is attached.
465	Duplicate Catalog Name	The name of the catalog exists.
466	No Catalog to Update	The catalog to be updated does not exist.
467	Publish Not Allowed	You attempted to publish a catalog that was not previously published.
468	Catalog Too Large	The size of the uploaded file exceeds the 4-MB limit. Zip the catalog to compress it before uploading it.
469	Bad Catalog Extension	The file name of the catalog must have .cif, .xml, or .zip extensions.
470	Catalog Has Errors	The message is the status of the catalog. (HasErrors)
499	Document Size Error	The cXML document is too large.
561	Too Many Catalogs	You cannot upload more than a specific number of catalogs per hour.
562	Publish Disabled	Catalog publishing is temporarily unavailable due to scheduled maintenance. It will be back online by the specified date and time.
563	Catalog Validating	You attempted to update a catalog before validation finished on a previous version of the catalog.

For other possible status codes, see [“Status”](#) on page 45.

Receiving Later Catalog Status

If you include the Notification element to request later catalog-status notification, the network sends a message when the catalog reaches its final status. The possible final catalog states are:

Validated	The catalog contains no syntax errors.
BadZipFormat	The zip format is incorrect.
HasErrors	The catalog contains syntax errors, and it cannot be published.
Published	The catalog has been published (private or public).

URLPost

The following example shows a StatusUpdateRequest notification sent by a network commerce hub:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.011/cXML.dtd">

<cXML timestamp="2001-01-23T18:39:44-08:00" payloadID="980303984882--
3544419350291593786@10.10.83.39">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="DUNS">
        <Identity>123456789</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>ANValidator</UserAgent>
    </Sender>
  </Header>
  <Request>
    <StatusUpdateRequest>
      <DocumentReference payloadID="123456669131--
123456789955556789@10.10.83.39"></DocumentReference>
      <Status text="Success" code="200">
        Validated
      </Status>
    </StatusUpdateRequest>
  </Request>
</cXML>
```

The possible status codes are:

Status Code	Meaning
200 Success	The catalog-upload request succeeded.
463 Bad Catalog Format	The zip file is invalid.
470 Catalog Has Errors	The message is the status of the catalog. (HasErrors)

Chapter 13

GetPending Transaction

Some buying organizations do not have HTTP entry points for receiving cXML messages from outside their corporate firewalls. The cXML specification allows for these environments.

This section introduces definitions that allow source systems to queue messages when target systems are unable to directly accept HTTP posts. Target systems instead pull messages at their convenience by using the GetPending transaction.

Two examples of documents that depend on the GetPending transaction are `SupplierChangeMessage` and `SubscriptionChangeMessage`, which are used to notify buying organizations about changes to supplier data and catalogs.

This chapter describes:

- [GetPendingRequest](#)
- [GetPendingResponse](#)

GetPendingRequest

This element pulls a set of messages that are waiting for the requester. The `MessageType` element and the `lastReceivedTimestamp` and `maxMessages` attributes control the type and count of the fetched messages.

lastReceivedTimestamp (optional)	The timestamp of the most recent message received.
maxMessages (optional)	Maximum number of messages in a single response that the requester can handle.

Upon receiving the request, the receiver returns the oldest messages, of the specified types, with timestamps equal to or later than the specified timestamp. If there are multiple messages meeting this criterion, multiple messages can be returned, subject to the `maxMessages` attribute. The queuing system discards all pending messages of the specified message types with timestamps earlier than the specified timestamp.

```
<Request>
  <GetPendingRequest lastReceivedTimestamp="2002-03-12T18:39:09-08:00"
    maxMessages="5">
    <MessageType>SubscriptionChangedMessage</MessageType>
  </GetPendingRequest>
</Request>
```

GetPendingResponse

This element contains one or more messages waiting for the requester.

```
<Response>
  <Status code="200" text="OK"/>
  <GetPendingResponse>
    <cXML xml:lang="en-US"
      payloadID="456778@ariba.com"
      timestamp="2002-03-12T18:39:09-08:00">
      <Header>
        <From>
          <Credential domain="AribaNetworkUserId">
            <Identity>admin@ariba.com</Identity>
          </Credential>
        </From>
        <To>
          <Credential domain="AribaNetworkUserId">
            <Identity>admin@acme.com</Identity>
          </Credential>
        </To>
        <Sender>
          <Credential domain="AribaNetworkUserId">
            <Identity>admin@ariba.com</Identity>
          </Credential>
          <UserAgent>Ariba.com</UserAgent>
        </Sender>
      </Header>
      <Message>
        <SubscriptionChangeMessage type="new">
          <Subscription>
            <InternalID>1234</InternalID>
            <Name xml:lang="en-US">Q2 Prices</Name>
```

```
<Changetime>2002-03-12T18:39:09-08:00
</Changetime>
<SupplierID domain="DUNS">123456789
</SupplierID>
<Format version="2.1">CIF</Format>
</Subscription>
</SubscriptionChangeMessage>
</Message>
</cXML>
</GetPendingResponse>
</Response>
```

For information about the `SupplierChangeMessage` element, see [“SupplierChangeMessage”](#) on page 284.

For information about the `SubscriptionChangeMessage` element, see [“SubscriptionChangeMessage”](#) on page 288.

Chapter 14

Provider PunchOut Transaction

Provider PunchOut enables applications to punch out to a remote application that supplies some service to the originating application, such as credit card validation, single login, or self registration.

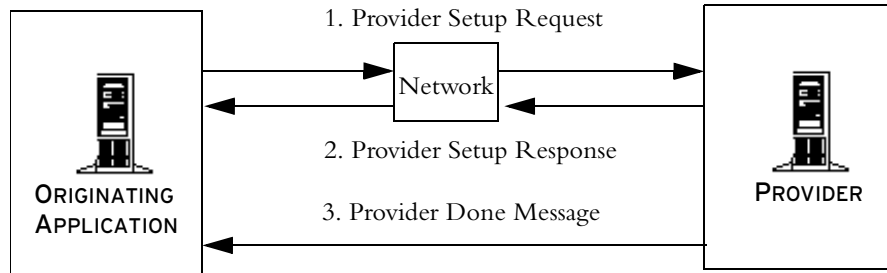
cXML messages provide a means for the originator and the provider to communicate during this transaction. These cXML documents are *ProviderSetupRequest*, *ProviderSetupResponse*, and *ProviderDoneMessage* and are tailored specifically to handle the interaction between an originating application and a service provider. They pass details such as what service is to be provided, session information, the return URL of the originator, and status or followup information.

This chapter describes:

- [Message Flow](#)
- [ProviderSetupRequest Document](#)
- [ProviderSetupResponse Document](#)
- [ProviderDoneMessage Document](#)

Message Flow

The order of cXML message flow in the Provider PunchOut transaction is shown in the following diagram.



To initiate a Provider PunchOut, the originating application sends a `ProviderSetupRequest` document to the provider. This document includes credential information for the user and the user's organization, the return URL, and the service requested from the provider. To acknowledge the request, the provider sends a `ProviderSetupResponse` document to the originating application and includes a URL for the start page indicating where the user should be redirected. When the user has finished, the provider sends a `ProviderDoneMessage` document back to the originating application, indicating that the user has completed their session at the provider's site.

ProviderSetupRequest Document

The *ProviderSetupRequest* document initiates a Provider PunchOut transaction and passes several items of information to the provider, including information about the member organization and user, the return URL, and which service is being requested.

The document contains two sections, one specified by a `Header` element, the other by a `Request` element. The `Header` contains credential information about the user and the requesting organization and the `Request` contains the actual `ProviderSetupRequest` element that contains information needed to initiate the Provider PunchOut.

Header

The *Header* portion of the document contains addressing and authentication information. The following sample is the header portion taken from a Provider Setup Request cXML document. The `UserAgent` element contains the digital signature of the

provider; a string that corresponds to the application and the version making the request. For example, “www.triton.com” or “Ariba Buyer 7.0 EA.” The two parties must agree on a common certificate format and authority.

```
<Header>
  <From>
    <!-- Triton bank -->
      <Credential domain="NetworkId" type="marketplace">
        <Identity>AN01000001709</Identity>
      </Credential>
      <Credential domain="triton.com">
        <Identity>9999</Identity>
      </Credential>
    </From>

    <To>
      <!-- AM-NE -->
        <Credential domain="NetworkId">
          <Identity>AN01000000003</Identity>
        </Credential>
      </To>
    <Sender>
      <!-- Triton bank -->
        <Credential domain="NetworkId">
          <Identity>AN01000001709</Identity>
          <SharedSecret>abracadabra</SharedSecret>
        </Credential>
        <UserAgent>www.triton.com</UserAgent>
      </Sender>
    </Header>
```

Because the Header element is similar for each message type, see “[Header](#)” on page 40 for specifics on how to construct this portion of the message.

Request

The *Request* portion of the document contains a *ProviderSetupRequest*, which has several items of information about the transaction from the originator, including a cookie to track the session for the originator, a return URL, what service is being requested from the provider, and other information contingent upon the type of service and the provider.

```
<Request>
  <ProviderSetupRequest>
    <OriginatorCookie>iTRk9bG49EJOGhJC</OriginatorCookie>
    <BrowserFormPost>
      <URL>https://www.triton.com/providerdone.asp</URL>
    </BrowserFormPost>
```

```

<SelectedService>AMNE.signin</SelectedService>
<Extrinsic name="Brand">Triton</Extrinsic>
<Extrinsic name="User">
  <Identity>0001</Identity>
</Extrinsic>
<Extrinsic name="QueryString">req=R532&login=gtou</Extrinsic>
</ProviderSetupRequest>
</Request>

```

The following table provides guidelines for the structure of the request section of the Provider PunchOut message.

Element	Instances	Parent Elements	Child Elements	Attributes
ProviderSetupRequest	1	Request	OriginatorCookie, BrowserFormPost, SelectedService, Extrinsic	None
OriginatorCookie	1	ProviderSetupRequest, ProviderDoneMessage	None	None
BrowserFormPost	0 or 1	ProviderSetupRequest	URL	None
URL	0 or 1	BrowserFormPost, Followup	None	None
SelectedService	1	ProviderSetupRequest	None	None
Extrinsic	Any	ProviderSetupRequest	Varies	name

The elements in the header section are:

Request

Contains a request to initiate a Provider PunchOut transaction, and in this case contains a ProviderSetupRequest element.

ProviderSetupRequest

A request from an originating application to a provider to initiate a transaction.

OriginatorCookie

OriginatorCookie is tied to the user's session on the requestor's site and is returned to the requestor later with the ProviderDoneMessage. This implements a one-time key allowing the user to return to the same session on the originating application.

BrowserFormPost URL

The originating application provides the BrowserFormPost location so that the provider can display a “Done” button, and provide information, such as a Status, at the end of the interactive session. Inclusion should lead to a ProviderDoneMessage document being sent from the provider at the end of each session. URL contains the location on the requestor’s site to return the user when they have finished at the provider site.

SelectedService

Identifies the service requested by the originating application and offered by the provider.

Extrinsic

The extrinsics for the Provider PunchOut depend upon what service the provider supplies. Please see specific documentation for your specific ProviderSetupRequest.

Note: XML content, elements, and their attributes must be defined in the cXML DTD or XML escaped.

Sample

To demonstrate a typical ProviderSetupRequest document, the following is a request from a marketplace host, called Triton bank, to AM-NE.

```
<cXML timestamp="2000-07-11T15:03:14-07:00" payloadID="963352994214--8721789825238347285@10.10.83.151">
```

```
<Header>
  <From>
    <Credential domain="NetworkId" type="marketplace">
      <Identity>AN01000001709</Identity>
    </Credential>

    <Credential domain="triton.com">
      <Identity>9999</Identity>
    </Credential>
  </From>

  <To>
    <Credential domain="NetworkId">
      <Identity>AN010000000003</Identity>
    </Credential>
  </To>
</Sender>
```

```

        <Credential domain="NetworkId">
            <Identity>AN01000001709</Identity>
            <SharedSecret>abracadabra</SharedSecret>
        </Credential>
        <UserAgent>www.triton.com</UserAgent>
    </Sender>
</Header>

<Request>
    <ProviderSetupRequest>
        <OriginatorCookie>iTRk9bG49EJOGhJC</OriginatorCookie>
        <BrowserFormPost>
            <URL>https://www.triton.com/providerdone.asp</URL>
        </BrowserFormPost>
        <SelectedService>AMNE.signin</SelectedService>
        <Extrinsic name="Brand">Triton</Extrinsic>
        <Extrinsic name="User">
            <Identity>0001</Identity>
        </Extrinsic>
        <Extrinsic name="QueryString">req=R532&login=gtou</Extrinsic>
    </ProviderSetupRequest>
</Request>
</cXML>

```

ProviderSetupResponse Document

The *ProviderSetupResponse* document notifies the originating application of the results of the request. Status and start page information is included.

```

<cXML payloadID="456789@amne.ariba.com"
  xml:lang="en-US" timestamp="2000-03-12T18:40:15-08:00">
    <Response>
        <Status code="200" text="OK"/>
        <ProviderSetupResponse>
            <StartPage>
                <URL>http://amne@ariba.com/enter?23423SDFSDF23</URL>
            </StartPage>
        </ProviderSetupResponse>
    </Response>
</cXML>

```

The following table provides guidelines for the structure of the ProviderSetupResponse document of the Provider PunchOut transaction.

Element	Instances	Parent Elements	Child Elements	Attributes
Response	1	cXML	Status, ProviderSetupResponse	None
Status	1	Response	None	code, text
ProviderSetupReponse	1	Response	StartPage	None
StartPage	1	ProviderSetupReponse	URL	None
URL	1	StartPage	None	None

Response

Contains the Status and ProviderSetupResponse elements.

Status

Provides information on the success or failure of the provider request. The content of the Status element can be any data needed by the requestor and can describe the error in more detail. Status has the following attributes:

code	The status code of the request. This follows the HTTP status code model. For example, 200 represents a successful request.
text	The text of the status message. This text aids user readability in logs, and it consists of canonical strings in English.

For a 200/OK status code, there might be no data. However, for a 500/Internal Server Error status code, it is strongly recommended that the actual XML parse error or application error be presented. This error allows better one-sided debugging and interoperability testing.

The provider should not include the ProviderSetupResponse element unless the status code is in the 200 range. See “Status” on page 45 for a list of all possible status code values.

ProviderSetupResponse

If the request was successful, the ProviderSetupResponse element is included in the response document and contains the StartPage and URL elements which indicate where the user should be redirected.

StartPage URL

This element contains a URL element that specifies the URL to pass to the browser to initiate the Provider PunchOut browsing session requested in the ProviderSetupRequest element. This URL must contain enough state information to bind to a session context on the provider website.

Sample

The following ProviderSetupResponse document is in reply to Triton Bank from a provider from the previous ProviderSetupRequest section.

```
<cXML payloadID="456789@amne.ariba.com"
  xml:lang="en-US" timestamp="2000-03-12T18:40:15-08:00">
  <Response>
    <Status code="200" text="OK"/>
    <ProviderSetupResponse>
      <StartPage>
        <URL>http://amne@ariba.com/enter?23423SDFSDF23</URL>
      </StartPage>
    </ProviderSetupResponse>
  </Response>
</cXML>
```

ProviderDoneMessage Document

The *ProviderDoneMessage* document contains any information the originating application must know about the completed operation at the provider site.

Header

The ProviderDoneMessage *Header* section is similar to the header sections in the Request and Response messages; however, because this message is sent with a Form Post, you should not include a SharedSecret in the Sender element. The UserAgent element contains the digital signature of the provider. The two parties must agree on a common certificate format and authority.

```
<Header>
  <From>
    <!-- AM-NE -->
    <Credential domain="NetworkId">
      <Identity>AN01000000003</Identity>
    </Credential>
```

```

</From>

<To>
  <!-- Triton bank -->
  <Credential domain="NetworkId">
    <Identity>AN01000001709</Identity>
  </Credential>
</To>

<Sender>
  <!-- AM-NE -->
  <Credential domain="NetworkId">
    <Identity>AN01000000003</Identity>
  </Credential>
  <UserAgent>Purchase</UserAgent>
</Sender>
</Header>

```

Because the Header element is similar for each message type, see [“Header”](#) on page 40 for the specifics on how to construct this portion of the message.

Message

The *Message* portion of the document contains the ProviderDoneMessage element, which contains any information requested by the originating application, and information to return to the user to their session at the originating application’s site.

```

<Message>
  <Status code="200" text="OK"/>
  <ProviderDoneMessage>
    <OriginatorCookie>c546794949</OriginatorCookie>
    <ReturnData name="method">
      <ReturnValue>Triton.transact</ReturnValue>
      <Name xml:lang="en-US">Triton OM transact</Name>
    </ReturnData>
  </ProviderDoneMessage>
</Message>

```

The following table details guidelines for the structure of the message section of the ProviderDoneMessage document.

Element	Instances	Parent Elements	Child Elements	Attributes
Message	1	None	Status, ProviderDoneMessage	None
Status	1	Message	None	text, code
ProviderDoneMessage	1	Message	OriginatorCookie, ReturnData, ReturnValue, Name	None
OriginatorCookie	1	ProviderDoneMessage	None	None
ReturnData	Any	ProviderDoneMessage	ReturnValue, Name	name
ReturnValue	1	ProviderSetupRequest	None	None
Name	1	BrowserFormPost, Followup	None	xml:lang

The elements in the message section are:

OriginatorCookie

The same element that was passed in the original ProviderSetupRequest document. It must be returned here to allow the requesting application to match the ProviderDoneMessage document with an earlier ProviderSetupRequest document and return the user to the correct session.

ReturnData

Contains any information the originator must know about the completed operation at the provider site. The name attribute identifies the type (domain) of the ReturnData to the requestor.

ReturnValue

A value that is used by the originating application. This value depends on what service the provider supplies.

Name

An identifier for the data returned. Provides a description for the contents of the ReturnData element.

When displaying values, keep in mind that Name and ReturnValue have similar semantics, but different uses in the originating application.

Sample

The provider sends the following ProviderDoneMessage document, which notifies the originating application, Triton Bank, that the user has finished with their session on the provider site.

```
<cXML timestamp="2000-07-11T15:13:28-07:00" payloadID="963353608827--
3642656259900210849@10.10.83.151">

  <Header>
    <From>
      <!-- AM-NE market cluster -->
      <Credential domain="NetworkId">
        <Identity>AN01000000003</Identity>
      </Credential>
    </From>

    <To>
      <!-- Triton bank -->
      <Credential domain="NetworkId">
        <Identity>AN01000001709</Identity>
      </Credential>
    </To>

    <Sender>
      <!-- AM-NE market cluster -->
      <Credential domain="NetworkId">
        <Identity>AN01000000003</Identity>
      </Credential>
      <UserAgent>Purchase</UserAgent>
    </Sender>
  </Header>

  <Message>
    <Status code="200" text="OK"/>
    <ProviderDoneMessage>
      <OriginatorCookie>c546794949</OriginatorCookie>
      <ReturnData name="method">
        <ReturnValue>Triton.transact</ReturnValue>
        <Name xml:lang="en-US">Triton OM transact</Name>
      </ReturnData>
    </ProviderDoneMessage>
  </Message>
</cXML>
```

Chapter 15

Alternative Authentication Methods

cXML supports alternatives to the shared secret authentication method for verifying the sender of cXML documents.

This chapter describes:

- [Message Authentication Code \(MAC\)](#)
- [Auth Transaction](#)

Message Authentication Code (MAC)

Message Authentication Code (MAC) authentication allows the authentication of documents sent directly from a client to a server without passing through a trusted third party (such as a network commerce hub) for authentication. These documents contain a credential with an authentication code that can be interpreted only by the trusted third party and the receiving server, not by the sender.

The format of the `Credential` element containing the MAC is described in [“Credential”](#) on page 41.

Overview of MACs

The primary purpose of MACs is to convey receivers’ shared secrets without revealing them to senders. MACs keep shared secrets secure by encoding them through a hash.

MACs are as secure as shared secrets. Senders must guard MACs as carefully as shared secrets. Compromising either piece of information could make trading partners vulnerable.

To use MAC authentication, both the trusted third party and the receiver must be able to compute MACs.

Computation Algorithm

MACs are created by an algorithm that combines data known by both the trusted third party and the receiver.

cXML specifies the use of the HMAC-SHA1 algorithm described in IETF RFC 2104, “HMAC: Keyed-Hashing for Message Authentication”.

The HMAC-SHA1 algorithm provide the security required for cXML, and it has been formally proven to be as secure as the underlying hash algorithm.

For more information about IETF RFC 2104, see www.ietf.org/rfc/rfc2104.txt.

Creation and Expiration Dates

Creation and expiration dates add additional security to MACs.

If a MAC is stolen, changing the sender’s shared secret has no effect. It is impractical to expect the sender to contact the receiver out-of-band to invalidate the MAC, because they might not have an established relationship. To address this problem, a creation date (creationDate) and an expiration date (expirationDate) are embedded in MACs. The expiration date limits the damage that can be result from a stolen MAC, because MACs eventually expire. The shorter the expiration period, the greater the security afforded. Receivers must reject MACs that are received after their expiration date.

Receivers can also reject unexpired MACs based on the amount of time that has elapsed since the creation date. For example, if a receiver receives a MAC that was created several years ago, but expires tomorrow, the receiver might not wish to accept the MAC. This decision is left with the implementors of the receiving systems.

It is mandatory for receivers to check that the creation date is in the past and the expiration date is in the future, and to reject it if either is not the case. However, it is optional for receivers to check whether the creation date is too long in the past.

Receivers must not only check that MACs are valid, but also that the data authenticated by MACs is acceptable. Specifically, receivers must validate that they wish to accept messages from the entities identified by the From and Sender credentials.

Computation Process

This section describes how to compute a MAC of type="FromSenderCredentials". The inputs for this MAC type are known only by the trusted third party and the receiver.

The trusted third party uses this computation to generate ProfileResponse Option elements and the receiving server uses it to validate the CredentialMac element.

Assembling the Hash Inputs

The MAC function takes two inputs, the *data input* and the *secret key input*:

- The data input is the UTF-8-encoded byte representation of each value listed below, in order, after normalization, with each value terminated by a single null byte (0x00):

From/Credential@domain

From/Credential/Identity

Sender/Credential@domain

Sender/Credential/Identity

Sender/Credential/CredentialMac@creationDate

Sender/Credential/CredentialMac@expirationDate
- The secret key input is the cXML shared secret used between the receiver and the third party.

Normalizing the Inputs

The values listed above must be normalized to remove differences in case and formatting before computation:

Value	Normalize by...	Example
domain	Use the lowercase version of the string, unless it is known to be case sensitive, for example, "AribaNetworkUserId". Note that "NetworkId" and "DUNS" are not case-sensitive.	networkid
Identity	Discard leading or trailing whitespace and use the lowercase version of the string.	an9900000100
creationDate expirationDate	No normalization needed, because they are in ISO8601 format described in "Date, Time, and other Data Types" on page 37.	2003-01-15T11:42:46-08:00

Do not normalize the shared secret.

MAC Algorithm

The only supported MAC algorithm value is "HMAC-SHA1-96", which corresponds to the HMAC-SHA1 algorithm, which produces a 160 bit (20 byte) output, and retaining only the left-most 96 bits (12 bytes). The 12 bytes are then base-64 encoded, yielding a 16-byte character string consisting only of characters in the set [A-Z a-z 0-9 +/].

▼ To computer the MAC:

1. Concatenate the UTF-8-encoded byte representation of the following strings, each followed by a null byte (0x00). (The strings have been normalized as described above):

“networkid”, “an9900000100”, “networkid”, “an9900000100”,
 “2003-01-15T08:42:46-08:00”, “2003-01-15T11:42:46-08:00”

The concatenation yields the following byte sequence:

```
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 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. Use HMAC-SHA1 to hash the above sequence with the receiver’s shared secret, for example, “abracadabra” (61 62 72 61 63 61 64 61 62 72 61), which yields:

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

Do not terminate or normalize the shared secret.

3. Truncate the above result to 96 bits (12 bytes):

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

Truncation helps increase the security of the hash.

4. Base-64 encode the above result to yield the final result:

```
cR6Jpz58nriXERDN
```

The trusted third party inserts the final result in ProfileResponse documents it sends to the entity that will be the client (document sender), and the client inserts it in a CredentialMac element in all direct communication to the server (document receiver).

ProfileResponse

The following cXML example shows a ProfileResponse sent from a trusted third party (such as a commerce network hub) to a client (such as a procurement application) so the client can send direct requests to the receiving server.

```
<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.ariba.com/ANCXMLDispatcher.aw/ad/cxml</URL>
      </Transaction>
      <Transaction requestName="PunchOutSetupRequest">
        <URL>https://service.ariba.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>
```

For more information about the Profile transaction, see Chapter 3, [“Profile Transaction.”](#)

CredentialMac

The following cXML document fragment shows an example CredentialMac element as it would be inserted by the client in documents sent directly to the server.

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

```

    </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="2003-01-15T08:42:46-0800">
            expirationDate="2003-01-15T11:42:46-0800">
              cR6Jpz58nriXERDN
            </CredentialMac>
            <UserAgent>Procure</UserAgent>
          </Credential>
        </Sender>
      </Header>
    [...]
  </cXML>

```

For more information about the CredentialMac element see [“Credential”](#) on page 41.

Auth Transaction

The Auth transaction allows receivers to validate organizations’ credentials through a mutually trusted third party. It should be used to authenticate received documents that do not contain either a shared secret or a MAC.

The receiver encloses the credential of the sender (the principal) in an AuthRequest document and sends it to the trusted third party for validation.

If the principal attempts to authenticate using a client digital certificate, the receiver includes both the principal’s credential and information about the principal’s certificate in the AuthRequest document. (The receiver obtains this certificate information from its Webserver or SSL/TLS implementation.)

The trusted third party receives the AuthRequest and looks up the principal’s credential to see if it is a recognized organization. If the principal’s certificate information was included, the trusted third party makes sure the certificate is valid and that it matches the organization associated with the credential.

If the credential (and optional certificate) authenticates, the trusted third party responds with a positive AuthResponse that contains the validated credential. If the credential is invalid, the trusted third party responds with an empty cXML response of status 403 (Forbidden).

The receiver can cache the results of the Auth transaction until the expiration date indicated in the AuthResponse. During this period, if the principal presents the same credential and certificate, the receiver need not send another AuthRequest.

AuthRequest

A request sent to a mutually trusted third party to authenticate an entity.

The following example includes X509 certificate information, which comes from the requesting entity's client digital certificate.

```
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/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>
```

```
        </AuthRequest>
    </Request>
</cXML>
```

Credential

A cXML credential.

For more information, see “[Credential](#)” on page 41.

X509Data

Describes the X.509 client certificate being used for authentication.

X509IssuerSerial

A container for the serial number and issuer name of the X.509 certificate.

X509IssuerName

The distinguished name of the issuer of the X.509 certificate. The distinguished name should be a string representation of an LDAP Distinguished Name, as described in RFC 2253. For example,

C=US, O="John Doe Data Security, Inc.", OU=Secure Server Certification Authority

X509SerialNumber

The serial number of the X.509 certificate.

X509SKI

The Subject Key Identifier of the X.509 certificate.

X509 SubjectName

The distinguished name of the subject of the X.509 certificate. This should be a string representation of an LDAP distinguished name, as described in RFC 2253.

X509Certificate

Contains the Base-64-encoded X.509v3 certificate.

X509CRL

Contains a Base-64-encoded X.509v3 Certificate Revocation List.

AuthResponse

Returns a list of valid credentials of the person entity in the AuthRequest document. Note that this response is for successful authentications only.

AuthResponse has the following attribute:

expirationDate (optional)	Specifies the time beyond which the information contained in the AuthResponse must be discarded. The inclusion of this attribute specifies that the receiver can cache the AuthResponse information until the expirationDate.
-------------------------------------	---

The absence of an expirationDate should be interpreted to forbid caching.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.011/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>
```

Chapter 16

cXML Digital Signatures

Any cXML request, response, or message can be signed using World Wide Web Consortium (W3C) XML Digital Signatures. Support for the XML Advanced Electronic Signature (XAdES) standard is also included.

This chapter will be of use to anyone signing cXML documents or verifying cXML digital signatures. It introduces digital signature elements and attributes in cXML 1.2.011 or later, discusses the properties of a valid cXML signature profile, and provides a complete example of a valid, correct cXML digital signature. Readers of this chapter should be familiar with electronic signature terminology and concepts such as asymmetric key pairs, certificates, and smart cards.

Digital Signature Overview

A digital signature is a series of bytes that contain cryptographic information, including the sender's public key and detailed information about the contents of the document being signed. The signature confirms the identity of the sender of an electronic document, and ensures that the document was not modified after it was generated by the signer. An XML digital signature—which is a specific arrangement of a digital signature—is an element that contains other information besides the cryptographic signature itself, including a list of what was signed, the signer's public key, and other attributes. A cXML signature is an XML digital signature of a certain form, as described later in this chapter.

W3C XML signatures and XAdESs have many options designed to allow for flexibility. These options are described in the following resources:

- For information about W3C XML digital signatures, see <http://www.w3.org>.
- For information about XAdES, see <http://uri.etsi.org/01903/v1.1.1/>.

Options for Signing

You can use a service to sign documents on your behalf, or you can implement the necessary hardware or software systems to sign the documents yourself. If you implement your own signing system, you must obtain a certificate signed by a Certificate Authority (CA) trusted by the receiver. Meeting receiver requirements might mean obtaining hardware that keeps the private key secret, such as a smart card or Hardware Security Module.

Note that signature and certificate requirements vary according to local laws and regulations. Prior to implementing a signing system, be sure to learn the requirements of the relevant locale.

Signing cXML Documents

A valid cXML digital signature is not just an XML signature, but an XML signature that uses particular options, has particular elements present, and signs (or does not sign) certain portions of the document.

cXML Digital Signatures

Note that namespace prefix conventions are used here when referring to elements that come from other specifications. All W3C XML Digital Signature elements use the `ds` prefix, and all XAdES elements use the `xades` prefix.

ds:Signature Element

The cXML element contains a space for the `ds:Signature` element after the Request, Response, or Message element. The `ds:Signature` element holds information about what is being signed, the signature, and the keys used to create the signature. It also has a place to store additional information such as XAdES extensions or attachment manifests.

The cXML element also contains a space for the `signatureVersion` attribute. For more general information about the cXML envelope, see page 36.

signatureVersion	If present, <code>signatureVersion</code> implies that the document is digitally signed, that is, that the document contains a valid <code>ds:Signature</code> element immediately following the Request, Response, or Message element. If the document is signed, this attribute must be present. The only valid value for the attribute is 1.0; other values are reserved for future use.
-------------------------	---

The Message, Request, and Response elements contain an Id attribute. For more complete information about the Message, Request, and Response elements, see the relevant sections of Chapter 2.

Id	This attribute can be used to call out an element and all its children as a target for signing. For example, if a document contains <Request Id="foo">, then in the digital signature <Reference URI="#foo"> will refer to the Request element and all its children. If the document is signed, this attribute must be present.
----	---

cXMLSignedInfo

The cXMLSignedInfo element includes cXML-specific details about the signature, and has the following attributes:

signatureVersion	Required. signatureVersion implies that the document is digitally signed, that is, that the document contains a valid ds:Signature element immediately following the Request, Response, or Message element. The only valid value for the attribute is 1.0; other values are reserved for future use.
payloadID	Required. The payloadID attribute is used to establish links between documents. The payloadID in the cXMLSignedInfo element must be the same as the payloadID in the document's main cXML element.

Signing Essentials

Because some information from the cXML header is significant, it must be signed. To sign these attributes from the header, repeat the information in a cXMLSignedInfo element placed within a ds:Object element. The ds:Object must be the first ds:Object in the signature. For example:

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

The value of the Id attribute must be "cXMLSignedInfo". The values of the signatureVersion and payloadID attributes must exactly match the values specified in the cXML element, and the receiver of the document must verify this match. No transforms should be used in this ds:Reference. This element must be signed via the first ds:Reference object in the ds:SignedInfo, as follows:

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

```
<ds:DigestValue>xxxxxxxxxxxxxxxxxxxxxxxxxxxx</ds:DigestValue>
</ds:Reference>
```

The Request, Response, or Message element should be signed in its entirety. To do this, specify the string "cXMLData" as the value of the Id attribute on the Request, Response, or Message element and include a ds:Reference element with the URI "#cXMLData" in the ds:SignedInfo. No transforms should be applied to this reference. This ds:Reference must be the second ds:Reference in the ds:SignedInfo.

The ds:KeyInfo element should be present with a single ds:X509Certificate element. This should include the Base64 encoding of the DER representation of an X.509 certificate containing the public key corresponding to the private key used to sign the document.

Using XAdES

The use of XAdES is a sender option, but it might be legally required in some situations. If XAdES is used, xades:QualifyingProperties should be the second ds:Object in the signature. The xades:SignedProperties element and all its children must be signed by specifying "XAdESSignedProps" as the value for the Id attribute of xades:SignedProperties and including a ds:Reference with the URI "#XAdESSignedProps" and no transforms in the ds:SignedInfo. Again, if XAdES is to be used, the certificate referred to in the xades:Cert element must be the same as that contained in the ds:KeyInfo element, the Id attribute of the ds:Signature element must be set to cXMLSignature and the Target attribute of xades:QualifyingProperties must be #cXMLSignature.

Signing Attachments

If the document in question includes attachments, digital signatures can be used to sign just the document, or both the document and its attachments. Signatures are structured in such a way that if the attachments are discarded, the signature on the document itself can still be validated.

The attachments should be signed using ds:Reference elements in a ds:Manifest element included under a ds:Object contained in the signature. The Id attribute of the ds:Manifest element must be "AttachmentManifest". The ds:Object should occur immediately after the ds:Object containing the xades:QualifyingProperties element, if it is present. Otherwise, it should occur immediately after the ds:Object containing the cXMLSignedInfo element.

Each ds:Reference in the manifest should use a URI with the "cid:" scheme to refer to the attachments through their MIME Content-Id. The ds:Manifest element itself should be signed using a fragment URI reference included in the ds:SignedInfo. This requirement exists because a compliant XML signature implementation must validate all the ds:Reference elements under ds:SignedInfo. Base validation ensures that the manifest

itself has not been corrupted, but will not validate the objects referred to in the manifest. This approach makes it possible to validate the document on its own if the attachments have been discarded. For example:

```
<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+lwPUgp2xqc=</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+lwPUgp2xqc=</ds:DigestValue>
</ds:Reference>
</ds:Manifest>
</ds:Object>
```

Error Status Codes for Digital Signatures

The following table lists cXML digital signature status codes:

Status	Text	Meaning
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 might be unacceptable, the type of certificate used might be unacceptable, or there might be some other problem.

Digital Signature Example

The following example shows a signed invoice. Note that the digest values and signature value are not correct, because parts of the invoice document have been abbreviated for this example.

```
<?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>
      <!-- Supplier -->
```

```

        <Credential domain="AribaNetworkUserId">
          <Identity>jdoe@company.com</Identity>
        </Credential>
      </From>
      <To>
        <Credential domain="AribaNetworkUserId">
          <Identity>smistry@company.com</Identity>
        </Credential>
      </To>
      <Sender>
        <!-- Supplier -->
        <Credential domain="AribaNetworkUserId">
          <Identity>jdoe@company.com</Identity>
          <SharedSecret>abracadabra</SharedSecret>
        </Credential>
        <UserAgent>Our Invoice Application</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-c14n
          20010315">
        </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>XlasOHckorH8fz/thdyZIZvV2yl=</ds:DigestValue>
    </ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
nNfsBpc22u9aypYLvgE5cuiHVO077vnaolS76LoAuks9bAwLO0kz/nkTQfb2zKSQTY8jj6W/
TJGCQj691PIKBnlqaMPPN3k+hbi6A5cJHPRd3HNPeXU5sSi4StTuxlWAIHe/
XEeBEclu7K6sR4Rh1gzzELg05v21aRX4oVGbjk=</ds:SignatureValue>
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIICgDCCAekCAw7cUTANBgkqhkiG9w0BAQQFADCBijELMAkGA1UEBhMCVVMxEzARBgNVBA.
..
zuRel/9tb8M95FuN5yR9GUGl5PgkzWuCQYobJqlcAs=</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Object>
    <cXMLSignedInfo Id="cXMLSignedInfo" payloadID="20030912.rsmith004@live.ariba.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.ariba.com, O="Ariba, 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>
```

Appendix A

New Features in cXML 1.2.011

cXML 1.2.011 contains the following new features:

- [CopyRequest Enhancement](#)
- [InvoiceDetailPaymentTerm Deprecated](#)
- [New PaymentTerm Element](#)
- [Digital Signature Support](#)
- [New SpendDetail Element](#)
- [New TimeCard Element](#)
- [New LeadTime Element](#)
- [Limited Changes in Use of UnitOfMeasure and UnitRate](#)
- [New preferredLang Attribute](#)
- [SearchDataElement Change](#)
- [New inspectionDate Attribute](#)
- [Followup Element Deprecated](#)

DTDs and documentation for this version are available at www.cXML.org.

CopyRequest Enhancement

Buyer and supplier organizations should now use attachments to fulfill copy requests. In cXML 1.2.011, the use of the cXML element as a child of CopyRequest is deprecated. Instead, use the cXMLAttachment element to attach another cXML document, whether or not it contains attachments itself.

For information on CopyRequest, see [“CopyRequest”](#) on page 165.

InvoiceDetailPaymentTerm Deprecated

InvoiceDetailPaymentTerm is deprecated in cXML 1.2.011, in favor of PaymentTerm(described below). cXML implementations for payment should use PaymentTerm.

New PaymentTerm Element

Defines the payment term in orders and invoices. This deprecates the InvoiceDetailPaymentTerm previously defined. PaymentTerm defines either the net term (without discount) or the discount term (with discount). For more information, see [“PaymentTerm”](#) on page 120.

Digital Signature Support

Any cXML Request, Response, or Message can now be signed using W3C XML Digital Signatures. Support for the XAdES (XML Advanced Electronic Signature) standard is also included. A valid cXML signature is an XML signature with certain properties, so additional signature elements now appear in all DTD files.

For information on digital signatures, see Chapter 16, “cXML Digital Signatures.”

New SpendDetail Element

The new SpendDetail element enables buyers and suppliers to automate the transaction process for spend in three new categories: travel, fees, and labor. Three new elements within SpendDetail, TravelDetail, FeeDetail, and LaborDetail, in addition to the Extrinsic element. For more information about SpendDetail, see [“SpendDetail”](#) on page 128.

TravelDetail Element

TravelDetail contains information about travel line items for air, car rental, hotel, and rail in ItemIn and ItemOut elements. In addition, cXML 1.2.011 supports the use of Extrinsic elements in SpendDetail, to enable buyer-supplier pairs to convey detailed information on spend that is not yet explicitly defined in cXML.

With cXML 1.2.011, users can access a travel booking provider's website and make travel reservations. To meet the particular requirements of the travel industry, `TravelDetail` supports an `OrderRequest` of type "delete", which can be sent to a travel booking provider after `PunchOut` to cancel an existing reservation. This also enables suppliers to cancel a leg of an air trip, for example, while replacing it with a new air leg. New transactions are in `cXML.dtd`.

For information on `TravelDetail`, see [“TravelDetail”](#) on page 129.

FeeDetail Element

The new `FeeDetail` element of `SpendDetail` conveys information about one-time or recurring fees that are not explicitly defined elsewhere in cXML. For example, a one-time fee for furniture rental would not fall into any category defined in `TravelDetail` or `LaborDetail`, but could be described in `FeeDetail`. For information, see `FeeDetail`.

LaborDetail Element

`LaborDetail` contains information about an item related to temporary labor. For information on `LaborDetail`, see [“LaborDetail”](#) on page 150.

Extrinsic

The use of Extrinsic details is now supported in `SpendDetail`, enabling buyer-supplier pairs to convey detailed information on spend that does not fit within `TravelDetail`, `FeeDetail`, or `LaborDetail`.

Extrinsic elements are intended to provide additional machine-readable information. They extend the cXML protocol to support features not required by all implementations. The cXML specification does not define the content of Extrinsic elements. Each buyer-supplier pair must agree on and implement their definitions of Extrinsic elements.

New TimeCard Element

Timecards for temporary employees can now be transmitted by cXML. The `TimeCard` element documents hours worked, dates worked, and pay rates. Two new requests contain the `TimeCard` element: `TimeCardInfoRequest`, which allows buyers to send time cards to suppliers or staffing agencies, and `TimeCardRequest`, which allows suppliers and staffing agencies to send timecards to buyers.

For information on TimeCard transaction, see Chapter 8, “TimeCard Transaction.”

New LeadTime Element

ItemDetail now contains an optional LeadTime element which describes the number of days needed for the buyer to receive the product in question. In an IndexItemAdd element, duplicate LeadTime information might come from both ItemDetail, where it is optional, and IndexItemDetail, where it is mandatory. If the LeadTime elements are defined in both cases, then they should be identical.

For information on LeadTime, see “[ItemDetail](#)” on page 108.

Limited Changes in Use of UnitOfMeasure and UnitRate

In the context of an InvoiceDetailServiceItem element or a LaborDetail element, UnitOfMeasure and UnitPrice are deprecated in cXML 1.2.011, and should not be used. Use UnitRate instead, because it includes the rate code. For some services, such as temporary labor, UnitRate is required.

UnitRate represents the amount to be paid per unit of time (or of some other measure). In the case of multiple UnitRates, each UnitRate should include a TermReference to distinguish it from others. TermReference is a generic base element that identifies the definition of the UnitRate in question.

For more information on UnitRate, see “[UnitRate](#)” on page 150.

New preferredLang Attribute

The Email element now contains an optional preferredLang attribute. This attribute allows optional language information to be associated with email addresses so that email messages are constructed in the preferred language.

For information on the preferredLang attribute, see “[Email](#)” on page 240.

SearchDataElement Change

A change to SearchDataElement allows suppliers producing catalog content to specify the searchable type name without being forced to specify additional searchable information about the item. Formerly, it was <ELEMENT SearchGroupData (Name, SearchDataElement+)>. Now, it is <ELEMENT SearchGroupData (Name, SearchDataElement*)>.

For information on SearchDataElement, see [“Type Definitions”](#) on page 277.

New inspectionDate Attribute

You can now specify inspection dates for invoice lines using a new optional attribute called inspectionDate, which has been added to the InvoiceDetailItem, InvoiceDetailServiceItem, and InvoiceDetailOrderSummary elements. The inspectionDate attribute represents the date when the transfer of goods or the delivery of services occurs according to legal tax definitions. In most locations, use of this attribute is optional.

For information on the inspectionDate attribute, see [“InvoiceDetailOrder”](#) on page 242.

Followup Element Deprecated

As of cXML 1.2.011, the use of the Followup element is strongly discouraged. In early implementations, Followup was used to specify the URL to which future StatusUpdateRequest documents should be posted.

All cXML implementations should use the more robust Profile transaction to retrieve and convey information about server capabilities, including supported cXML version, supported transactions, and options to those transactions. See [“Profile Transaction”](#) on page 24 for more information.

Index

A

- Accounting element 153
- AccountingSegment element 153
- Address element 119
- addressID attribute 119
- agreementDate attribute 199
- agreementID attribute 199
 - OrderRequestHeader element 118
- agreementItemNumer attribute 125
- AgreementItemOut element 200
- agreementPayloadID attribute
 - OrderRequestHeader element 118
- AirDetail element
 - about 136
 - AirLeg element 137
 - AirLegDestination element 138
 - AirLegOrigin element 138
 - TripType element 136
 - type attribute of TripType 137
- AirLeg element 137
 - about 137
 - AirLeg attributes 138
 - Airport element 139
 - arrivalTime attribute 138
 - BookingClassCode element 140
 - departureTime attribute 138
 - equipment attribute 138
 - flightNumber attribute 138
 - Meal element 140
 - Rate element 140
 - seatNumber attribute 138
 - seatType attribute 138
 - stops attribute 138
 - travelSegment attribute 138
 - upgrade attribute 138
 - Vendor element 138
- alternateAmount attribute 55
- alternateCurrency attribute 55
- < through ' entities 38
- ApprovalInfo element 191
 - approvedDate attribute 191
- arrivalTime attribute
 - AirLeg element 138
 - HotelDetail element 144
 - RailLeg element 147
- Attachment element 123
- attachments to purchase orders 154
- attachments, signing 326
- AvailablePrice element
 - CarRentalDetail element 144

B

- backordered status
 - header level 212
 - line-item level 219
- bedType attribute
 - RoomType element 145
- BillTo element 119
- booking orders 77
- BrowserFormPost element 101, 304, 305
- buyer and supplier cookies 86, 96
- BuyerCookie element 101, 105

C

- caching DTDs 23
- CarRentalDetail element
 - about 140
 - attributes 141
 - AvailablePrice element 144
 - BookingClassCode element 142
 - CarRentalDropoff element 141
 - CarRentalFee element 142
 - CarRentalPickup element 141
 - LimitedMileage element 143
 - Vendor element 141, 145, 148
- CarRentalFee element
 - attribute 143
 - values of type attribute 143
- CarrierIdentifier element 227
- carType attribute
 - RailLeg element 147
- Catalog.dtd 23
- character encoding 30
- character entities 37
- Charge element 153
- checkinTime attribute
 - HotelDetail element 144
- checkoutTime attribute
 - HotelDetail element 144
- Classification element 79
- code attribute 45, 307
- Comments element
 - about 123
 - ConfirmationHeader 216
 - ShipNoticePortion element 229

- ConfirmationHeader element 209
- ConfirmationItem element 208, 216
- ConfirmationRequest element 206, 208
- ConfirmationStatus element 217
- confirmID attribute 213
- Contact element 56, 151, 189, 191
 - about 121
 - attributes 121
 - ShipNoticePortion element 229
 - submittedDate attribute 191
- Contractor element 151, 189
- ContractorIdentifier element 151, 189
- cookies
 - buyer and supplier 86, 96
- copy node 160
- corporateURL attribute 272
- Credential element 41
- CredentialMac
 - in ProfileResponse 61
- CredentialMac options in ProfileRequest 60
- currency attribute 55
- cXML digital signature overview 323
- cXML Digital Signatures 324
- cXML element 36
- cXML license agreement ii
- cXML.dtd 23
- cxml.org Website 23
- cXML-base64 hidden field 52, 94
- cXMLSignedInfo element 325
- cXML-urlencoded hidden field 51, 94

D

- date and time format 37, 130
- debit and credit amounts
 - PCard orders 234
- DeliverTo element 119
- deliveryDate attribute 224
- departureTime attribute
 - AirLeg element 138
 - HotelDetail element 144
 - RailLeg element 147
- deploymentMode attribute 44, 50
- Description element 79, 108
- Meal element 135

- Dimension element 231
- direct marketplace 157
- direct PunchOut 110
 - CredentialMac 61
 - ProfileResponse 61
- Direct.AuthenticationMethod attributes 61
- Distribution element 152
- Document Type Definitions (DTDs) 22
- DocumentReference element 124, 159, 191, 203
- domain attribute 151
 - CarrierIdentifier element 227
 - Credential element 42
- dropoffTime attribute
 - CarRentalDetail element 141
- ds
 - Signature element 324
- DTDs (Document Type Definitions) 22

E

- earlyCheckinAllowed attribute
 - HotelDetail element 144
- EDI (X.12 Electronic Data Interchange) 19
- editors for XML 24
- effectiveDate attribute 59, 199
- encoding
 - character 30
- entities 37
- equipment attribute
 - AirLeg element 138
- error codes for digital signatures 327
- expirationDate attribute 199
- Extrinsic element 84, 96, 101, 304, 305
 - about 124
 - ConfirmationHeader element 216
 - of SpendDetail element 152
 - ShipNoticePortion element 229

F

- Fax element 122
- FeeDetail element
 - about 149

- flightNumber attribute
 - AirLeg element 138
- Followup element 58, 123
- form encoding 51, 94
- From element 81
- From, To, and Sender elements 41
- Fulfill.dtd 23

G

- GetPendingRequest element 297
- GetPendingResponse element 298

H

- Hazard element 231
 - ConfirmationHeader 215
- Header element 40
 - PunchOutSetupRequest 98
- HotelDetail element
 - about 144
 - Address element 145
 - attributes 144
 - attributes of RoomType element 145
 - AvailablePrice element 146
 - BookingClassCode 146
 - Meal element 146
 - Rate element 146
 - RoomType element 145
 - Vendor element 145
- HTML form encoding 51, 94

I

- Id attribute 325
- id attribute 153
- Index element 274
- IndexItemAdd element 275
- IndexItemDelete element 275
- IndexItemDetail element 276
- IndexItemPunchout element 275
- indirect marketplace 157
- inReplyTo attribute 50
- InvoiceDetail.dtd 23
- InvoiceDetailPaymentTerm element 242

InvoiceDetailRequest 167–??, 233–257
InvoiceDetailRequest element
 about 236
InvoiceDetailRequestHeader
 PaymentTerm element 242
InvoiceLaborDetail element 250
InvoiceRequest 234
InvoiceRequest element 234
invoices 235
 about 233
 cancel type 235
 credit memo type 235
 debit and credit amounts 234
 debit memo type 235
 detailed invoices 235
 dtd location 236
 header invoice 235
 individual and summary invoices 235
 information only type 235
 invoice levels 235
 shipping information 234
 specifying the purpose of 235
 standard type 235
 types 234
isAdHoc attribute 125, 127
isInternalVersion attribute
 OrderRequestHeader element 118
IsoCountryCode element 54
IsoLanguageCode element 54
isRecurring attribute
 FeeDetail element 149
isTriangularTransaction 246
isVatRecoverable 246
ItemDetail element 108, 276
 of an ItemOut 128
ItemID
 of an ItemOut 127
ItemID element 107
ItemIn element 106
ItemOut element 125

J

JobDescription element 151

L

LaborDetail element
 about 150
language
 in cXML header 88
lastReceivedTimestamp attribute 297
lastRefresh attribute 59
lateCheckoutAllowed attribute
 HotelDetail element 144
Launch Page 87
license agreement, cXML ii
LimitedMileage element
 attribute 143
lineNumber attribute 107, 125
 ShipNoticeItem element 230
loadmode attribute for catalogs 275
locale
 in cXML header 88

M

MAC options in ProfileRequest 60
marketplace credential 42
MasterAgreementRequest 197
MaxAmount element 199
maxMessages attribute 297
maxQuantity element 200
MaxReleaseAmount element 199
maxReleaseQuantity element 200
Message element 49
method attribute 227
MIME attachments 30, 154
MinAmount element 199
minQuantity element 200
MinReleaseAmount element 200
minReleaseQuantity element 200
Money element 55
multiple credentials 43

N

Name element 119
new elements
 PaymentTerm 242
non-catalog (ad-hoc) items 125

noticeDate attribute 212, 224
numberOfBed attribute
 RoomType element 145

O

operation attribute 81, 100, 199
 ConfirmationHeader element 213
 ShipNoticeHeader element 224
operationAllowed attribute 105
Order Receiver Page 95
orderDate attribute
 OrderReference element 209, 232
 OrderRequestHeader element 118
orderID attribute 209
 OrderReference element 209, 232
 OrderRequestHeader element 118
OrderInfo element 189
OrderMethods element 274
OrderReference element 209, 232
 ShipNoticePortion element 229
OrderRequest documents 112
OrderRequest element
 diagram 113
 example 114
 structure 113
OrderRequestHeader element 116
 attributes 118
 example 116
orderVersion attribute
 OrderRequestHeader element 118
OriginalDocument element 159
OriginatorCookie element 304

P

PackageIdentification element 228
Packaging element 230
PackagingCode element 230
parentAgreementPayloadID attribute 199
Path element 158
 of an ItemOut 127
path routing 157–??
payInNumberOfDays attribute 121
payloadID 159

payloadID attribute 36, 81, 204, 325
Payment element
 about 120
 payment, RequestToPay 204
 paymentDate 246
 PaymentRemittance.dtd 23
 PaymentTerm element 332
 about 120
 attribute 121
 InvoiceDetailRequestHeader 242
Period element 151, 189
pickupTime attribute
 CarRentalDetail element 141
pnrLocator attribute
 TravelDetail element 130
preferredLang attribute 241
Profile transaction 24
ProfileRequest element 58
ProfileResponse element 58
Provider PunchOut 301
ProviderDoneMessage 308
ProviderSetupRequest 302
ProviderSetupRequest element 304
ProviderSetupResponse 306
PunchOut index catalog 78, 102
PunchoutDetail element 277
PunchOutOrderMessage document 85
PunchOutOrderMessage element 104
PunchOutOrderMessageHeader element 105
PunchOutSetupRequest document 79
PunchOutSetupRequest element 99
PunchOutSetupResponse document 84
PunchOutSetupResponse element 103
purchase orders 111–155
 about 111
 attachments 154

Q

quantity attribute 107
 ConfirmationStatus element 218
 Dimension element 231
 ItemOut element 125
 ShipNoticeItem element 230

quoteExpirationTime attribute
 TravelDetail element 130
 quoteStatus attribute 105
 quoting orders 76

R

RailDetail element
 about 146
 TripType element 146
 values of type attribute of TripType 147
 RailLeg element
 attributes 147
 BookingClassCode element 148
 Meal element 148
 RailLegOrigin, RailLegDestination 148
 Rate element 148
 Vendor element 148
 rangeBegin attribute
 PackageIdentification element 228
 rangeEnd attribute
 PackageIdentification element 228
 remittance advice, about 167
 ReportedTime element 189
 Request element 44
 requestedDeliveryDate attribute 125
 requestName attribute 63
 RequestToPay 204
 requestToPay 207, 212
 requisitionID attribute 118, 125
 Response element 44
 ReturnData element 310
 ReturnValue element 310
 role attribute 121
 RoomType element
 bedType attribute 145
 numberOfBed attribute 145
 smoking attribute 145
 Route element 226
 router node 159

S

seatNumber attribute
 AirLeg element 138
 RailLeg element 147
 seatType attribute
 AirLeg element 138
 SelectedItem element 83, 102
 SelectedService element 304, 305
 Sender element 81
 Sender Page 91
 Sender, To, and From elements 41
 shipComplete attribute 118
 shipmentDate attribute 224
 shipmentID attribute 223
 ShipmentIdentifier element 228
 ShipNoticeHeader element 222
 ShipNoticeItem element 230
 ShipNoticePortion element 229
 ShipNoticeRequest element 221
 Shipping element
 about 120
 ShipTo element 119
 ShortName element 108
 signatureVersion attribute 36, 324, 325
 smoking attribute
 RoomType element 145
 SourcingStatus element 105
 SpendDetail
 grouping Extrinsics elements 152
 undefined spend 152
 SpendDetail element
 about 128
 Start Page 91
 StartPage element 103, 307, 308
 Status element 45, 307
 StatusUpdateRequest element 202
 stops attribute
 AirLeg element 138
 storeFrontURL attribute 272
 SubmitterInfo element 191
 Subscription element 285
 SubscriptionContentRequest element 287
 SubscriptionContentResponse element 287
 SubscriptionListRequest element 286
 SubscriptionListResponse element 286

- subsequentBuyer role 246
- Supervisor element 152
- supplier and buyer cookies 86, 96
- Supplier element 272
- SupplierChangeMessage element 284
- SupplierDataRequest element 282
- SupplierDataResponse element 283
- SupplierID element 78
- SupplierListRequest element 282
- SupplierListResponse element 282
- SupplierLocation element 273
- SupplierOrderInfo element 124
- SupplierPartAuxiliaryID element (Supplier Cookie) 86, 96, 276
- SupplierSetup element 83, 103

T

- Tax element
 - about 120
- taxPointDate 246
- TelephoneNumber element 122
- text attribute 307
- time and date format 37, 130
- TimeCard element 187, 188
- TimeCardInterval element 190
 - duration attribute 190
 - isNonBillable attribute 190
 - payCode attribute 190
- TimeRange element 190
 - endDate attribute 191
 - startDate attribute 191
- timestamp attribute 36, 81
- To element 81
- To, From, and Sender elements 41
- tools for working with XML 24
- trainNumber attribute
 - RailLeg element 147
- Transaction element 62
- TravelDetail
 - Money element of AvailablePrice 133

- TravelDetail element
 - about 129
 - AirDetail element 136
 - Airport element, about 134
 - attributes 130
 - AvailablePrice element 132
 - BookingClassCode, about 134
 - CarRentalDetail element 140
 - common elements in 130
 - HotelDetail element 144
 - Meal element, about 135
 - RailDetail element 146
 - SupplierID element 131
 - TermsAndConditions element 131
 - type attribute of AvailablePrice element 132
 - Vendor element 130
- travelSegment attribute
 - AirLeg element 138
 - CarRentalDetail element 141
 - HotelDetail element 144
 - RailLeg element 147
- TriangularTransactionLawReference 247
- type attribute
 - ConfirmationHeader element 211
 - ConfirmationStatus element 219
 - Credential element 42
 - Dimension element 231
 - OrderRequestHeader element 118
 - TripType element 137
- type definitions 277–281

U

- Unit of Measure 55, 87
- UnitOfMeasure element 249
- UnitPrice element 249
- UnitRate element 150, 249
- upgrade attribute
 - AirLeg element 138
- URL element 55, 304
- UserAgent element 41
- utilities for use with XML 24

V

- validating cXML 22
- Vendor element
 - attribute 130
 - CarRentalDetail element 141, 145, 148
- version attribute 36

W

- W3C XML Digital Signatures 323
- WorkLocation element 152

X

- XAdES, using 326
- xml
 - lang attribute 36
- xml:lang 88
- xmllanguageCode element 54



www.cxml.org