

**Ariba Network®**

# cXML Solutions Guide

**Release 12s3 (cXML version 1.2.024)**

November 2013



**Copyright © 1996–2013 Ariba, Inc. All rights reserved.**

This documentation, as well as the Ariba software and/or services described in it, contain proprietary information. They are provided under a license or other agreement containing restrictions on use and disclosure and are also protected by copyright, patent and/or other intellectual property laws. Except as permitted by such agreement, no part of the document may be reproduced or transmitted in any form by any means, electronic, mechanical or otherwise, without the prior written permission of Ariba, Inc.

Ariba, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in the documentation. The information contained in the documentation is subject to change without notice.

Ariba, the Ariba logo, AribaLIVE, SupplyWatch, Ariba.com, Ariba.com Network and Ariba Spend Management. Find it. Get it. Keep it. and PO-Flip are registered trademarks of Ariba, Inc. Ariba Procure-to-Pay, Ariba Buyer, Ariba eForms, Ariba PunchOut, Ariba Services Procurement, Ariba Travel and Expense, Ariba Procure-to-Order, Ariba Procurement Content, Ariba Sourcing, Ariba Savings and Pipeline Tracking, Ariba Category Management, Ariba Category Playbooks, Ariba StartSourcing, Ariba Spend Visibility, Ariba Analysis, Ariba Data Enrichment, Ariba Contract Management, Ariba Contract Compliance, Ariba Electronic Signatures, Ariba StartContracts, Ariba Invoice Management, Ariba Payment Management, Ariba Working Capital Management, Ariba Settlement, Ariba Supplier Information and Performance Management, Ariba Supplier Information Management, Ariba Discovery, Ariba Invoice Automation, Ariba PO Automation, Ariba Express Content, Ariba Ready, and Ariba LIVE are trademarks or service marks of Ariba, Inc. All other brand or product names may be trademarks or registered trademarks of their respective companies or organizations in the United States and/or other countries.

Ariba Sourcing solutions (On Demand and software) are protected by one or more of the following patents, including without limitation: U.S. Patent Nos. 6,199,050; 6,216,114; 6,223,167; 6,230,146; 6,230,147; 6,285,989; 6,408,283; 6,499,018; 6,564,192; 6,871,191; 6,952,682; 7,010,511; 7,072,061; 7,130,815; 7,146,331; 7,152,043; 7,225,152; 7,277,878; 7,249,085; 7,283,979; 7,283,980; 7,296,001; 7,346,574; 7,383,206; 7,395,238; 7,401,035; 7,407,035; 7,444,299; 7,483,852; 7,499,876; 7,536,362; 7,558,746; 7,558,752; 7,571,137; 7,599,878; 7,634,439; 7,657,461; and 7,693,747. Patents pending.

Other Ariba product solutions are protected by one or more of the following patents:

U.S. Patent Nos. 6,199,050, 6,216,114, 6,223,167, 6,230,146, 6,230,147, 6,285,989, 6,408,283, 6,499,018, 6,564,192, 6,584,451, 6,606,603, 6,714,939, 6,871,191, 6,952,682, 7,010,511, 7,047,318, 7,072,061, 7,084,998; 7,117,165; 7,225,145; 7,324,936; and 7,536,362. Patents pending.

Certain Ariba products may include third party software or other intellectual property licensed from a third party. For information regarding software or other intellectual property licensed from a third party, go to <http://www.ariba.com/copyrights.cfm>.

## Revision History

The following table provides a brief history of the updates to this guide. Ariba updates the technical documentation for its On Demand solutions when:

- Software changes delivered in service packs or hot fixes require a documentation update to correctly reflect the new or changed functionality;
- The existing content is incorrect or user feedback indicated that important content is missing.

Ariba reserves the right to update its technical documentation without prior notification. Most documentation updates will be made available in the same week as the software service packs are released, but critical documentation updates may be released at any time.

Document Version	Month/Year of Update	Updated Chapter/Section	Short Description of Change
1	April 2013	n/a	Reset revision history for release 12s3.
		Multiple chapters	Updated to support the new cXML elements and attributes added for the 12s3 features.
2	June 2013	Document Addressing and Security	Updated with information on with VendorID, VendorSiteID, and SSPPriateID domain.
		Invoices and Scheduled Payments	Added information about the ProviderDataRequest used by ICS providers to send incomplete or image-only invoices.
3	June 2013	Invoices and Scheduled Payments	Removed information about the ProviderDataRequest used by ICS providers to send incomplete or image-only invoices. This information is covered in the ICS guide.
		Document Addressing and Security	Clarified that the domains VendorID and VendorSiteID also apply to Invoice Automation and Purchase Order Automation with Oracle Fusion Middleware adapters.  Updated information on CC Invoices to specify that they are invoices sent from the Ariba procurement solution or the ERP system.
4	August 2013	Introduction to Catalog Upload	Updated with information on the maximum upload file size for catalogs.
		Receiving the Response	
5	September 2013	Invoices and Scheduled Payments	Updated with information on the extrinsic supported in invoices and credit memos for Polish suppliers.
		Purchase Orders	Added the limitation that Ariba Procurement and Invoicing Solutions support only item-level blanket purchase orders.

---

<b>Document Version</b>	<b>Month/Year of Update</b>	<b>Updated Chapter/Section</b>	<b>Short Description of Change</b>
6	October 2013	PunchOut Transactions	Added information that the sender credentials for punchOut users of Ariba procurement solutions is NetworkID.
7	November 2013	Invoices and Scheduled Payments	Removed information on the “Invalid” status no longer supported in the StatusUpdateRequest document for invoices.

---

---

# Table of Contents

<b>Revision History</b> . . . . .	<b>iii</b>
 <b>Chapter 1 Introduction to cXML Solutions</b> . . . . .	 <b>13</b>
cXML Recommended Usage . . . . .	13
Data Mapping . . . . .	13
cXML Versions . . . . .	14
cXML Versions Used by Ariba Applications . . . . .	14
Notification of cXML Changes . . . . .	15
 <b>Chapter 2 Data Conventions</b> . . . . .	 <b>17</b>
Number Format . . . . .	17
Money Element . . . . .	18
Money Format . . . . .	18
alternateCurrency and alternateAmount Attributes . . . . .	18
UnitOfMeasurement Element . . . . .	19
Name Formats . . . . .	20
PostalAddress Element . . . . .	21
Multiple Lines in Street Element . . . . .	21
Country, State, and PostalCode Elements . . . . .	22
TelephoneNumber Element . . . . .	23
CountryCode Element . . . . .	23
AreaOrCityCode and Number Elements . . . . .	24
Date and Time Format . . . . .	24
Extrinsic Elements . . . . .	24
 <b>Chapter 3 Document Addressing and Security</b> . . . . .	 <b>25</b>
Overview of Document Security . . . . .	25
Document Encryption . . . . .	25
cXML Document Authentication . . . . .	25
HTTPS Connections . . . . .	26
For Buying Organizations . . . . .	26
For Suppliers and Service Providers . . . . .	27
cXML Document Authentication . . . . .	27
Available Authentication Methods . . . . .	27
Certificate Authentication Set Up . . . . .	28
Authenticating Ariba Network's Certificate . . . . .	29
Digital Certificates . . . . .	30
Overview of Digital Certificates . . . . .	30
Client and Server Certificates . . . . .	30
Trusted Certificate Authorities . . . . .	31
Encryption Strength . . . . .	31
Domain Name in Server Certificates . . . . .	31
Obtaining a Digital Certificate . . . . .	32

cXML Credentials . . . . .	32
Supported ID Domains . . . . .	33
Multiple Credentials . . . . .	34
Case Sensitivity . . . . .	35
Domain Changes . . . . .	35
NetworkID and Older Versions of Ariba Buyer . . . . .	35
DigitalSignature Element . . . . .	36
Test Accounts . . . . .	36
Required Credentials . . . . .	37
Ariba Buyer . . . . .	37
Ariba Marketplace, Standard Edition . . . . .	39
Service Providers . . . . .	40
Suppliers . . . . .	41
Quick Enablement . . . . .	43
Using Purchase Orders . . . . .	44
Using ICS Invoices . . . . .	45
Using Payment Proposals . . . . .	48
Using CC Invoices . . . . .	50
 <b>Chapter 4 Profile Transaction . . . . .</b>	<b>53</b>
Using the Profile Transaction . . . . .	53
Sending ProfileRequest Documents . . . . .	53
Receiving ProfileRequest Documents . . . . .	54
From Credential Element . . . . .	54
Query Frequency . . . . .	54
ProfileRequest Document . . . . .	55
ProfileResponse Document . . . . .	56
Implementation Hints and Limitations . . . . .	57
HTTPS URLs . . . . .	57
Options for OrderRequest . . . . .	57
Ariba Network Test URLs . . . . .	57
 <b>Chapter 5 PunchOut Site Planning . . . . .</b>	<b>59</b>
PunchOut Versus Static Catalogs . . . . .	59
Catalog Hierarchy and Requisition Line Items . . . . .	59
Deciding Whether to Use PunchOut . . . . .	59
Implementation Methodology . . . . .	60
Planning . . . . .	60
Design . . . . .	63
Development . . . . .	64
Testing . . . . .	67
Deployment . . . . .	68
Becoming Ariba Ready Certified . . . . .	69
Retrofitting an Existing Website . . . . .	69
Leveraging an Existing Site . . . . .	69
Quick Retrofitting . . . . .	70
PunchOut for Services . . . . .	70
Services Exchange . . . . .	70
Create Session . . . . .	71
Edit Sessions . . . . .	71

Writing a PunchOut Deployment Guide .....	71
Connectivity Overview .....	71
Authentication and Identification .....	72
Required Extrinsic .....	72
Content Requirements/Specification .....	72
Address Information .....	72
Accounting Structure .....	73
Commodity Codes .....	73
Transactions Supported .....	74
<b>Chapter 6 PunchOut Transaction .....</b>	<b>77</b>
PunchOut Index Catalog .....	77
SupplierID Element .....	77
URL Element .....	77
Classification Element .....	78
punchoutLevel Attribute .....	78
Example PunchOut Index File .....	78
PunchOutSetupRequest Document .....	79
Credential Elements .....	79
UserAgent Element .....	80
PunchOutSetupRequest operation Attribute .....	80
BuyerCookie Element .....	81
Extrinsic Element .....	81
BrowserFormPost Element .....	82
SupplierSetup Element .....	82
SelectedItem Element .....	82
Example PunchOutSetupRequest Document .....	83
PunchOutSetupResponse Document .....	84
Overview of the PunchOutSetupResponse Documents .....	84
Status Element .....	84
StartPage URL Element .....	84
Example PunchOutSetupResponse Document .....	85
PunchOutOrderMessage Document .....	85
operationAllowed Attribute .....	85
cxml-base64 and cxml-urlencoded .....	86
BuyerCookie Element .....	86
Total Element .....	86
ItemIn Element .....	87
ItemID Element .....	87
ItemDetail Element .....	88
Example PunchOutOrderMessage Document .....	90
PunchOut URL .....	92
URL Specified on the Supplier's PunchOut Site .....	92
URL Specified on Ariba Network .....	92
URL Specified in the Supplier's Index Catalog (Deprecated) .....	92
URL vs. SelectedItem Elements .....	94

---

Level 2 PunchOut . . . . .	94
User Experience . . . . .	94
Level 2 PunchOut Format . . . . .	96
Level 2 PunchOut Requirements . . . . .	97
Example Index Catalogs . . . . .	98
Shelf-Level PunchOut . . . . .	102
Uploading Index Catalogs . . . . .	102
PunchOut Session Timeout . . . . .	103
ASP Examples . . . . .	103
receivePunchoutSetupRequest.asp . . . . .	103
resolveXML.asp . . . . .	105
TCcXMLFormatDTime.asp . . . . .	106
 <b>Chapter 7 Purchase Orders . . . . .</b>	 <b>109</b>
OrderRequestHeader Element . . . . .	109
orderID Attribute . . . . .	109
orderDate Attribute . . . . .	109
orderType Attribute . . . . .	109
releaseRequired Attribute . . . . .	110
effectiveDate Attribute . . . . .	110
expirationDate Attribute . . . . .	110
addressID Attribute . . . . .	110
Name Element . . . . .	110
DeliverTo Element, Line One . . . . .	110
DeliverTo Element, Line Two . . . . .	111
Street Element . . . . .	111
City Element . . . . .	111
State Element . . . . .	111
PostalCode Element . . . . .	111
CarrierIdentifier Element . . . . .	112
TransportInformation Element . . . . .	112
Country isoCountryCode . . . . .	113
TelephoneNumber Element . . . . .	113
Fax Element . . . . .	113
Email Element . . . . .	114
TermsofDelivery Element . . . . .	114
URL Element . . . . .	115
Tax Element . . . . .	115
Modifications Element . . . . .	116
ItemOut Element . . . . .	118
Distribution Element . . . . .	119
Receiving Types in Ariba Invoice Professional . . . . .	123
BlanketItemDetail Element . . . . .	124
Money currency Attribute . . . . .	125
Modifications Element . . . . .	125
TermsofDelivery Element . . . . .	125
PriceBasisQuantity Element . . . . .	126
Tolerances Element . . . . .	126
Ariba Network Currency Code Mapping . . . . .	127

---



Cancel Orders and Change Orders . . . . .	128
Cancel and Change Order Requirements . . . . .	128
Cancel Orders . . . . .	129
Change Orders . . . . .	130
Implementation Hints and Limitations . . . . .	131
Credential Elements . . . . .	131
Contact Roles . . . . .	131
Sold To Address and VAT ID . . . . .	132
Path Routing . . . . .	132
Non-Catalog Items . . . . .	133
Attachments . . . . .	134
PCards and vPayment . . . . .	135
Blanket Purchase Orders . . . . .	135
Maximum Document Size . . . . .	135
Extrinsic for Clickable Links . . . . .	136
Extrinsic for Legacy Purchase Orders . . . . .	136
Changes Introduced by Ariba Buyer . . . . .	136
Response Documents . . . . .	138
Example Response Document . . . . .	138
Purchase Order Acknowledgments . . . . .	139
Other Ways to Acknowledge Purchase Orders . . . . .	139
Blanket Purchase Order Acknowledgements . . . . .	140
Duplicate Documents . . . . .	140
 <b>Chapter 8 Order Confirmations and Ship Notices . . . . .</b>	 <b>141</b>
Overview . . . . .	141
Visibility of Order Status . . . . .	141
Ariba Network Account Configuration . . . . .	142
Transaction Flow . . . . .	143
ConfirmationRequest Documents . . . . .	144
ConfirmationRequest Element . . . . .	144
Implementation Hints and Limitations . . . . .	149
Example ConfirmationRequest . . . . .	150
ShipNoticeRequest Documents . . . . .	151
Implementation Hints and Limitations . . . . .	151
Example ShipNoticeRequest . . . . .	154
Attachments . . . . .	155
Serial Numbers and Asset Tags . . . . .	156
 <b>Chapter 9 Invoices and Scheduled Payments . . . . .</b>	 <b>157</b>
Basic Invoice Functionality . . . . .	157
Purchase Order Matching . . . . .	158
Matching on Ariba Network . . . . .	158
Matching in Ariba Buyer . . . . .	159
Invoicing Business Rules . . . . .	159
Rules that Affect PO-Flip Only . . . . .	159
Supplier Visibility of Business Rules . . . . .	160
Numeric Validation . . . . .	160

Implementation Hints . . . . .	161
eSignatures . . . . .	161
Supplier Self-Signed Invoices . . . . .	162
IdReference Element . . . . .	164
Contact Roles . . . . .	165
Conversion of Segment to AccountingSegment . . . . .	166
Invoice payloadID Attribute . . . . .	166
SerialNumber Elements . . . . .	166
PaymentTerm Element . . . . .	166
Remittance IDs . . . . .	167
Credits . . . . .	167
Inspection Date Attribute . . . . .	168
PriceBasisQuantity Element . . . . .	168
itemType Attribute . . . . .	168
Tax Exchange Rate Extrinsic . . . . .	169
Tax Invoice Extrinsic . . . . .	169
Tax Detail Category . . . . .	169
Tax on Shipping and Special Handling . . . . .	170
Withholding Tax Support . . . . .	172
Fields for Countries with a VAT System . . . . .	172
Attachments . . . . .	176
Maximum Document Size . . . . .	177
Extrinsic Elements . . . . .	177
Field Lengths . . . . .	178
PDF Invoice Copy Attachments . . . . .	179
Transmission to Buying Organizations . . . . .	180
Status and Cancel Invoices . . . . .	181
Invoice Status . . . . .	181
Cancel Invoices . . . . .	181
Example Invoices . . . . .	182
Summary Invoice . . . . .	182
Header-Level Invoice . . . . .	185
Blanket Purchase Order Invoice . . . . .	188
Complex Scenario . . . . .	191
Cancel Invoice . . . . .	201
Scheduled Payments . . . . .	203
For Information Only . . . . .	203
EFT Payment . . . . .	203
Discount Management . . . . .	203
Example Scheduled Payment . . . . .	204
End Points Integration with Ariba Network Adapters . . . . .	206
Enabling End Point Integration with Ariba Network Adapters . . . . .	207
End Points Integration in a Multiple ERP Setup . . . . .	208
<b>Chapter 10 Receipts . . . . .</b>	<b>209</b>
ReceiptRequest Routing . . . . .	209
cXML ReceiptRequest Document . . . . .	209
ReceiptRequestHeader . . . . .	210
ReceiptOrder . . . . .	210
Total . . . . .	213
Example Receipt . . . . .	214

<b>Chapter 11 Catalog Upload Transaction</b>	<b>217</b>
Introduction to Catalog Upload	217
Transaction Overview	217
Prerequisites	218
Sending a CatalogUploadRequest Document	219
To Element	220
CatalogUploadRequest Element	220
Attaching Your Catalog	222
Receiving the Response	222
Receiving Later Catalog Status	223
Email	224
URLOPost	224
 <b>Chapter 12 Get Pending/Data Download Transaction</b>	 <b>227</b>
Overview of Polling and Downloading	227
Polling Frequency	227
Polling Time	227
Download All Waiting Documents	228
Confirm the Receipt of Documents	228
GetPendingRequest	228
GetPendingResponse	230
DataRequest	231
DataResponse	232
Invoice Attachments	233
lastReceivedTimeStamp and Waiting Documents	233
Authenticating Waiting Documents	234
 <b>Chapter 13 cXML Transformations on Ariba Network</b>	 <b>235</b>
About Transformations on Ariba Network	235
Turning on Transformations	235
Moving Extrinsic to the Contact Element	236
Transforming ItemOut Extrinsic	237
Consolidating ItemOut to OrderRequestHeader	238
Mapping Custom Extrinsic to Ariba Network Extrinsic	238
Mapping Extrinsic from cXML to EDI	239
 <b>Appendix A Recommended Coding Systems</b>	 <b>241</b>
Commodity Codes	241
Currency Codes	242
Unit of Measure Codes	242
Country Codes	243
Language Codes	243
Dialing Codes	244
 <b>Index</b>	 <b>245</b>



---

## Chapter 1 Introduction to cXML Solutions

- “cXML Recommended Usage” on page 13
- “Data Mapping” on page 13
- “cXML Versions” on page 14

This guide describes recommended cXML usage and how to map data to and from cXML documents. The guidelines and recommendations in this document supplement the general description of cXML provided by the *cXML User's Guide*.

### cXML Recommended Usage

---

cXML is an open language defined by public Document Type Definitions (DTDs). These DTDs define cXML so that it is extremely flexible, which encourages its wide adoption.

Ariba applications use a specific implementation of the cXML language. To successfully interact with Ariba applications through cXML, you must understand the particular cXML behavior of those applications.

This document describes the external cXML behavior of the following Ariba applications:

- Ariba Buyer
- The Ariba Network (Ariba Network)

### Data Mapping

---

This document contains recommendations on mapping data to and from cXML documents.

cXML is a general-purpose language for conveying commerce related information. Organizations use it to communicate business data between diverse applications.

For example, trading partners use cXML for communicating purchase orders. When buying organizations generate cXML purchase orders, they include data from their Enterprise Resource Planning (ERP) systems and procurement systems. They route these purchase orders to their suppliers through Ariba Network. Suppliers then extract data from the purchase orders and use that data within their order-fulfillment systems.

Trading partners' back-end systems must be able to map data either to or from the cXML documents. The default configurations of these external systems do not always have the data or formatting necessary for the cXML protocol. Even if messages are well-formed cXML, the actual data within these elements might have many variations. This inconsistency affects both buying organizations and suppliers, and can lead to longer integration times.

This document helps implementors provide consistent data between trading partners. It provides best practices for the conversion of back-end-system data to cXML to help prevent variation in the data transferred between trading partners.

## cXML Versions

---

New versions of cXML are periodically released as the cXML standard is enhanced. For detailed information about each cXML version, see the *cXML Release Notes* on [www.cXML.org](http://www.cXML.org).

cXML-enabled applications must be able to detect the cXML version of received documents and process those documents appropriately.

### cXML Versions Used by Ariba Applications

Each Ariba application that connects to Ariba Network uses a specific cXML version:

Application	cXML Version
Ariba Buyer 7.0	cXML 1.1.008
Ariba Buyer 7.1	cXML 1.2.007
Ariba Buyer 8.0	cXML 1.2.008
Ariba Buyer 8.1	cXML 1.2.009
Ariba Buyer 8.2	cXML 1.2.011
Ariba Buyer 8.2.2	cXML 1.2.013
Ariba Buyer 9r1	cXML 1.2.019
Ariba Procure-to-Order	cXML 1.2.021
Ariba Procure-to-Pay	cXML 1.2.021
Ariba Services Procurement	cXML 1.2.021
Ariba Invoice Professional	cXML 1.2.021
Ariba Sourcing 4.0	cXML 1.2.008
Ariba Sourcing 4.1	cXML 1.2.009
Ariba Sourcing 4.2	cXML 1.2.011
Ariba Sourcing 4.3	cXML 1.2.011
Ariba Sourcing 4.4	cXML 1.2.011
Ariba Sourcing 9r1	cXML 1.2.019
Ariba Sourcing Basic/Professional	cXML 1.2.021
Ariba Analysis 2.5	cXML 1.2.009
Ariba Analysis 3.0	cXML 1.2.011
Ariba Analysis 9r1	cXML 1.2.019
Ariba Spend Visibility Basic/Professional	cXML 1.2.021
Ariba Category Management 2.0	cXML 1.2.011
Ariba Contract Management 9r1	cXML 1.2.019
Ariba Contract Management Basic/Professional	cXML 1.2.021
Ariba Supplier Information Management	cXML 1.2.021
Ariba Supplier Performance Management	cXML 1.2.021
Ariba Network	cXML 1.2.021
Ariba Discovery	cXML 1.2.021

## Notification of cXML Changes

Ariba notifies you about new cXML releases and changes in cXML behavior through Ariba Network and

Ariba Customer Support. Look in your Ariba Network account for notices of new cXML versions. Also, make sure the notification email addresses in your Ariba Network account are up-to-date.

cXML release notes are available on [www.cXML.org](http://www.cXML.org).





---

## Chapter 2 Data Conventions

- “**Number Format**” on page 17
- “**Money Element**” on page 18
- “**UnitOfMeasurement Element**” on page 19
- “**Name Formats**” on page 20
- “**PostalAddress Element**” on page 21
- “**TelephoneNumber Element**” on page 23
- “**Date and Time Format**” on page 24
- “**Extrinsic Elements**” on page 24

This chapter lists cXML data conventions, which ensure that data is consistent and can be used among multiple trading partners.

### Number Format

---

The World Wide Web Consortium (W3C) XML Data Proposal ([www.w3c.org/TR/1998/NOTE-XML-data-0105](http://www.w3c.org/TR/1998/NOTE-XML-data-0105)) defines data types used by cXML. It lists the following information for numeric data:

A number, with no limit on digits, may potentially have a leading sign, fractional digits, and optionally an exponent. Punctuation as in US English.

Numbers in cXML use “punctuation as in US English.” Other requirements are:

- Do not include symbols such as “\$” or “%”.
- Use only a period (.) as the decimal separator.
- You can optionally use commas (,) as thousands separators.
- Scientific notation is not allowed.

**Correct** examples:

```
<Discount>23</Discount>
<Discount>4.01</Discount>
<Discount>3001.01</Discount>
<Discount>3,001.01</Discount>
```

**Incorrect** examples:

```
<Discount>1.34%</Discount>
```

Should not include the percent symbol (%).

```
<Discount>1234,00</Discount>
```

Decimal separator must be a period (.).

```
<Discount>123.567.890</Discount>
```

Thousands separators are optional and if used, they must be commas (,).

---

## Money Element

---

The correct format for money values is:

```
<Money currency="USD">1.34</Money>
```

Specify currency with three-letter ISO 4217 currency codes. For a list of these codes, see “[Currency Codes](#)” on page 242.

### Money Format

cXML follows the W3C XML recommendation of “punctuation as in US English.” Do not include currency symbols, such as “\$.”

**Correct** examples:

```
<Money currency="USD">1234.00</Money>
<Money currency="USD">1,234.00</Money>
```

**Incorrect** examples:

```
<Money currency="USD">$1.34</Money>
```

Should not include the dollar symbol (\$).

```
<Money currency="US dollar">1.34</Money>
```

Incorrect ISO currency code.

```
<Money currency="USD">1234,00</Money>
```

Decimal separator must be a period (.).

**Note:** Although “XXX” is a valid currency code in ISO 4217, Ariba Network does not support the use of “XXX” as the currency code.

### alternateCurrency and alternateAmount Attributes

For cXML documents involving countries that use euros, you might need to include `alternateCurrency` and `alternateAmount` attributes:

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

You might also need to use these attributes to specify a “local” currency in invoices. For more information, see “[Mandatory VAT in Both Buyer’s and Supplier’s Local Currency](#)” on page 175.

Ariba Buyer 7.0 and later implements these attributes.

---

## UnitOfMeasurement Element

---

Use United Nations Units of Measure (UNUOM) codes to describe how items are packaged or delivered. The following table lists frequently used UNUOM codes:

UNUOM Code	Meaning
EA	each
BX	box
DZN	dozen
GRO	gross
HUR	hour
KGM	kilogram
LBR	pound
M4	monetary units
RO	roll
RL	reel
PR	pair
PK	pack

For more information, see “[Unit of Measure Codes](#)” on page 242.

Ariba Network can translate documents between cXML and Electronic Data Interchange (EDI). cXML and EDI UN EDIFACT both use the UNUOM standard, so Ariba Network performs no code translation between them. However, EDI X12 uses the ANSI UOM standard, so Ariba Network must translate UOM codes. Some UOM codes are the same in both standards, some codes are different, and some codes appear in both standards but mean different units of measurement.

It is vital that trading partners use UOM codes that map correctly. Do not use custom UOM codes, because the translation behavior is unknown and might produce inappropriate codes.

Some commonly encountered UOM mapping problems are:

- Buying organizations might want to convey “so much money worth of time” in labor requisitions. They should implement purchase orders to order by hour (HUR) or by units of money (M4). The code M4 means “monetary units” in both UNUOM and ANSI UOM.
- Trading partners might want to use the illegal code DOL or \$ to mean “US dollars.” To measure items in units of money, use code M4.
- Many buying organizations use ANSI UOM codes in their internal systems. If they accidentally allow ANSI UOM codes to appear in cXML documents, and Ariba Network translates those documents to X12 EDI, some codes might work, but others will produce incorrect data. To prevent confusion, they must use UNUOM codes in cXML.

For example, the code RL means “reel” in UNUOM and “roll” in ANSI. If a buying organization erroneously orders using RE for reel, the resulting X12 EDI purchase order happens to use the correct ANSI code for reel. The supplier might recognize that the item is sold in rolls and reject the purchase order. Or, the supplier might accept the purchase order and generate an X12 EDI invoice using the incorrect ANSI code RO for roll. Ariba Network translates RO to D65, which would cause an invoice-to-purchase-order mismatch.

The following table shows the mapping for RL and RO:

UNUOM Code	ANSI UOM Code	Meaning
RL	RE	reel
RO	RL	roll
D65	RO	round

**Correct** examples:

```
<UnitOfMeasure>EA</UnitOfMeasure>
<UnitOfMeasure>M4</UnitOfMeasure>
<UnitOfMeasure>HUR</UnitOfMeasure>
```

**Incorrect** examples:

```
<UnitOfMeasure>$</UnitOfMeasure>
<UnitOfMeasure>DO</UnitOfMeasure>
<UnitOfMeasure>DOL</UnitOfMeasure>
<UnitOfMeasure>HOUR</UnitOfMeasure>
```

## Name Formats

Procurement system implementations should parse name strings and determine how they are formatted. The code performing this parse must know whether a name is a personal name, a company name, or something else. For example,

```
<Name xml:lang="en-US">Workchairs, Inc.</Name>
<Name xml:lang="en-US">Smith, Bob</Name>
```

Personal name formats should match the national locale specified in the Name element by the xml:lang attribute. For English names, use “last, first [middle].”

Prefixes, such as Dr., should appear after the first name. Suffixes should appear after the last name: “last [suffix], first [prefix].” For example, Dr. Bob Smith III would be:

```
<Name xml:lang="en-US">Smith III, Bob Dr.</Name>
```

## PostalAddress Element

---

The cXML specification defines the data requirements for Name, DeliverTo, and Street portions of a PostalAddress element loosely, although it makes a relatively clear distinction between DeliverTo and Street information. Describe delivery data as an end point known to delivery services such as the Postal Service.

An address can direct to a specific person, identifying that person in either the Name or first DeliverTo element of the PostalAddress element. In common usage, an address might appear as:

John Smith  
c/o Acme, Inc.  
1565 Charleston Rd., M/S B.1  
Mountain View, CA 94043

or

Acme, Inc.  
Attn: John Smith  
1565 Charleston Rd., M/S B.1  
Mountain View, CA 94043

cXML supports both forms by putting the first line into a Name element, the second into a DeliverTo element and the third into a Street element. The resulting cXML documents contain addresses with personal names in either the Name or DeliverTo element. Many organizations do not include the company name at all. However, they should consistently use one format and include the company name in all addresses.

```
<Name xml:lang="en">Acme, Inc</Name>
<PostalAddress name="Headquarters">
  <DeliverTo>John Smith</DeliverTo>
  <DeliverTo>M/S B.1</DeliverTo>
  <Street>1565 Charleston Rd.</Street>
  <City>Mountain View</City>
  <State>CA</State>
  <PostalCode>94043</PostalCode>
  <Country isoCountryCode="US">United States</Country>
</PostalAddress>
```

It is recommended that the mail stop appear in a second DeliverTo element.

## Multiple Lines in Street Element

Early versions of Ariba Buyer placed all lines of an address into the same Street element, violating the cXML protocol. Ariba Buyer 6.1 patch 5 and later corrected this problem.

For information about transformations performed on this element by Ariba Network see Chapter 13, “[cXML Transformations on Ariba Network](#).”

Split Street elements where a comma might appear into additional Street elements. For example,

1565 Charleston Rd., Suite 45

should appear as two Street elements:

```
<Street>1565 Charleston Rd.</Street>
<Street>Suite 45</Street>
```

Ariba Network supports four Street elements per address.

## Country, State, and PostalCode Elements

The Country element is required. Without a valid Country element, the PostalAddress element is incomplete. For the isoCountryCode attribute value, use ISO 3166-1 two letter codes. For more information, see “[Country Codes](#)” on page 243. The Country value is a human-readable name.

Do not confuse the Country element with the CountryCode element. Use Country in postal addresses and CountryCode in telephone numbers.

If isoCountryCode="US", then the State element is required and must be a valid US Postal Service two-letter state abbreviation; for example, TX for Texas. PostalCode is also required and it must be a valid five- or nine-digit US Postal Service zip code. If using nine-digit zip codes, do not include a dash separator.

If isoCountryCode="CA", then the State element is required and must be a valid Canada Post two-letter province abbreviation; for example, QC for Quebec. PostalCode is required and must be a valid six-character Canada Post Postal Code. The format must be A9A9A9, with no space or separator.

The PostalCode element should not contain a dash (-) for any country. It is up to the receiving application to format the postal code for display. For example, a U.S. purchase order might contain:

```
<PostalCode>940431234</PostalCode>
```

Applications should display it for users as:

94043-1234

**Correct** example:

```
<State>FL</State>
<PostalCode>342300001</PostalCode>
<Country isoCountryCode="US">United States</Country>
```

**Incorrect** example:

```
<State>FLA</State>
<PostalCode>34230-0001</PostalCode>
<Country isoCountryCode="USA">United States</Country>
```

“FLA” is not a valid US Postal Service state code. Postal codes should not contain dashes. “USA” is not a valid ISO 3166-1 country code.

---

## TelephoneNumber Element

---

Complete telephone numbers consist of three elements: country code, area or city code, and dialed number.

cXML TelephoneNumber elements have the following format:

```
<TelephoneNumber>
  <CountryCode isoCountryCode="US">1</CountryCode>
  <AreaOrCityCode>650</AreaOrCityCode>
  <Number>9306200</Number>
</TelephoneNumber>
```

Example of a London phone number:

```
<TelephoneNumber>
  <CountryCode isoCountryCode="UK">44</CountryCode>
  <AreaOrCityCode>20</AreaOrCityCode>
  <Number>78628500</Number>
</TelephoneNumber>
```

## CountryCode Element

The CountryCode element contains an ISO 3166-1 two letter country code **and** an ITU dialing code.

For ISO 3166-1 Country Codes, see “[Country Codes](#)” on page 243. For ITU dialing codes, see “[Dialing Codes](#)” on page 244. Note that the ITU dialing code is not an ISO 3166-1 numeric country code (the United States has the value “840” in that system.)

Do not confuse the Country element with the CountryCode element. Use Country in postal addresses and CountryCode in telephone numbers.

**Correct** examples:

```
<CountryCode isoCountryCode="US">1</CountryCode>
<CountryCode isoCountryCode="CA">1</CountryCode>
<CountryCode isoCountryCode="FR">33</CountryCode>
<CountryCode isoCountryCode="MC">377</CountryCode>
```

**Note:** A single ISO Country Code can address multiple countries.

**Incorrect** examples:

```
<CountryCode isoCountryCode="US">US</CountryCode>
```

Contains no ITU dialing code.

```
<CountryCode isoCountryCode="US">011</CountryCode>
```

Contains “011”, which is the escape code for international dialing from the United States, instead of the ITU dialing code.

## AreaOrCityCode and Number Elements

Do not add “1” in the AreaOrCityCode element.

Do not use punctuation such as “(800) 555-5555” in the Number element.

Area and city codes have different lengths depending upon the locality. Dialing “escape” codes and country codes sometimes also appear in the Number element.

**Correct** example:

```
<AreaOrCityCode>800</AreaOrCityCode>
<Number>5555555</Number>
```

**Incorrect** examples:

```
<AreaOrCityCode>1800</AreaOrCityCode>
<Number>5555555</Number>
```

Contains a “1” before the area code.

```
<AreaOrCityCode>800</AreaOrCityCode>
<Number>555-5555</Number>
```

Telephone number should contain no punctuation.

## Date and Time Format

---

Use ISO 8601 format for date and time values. The format is YYYY-MM-DDThh:mm:ss-hh:mm. For example:

```
<OrderRequestHeader orderID="3333" orderDate="2001-12-20T09:25:57+01:00" type="new">
```

The “+01:00” component means one hour ahead of UTC (Coordinated Universal Time). Do not convert the time component to UTC time; leave it as local time.

**Incorrect** examples:

```
19990817T233535Z
```

```
2000-3-18T11:33:32
```

Both examples lack the required timezone information.

## Extrinsic Elements

---

Ariba Buyer 6.1 and 7.0 include a different set of extrinsics. By default, Ariba Buyer 6.1, includes the extrinsics "User" and "CostCenter". By default, Ariba Buyer 7.0 includes the extrinsics "UniqueName", "UserEmail", and "CostCenter". Any additional extrinsics would have to be agreed upon between the buying organization and the supplier.

**Note:** For older Ariba Buyer installations, it is recommended that buying organizations change "User" to "UniqueName" and notify their suppliers.

For a list of transformations performed on extrinsics by Ariba Network, see Chapter 13, “[cXML Transformations on Ariba Network](#).”



---

## Chapter 3 Document Addressing and Security

- “[Overview of Document Security](#)” on page 25
- “[HTTPS Connections](#)” on page 26
- “[cXML Document Authentication](#)” on page 27
- “[Digital Certificates](#)” on page 30
- “[cXML Credentials](#)” on page 32
- “[Required Credentials](#)” on page 37
- “[Quick Enablement](#)” on page 43

This chapter lists the document security and authentication requirements of Ariba cXML applications.

### Overview of Document Security

---

cXML documents travel between geographically disparate applications through corporate intranets and the public Internet. To keep business data confidential and to protect against unauthorized access, cXML-enabled applications support secure communication.

Secure communication is made possible through two mechanisms: document encryption and document authentication.

### Document Encryption

Applications encrypt cXML documents before transmission by sending them through HTTPS connections.

HTTPS is a secure form of HTTP (HyperText Transfer Protocol) that is supported by most Web servers and Web browsers. It protects network communication by encrypting data so that unauthorized parties are unable to interpret it.

For more information about HTTP, see “[HTTPS Connections](#)” on page 26.

### cXML Document Authentication

Applications authenticate received cXML documents to check their validity. Authentication ensures that each document is from a recognized organization.

Because cXML-enabled applications communicate through the Internet, they must perform authentication on all received cXML documents to prevent unauthorized access.

For more information about document authentication, see “[cXML Document Authentication](#)” on page 27.

---

## HTTPS Connections

---

Ariba Network is a service available on the public Internet. All cXML documents it receives or sends travel on the Internet.

To ensure secure document transmission through the Internet, Ariba Network requires all incoming and outgoing connections to be established through HTTPS. HTTPS connections allow web servers and clients to encrypt cXML documents for safe transmission over the Internet.

Ariba Network does not initiate or accept plain HTTP connection requests, therefore buying organizations, cXML-enabled suppliers, and service providers must support HTTPS.

HTTPS relies on encryption provided by a lower-level protocol called Secure Sockets Layer (SSL). SSL adds cryptological enhancements to TCP/IP (Transmission Control Protocol/Internet Protocol), the communications protocol of the Internet.

- **To receive cXML documents from Ariba Network**, external Web servers (including the Ariba Buyer 8.1 or later PunchIn site) must accept HTTPS connection requests and create HTTPS connections. You must install an SSL Web server certificate from a trusted certificate authority on your Web server. For more information about these certificates, see “[Overview of Digital Certificates](#)” on page 30.
- **To send cXML documents to Ariba Network**, all cXML-enabled applications must be able to initiate HTTPS connection requests.

All URLs entered in Ariba Network accounts must have values that begin with `https://`.

Ariba Network members must perform specific tasks to support HTTPS communication, as described in the following sections:

- “[For Buying Organizations](#)” on page 26
- “[For Suppliers and Service Providers](#)” on page 27

### For Buying Organizations

Configure Ariba Buyer and Ariba Sourcing to send documents to Ariba Network’s `https://...` URLs.

Ariba Buyer and Ariba Sourcing can successfully initiate HTTPS connections with Ariba Network because they come with a preinstalled CA certificate that enables them to recognize the issuer of Ariba Network’s SSL certificate.

For complete information on setting up HTTPS communication between Ariba applications, see the *Ariba Spend Management Integration Guide*.

If you use the collaborative invoicing PunchIn site, it must be enabled to accept HTTPS communication from Ariba Network. Also, you must ensure all URLs entered in your Ariba Network account specify `https://` addresses.

#### ▼ To check your URLs in your Ariba Network account:

(Required only if you use Collaborative Invoicing PunchIn site.)

- 1 Log in to your Ariba Network account.
- 2 In the Configuration area of your account, ensure the cXML ProfileRequest URL uses `https://...`

Ariba Network caches your cXML profile; you can clear this cache by clicking the **Reset Profile** button in your account. For more information, see “[Using the Profile Transaction](#)” on page 53.

## For Suppliers and Service Providers

Enable HTTPS on Web Servers connected to your cXML applications, and ensure all URLs entered in your Ariba Network account specify https:// addresses.

### ▼ To enable HTTPS on Web servers:

- 1 Ensure that your Web server is capable of supporting SSL (Secure Sockets Layer).
- 2 Ensure that your Web server is configured for DNS (Domain Name Service).
- 3 Purchase and install an SSL Web server certificate. For more information, see “[Obtaining a Digital Certificate](#)” on page 32.

Follow your Web server’s instructions for installing the certificate.

- 4 Ensure you send proper HTTPS headers in responses.
- 5 (Optional) Configure your Web server to disallow non-secure cXML HTTP requests.

### ▼ To check your URLs in your Ariba Network account:

- 1 Log on to your Ariba Network account.
- 2 In the Configuration area of your account, ensure all cXML URLs use “https://”. You can configure the following cXML URLs in your account:
  - ProfileRequest URL
  - PunchOutSetupRequest URL
  - OrderRequest URL

- 3 If you support the cXML Profile transaction, ensure your ProfileResponse returns https URLs.

Ariba Network caches your cXML profile; you can clear this cache by clicking the **Reset Profile** button in your account. For more information, see “[Using the Profile Transaction](#)” on page 53.

- 4 If you support the cXML PunchOut or ProviderSetup transactions, it is strongly recommended that you return https URLs in your PunchOutSetupResponse or ProviderSetupResponse documents.

## cXML Document Authentication

Ariba Buyer, Ariba Sourcing, Ariba Network, and cXML-enabled sites (such as PunchOut sites) are all applications that send and receive cXML documents. These applications authenticate all cXML documents they receive to ensure they are from valid organizations.

## Available Authentication Methods

When you configure your Ariba Network account, you select from two available cXML authentication methods:

- **Shared Secret**—(default) You enter a confidential text string into your Ariba Network account and configure your cXML application with that same string (if the shared secrets do not match, documents cannot be delivered). Then, those applications insert the shared secret string in cXML documents they generate. Each application authenticates received cXML documents by comparing the shared secret in them to the one it knows.

Shared secret authentication is simple to set up, it is free, and it requires little maintenance. To learn how to specify your shared secret, see the *Ariba Network Account Management Guide* (for suppliers) or the *Ariba Network Buyer Administration Guide* (for buying organizations).

- **Digital Certificate**—You purchase and maintain a client digital certificate from a trusted certificate authority. Then, you enter that certificate into your Ariba Network account. Ariba Network and your application refer to that digital certificate for authentication. The certificate does not appear in the cXML document or attached to the document; instead, the SSL protocol exchanges it before the document exchange takes place.

Digital certificate authentication requires more setup, certificates cost money, and they expire over time. However, it might be more compatible with your organization's security strategy.

**Note:** Buying organizations that use digital certificate authentication cannot use the Ariba Buyer Collaboration PunchIn site. The PunchIn site supports only shared secret authentication.

## Certificate Authentication Set Up

Organizations can set up digital certificates as an alternative to shared secrets for authenticating cXML documents.

To set up digital certificate authentication, obtain a signed client certificate, install it in your Ariba Network account, and change the URLs to which you post documents.

### ▼ To install your certificate on Ariba Network:

- 1 Obtain a signed client certificate from one of Ariba Network's trusted certificate authorities, as described in "[Obtaining a Digital Certificate](#)" on page 32.
- 2 Log in to your Ariba Network account. You must have account administrator privileges.
- 3 Go to the **Configuration** area of your account and click **cXML Setup**.
- 4 Set **Authentication Method** to "Certificate".
- 5 On your computer, open the signed certificate keystore file from the certificate authority in a text editor, such as Notepad.
- 6 Copy the text, starting with "-----BEGIN CERTIFICATE-----" and ending with "-----END CERTIFICATE-----".

## 7 Paste the text into the Certificate text box on Ariba Network.

**Authentication Method**

Ariba SN authenticates cXML documents sent by your organization

Select an authentication method: Certificate This selection will refresh the page content

**Certificate**

Please paste your Base64 encoded certificate here. It must begin with "----- BEGIN CERTIFICATE ----" and end with "---- END CERTIFICATE ----".

```
-----BEGIN CERTIFICATE-----
GYrVbWzVeBaVounICjrr8iQTT3NUkxOFOhu8HjS1iwWMuXeLsdfsIjGr
CVNukM57]
N3S5cEeRIIfjFnmusa5B3gjIGSvRRqpI1mQq14M0/ywqwWwZQ0oHhe
fTFPyhaO/q
8IkF5OQzwIDAQABo30wezANBgNVHQoEBjAEawIHgDAgBgNVHSUE
GTAXBgorBgEE
AYI3CgMDBgIghkgBhvCBAAEWsAYDVR0BBEEwP4AQDScp5AUql7R3
WDVHky0GuKEf
MB0xGzAZBgNVBAMTEIv3QgU0dIEF1dGhvcml0eYIKIj0R0Q5/e4
V0gDANBgkq
hkiG9w0BAQQAQFAAOCAQEA0oLuVTYlV0K5y6hwnEKORqfXmZHSzKlb
8qDGv8bbRff6
lt9pwEUkoWLPqYE54HeytuIo6Z8kc+GR3aX5pf3Gi7XA/
-----END CERTIFICATE-----
```

You can also use a backup digital certificate. Ariba Network uses the backup certificate if your primary certificate becomes invalid. Backup certificates are useful when your primary certificate expires or when you need to perform certificate maintenance

## 8 Click **Save**.

Digital certificates expire after a predetermined period of time. It is your responsibility to periodically obtain updated certificates from your certificate authority and install them on Ariba Network.

### ▼ To change your Ariba Network URLs:

Configure your cXML-enabled application to use the cXML Profile transaction to obtain the Ariba Network URLs for posting documents.

Ariba Network has a different set of URLs for shared-secret-authenticated and certificate-authenticated cXML documents. For example:

Authentication Method	Ariba Network URL
shared secret	<a href="https://service.ariba.com/service/transaction/cxml.asp">https://service.ariba.com/service/transaction/cxml.asp</a>
certificate	<a href="https://certservice.ariba.com/service/transaction/cxml.asp">https://certservice.ariba.com/service/transaction/cxml.asp</a>

The certificate URL works for both certificate- and shared-secret-based authentication.

The Profile transaction is the best way to look up the latest URLs for posting cXML documents.

## Authenticating Ariba Network's Certificate

If your cXML application authenticates Ariba Network by verifying Ariba Network's client certificate, verify certificate fields that do not change, such as Subject Distinguished Name and Issuer Distinguished Name. Do not verify certificates based on fields that regularly change, such as validity period or serial number.

---

## Digital Certificates

---

Digital certificates are integral to secure cXML communication. They are used both for enabling HTTPS connections and for authenticating cXML documents.

This section discusses digital certificates in terms of:

- [Overview of Digital Certificates](#)
- [Client and Server Certificates](#)
- [Trusted Certificate Authorities](#)
- [Encryption Strength](#)
- [Domain Name in Server Certificates](#)
- [Obtaining a Digital Certificate](#)

### Overview of Digital Certificates

Digital certificates use public key cryptography, which is a form of cryptography that uses two keys called a *matched keypair*. The matched keypair consists of a *private key*, which is known only to the owner of the keypair, and a *public key*, which is available to anyone.

The components of the matched keypair are related mathematically so that the encrypted text created using one component of the keypair can be decrypted by using only the other component of the keypair.

You obtain a signed certificate by sending a certificate signing request signed with your private key and containing your public key to a *trusted certificate authority*, which sends back a signed certificate containing your public key and signed with their private key.

**Note:** Digital certificates expire after a predetermined period of time. When they expire, HTTPS connection requests to your Web servers are rejected, and certificate-based cXML document authentication fails. You are responsible for renewing your certificates in a timely manner.

### Client and Server Certificates

There are two types of digital certificates:

- **server** certificates, used for SSL, which is required for accepting HTTPS connection requests. These certificates are also called *Web server certificates*, *secure server certificates*, and *secure server IDs*.

For more information about HTTP, see “[HTTPS Connections](#)” on page 26.

- **client** certificates, used for “certificate based cXML document authentication.”

For more information about cXML authentication, see “[cXML Document Authentication](#)” on page 27.

You can use a single certificate as both a server and a client certificate, depending on the information the certificate authority includes:

- If X.509v3 or Netscape extended key usage sections **are present**, the certificate cannot be used for any purpose not specified by the certificate (for example, server or client).
- If X.509v3 or Netscape extended key usage section **are not present**, the certificate can be used for any purpose (including server and client).

If you plan to use a single certificate for both SSL and certificate based authentication, ask your certificate authority to issue one that is both a server and a client certificate.

## Trusted Certificate Authorities

When you purchase a signed digital certificate, it must refer to an organization that is trusted by Ariba Network. You can use a digital certificate issued by any issuing organization, however it must reference a root certificate from a trusted Certificate Authority. The trusted Certificate Authorities are:

- ABAecom
- AddTrust
- American Express
- ANX Network
- Belgacom
- BelSign
- Deutsche Telekom AG
- Digital Signature Trust Co.
- Entrust
- Equifax
- GlobalSign
- GoDaddy
- GTE CyberTrust
- National Retail Federation
- Thawte
- TrustCenter
- United Parcel Service
- U.S. Department of Defense
- ValiCert
- VeriSign

## Encryption Strength

Ariba recommends use of 128-bit encryption or greater.

Common encryption strengths are 40, 56 and 128 bits; The greater the encryption bit width, the stronger the encryption, and the more secure the SSL connection.

Note that “128-bit certificates” are not necessary to support 128-bit encryption. Most “40-bit certificates” support 128-bit encryption or greater. “128-bit certificates” enable server-gated cryptography, which Ariba Network does not use. In most cases, you can use less expensive “40-bit certificates”; consult with your certificate authority about supported encryption strengths.

## Domain Name in Server Certificates

The CN (Common Name) field in server certificates for HTTPS must be a fully qualified DNS domain name of a Web server. For example, [www.workchairs.com](http://www.workchairs.com).

You must enter the same name in the transactive URL fields the Configuration area of your Ariba Network account and the name contained in your cXML ProfileResponse. Do not use IP addresses, because you cannot enter IP addresses in your Ariba Network account.

Use a separate certificate for each DNS name that clients will attempt to connect to. Certificate names do not specify Web server ports, so multiple Web server instances on different ports can use the same certificate. However, multiple Web servers cannot share a single certificate.

## Obtaining a Digital Certificate

Obtain a signed digital certificate from any of the certificate authorities listed in “[Trusted Certificate Authorities](#)” on page 31.

▼ **To obtain a signed certificate:**

- 1 Use your Web server or a third party tool, such as keytool or ikeyman, to generate a Certificate Signing Request (CSR). This is a file for describing your organization to a certificate authority.
- 2 Submit the CSR to an Ariba Network trusted certificate authority and request a **Base64-encoded X.509 V3 Class 3 signed digital certificate**.

Ariba Network’s trusted certificate authorities are listed in “[Trusted Certificate Authorities](#)” on page 31.

Request a *server certificate* for SSL or a *client certificate* for certificate based authentication. You might be able to obtain a certificate that works for both purposes; for more information, see “[Client and Server Certificates](#)” on page 30

- 3 The certificate authority returns a signed certificate.

Digital certificate files can be in binary Distinguished Encoding Rules (DER) format or base64-encoded Privacy-Enhanced Mail (PEM) format.

The contents of the file must begin with:

----- BEGIN CERTIFICATE -----

Likewise, the file must end with:

----- END CERTIFICATE -----

To use the certificate to enable HTTPS, see “[HTTPS Connections](#)” on page 26.

To use the certificate for cXML document authentication, see “[cXML Document Authentication](#)” on page 27.

---

## cXML Credentials

cXML Credential elements in the header of each cXML document identify the sender and receiver organizations.

The following topics describe Ariba Network requirements for cXML document credentials:

- [Supported ID Domains](#)
- [Multiple Credentials](#)
- [Case Sensitivity](#)
- [Domain Changes](#)
- [NetworkID and Older Versions of Ariba Buyer](#)
- [DigitalSignature Element](#)
- [Test Accounts](#)



## Supported ID Domains

Ariba applications use the following domains for cXML credential IDs:

- **NetworkID**—A unique alphanumeric value assigned to every organization registered on Ariba Network; for example, AN01000000123. Organizations can see their own and their trading partners' NetworkIDs by logging on to Ariba Network.
- **DUNS**—A unique number assigned to organizations by Dun & Bradstreet; for example, 942888711. To request a Dun & Bradstreet D-U-N-S® number or to see if your organization already has one, go to [www.dnb.com](http://www.dnb.com).
- **AribaNetworkUserId**—A login name of an Ariba Network user. These names typically have the format of an email address; for example, judy@workchairs.com. This domain is not preferred, because if users change their login names, cXML documents might fail to route.
- **PrivateId**—(for Quick Enablement or Supplier Connectivity only) A unique alphanumeric value defined by a buying organization for a particular supplier; for example, Supplier123. When Ariba Network receives documents from buying organizations, it maps these IDs to NetworkID. When Ariba Network sends documents to buying organizations, it maps suppliers' NetworkID, DUNS, and AribaNetworkID to PrivateId. Suppliers cannot use this domain.
- **ProviderId**—(for Quick Enablement only) A unique alphanumeric value defined by a service provider for a particular supplier; for example, Supplier123. When Ariba Network receives invoices from service providers, it attempts to map these IDs to NetworkID or PrivateId. If it cannot map IDs, it sends the invoices to the buying organization's **Unassigned Invoices** page. Buying organizations and suppliers cannot use this domain.
- **VendorID**—Supplier unique name, which is the supplier ID in the ERP system. It is an alphanumeric value. The combination of VendorID and VendorSiteID forms the *vendor compound key*, which uniquely defines a supplier. Ariba Network can use the vendor compound key for Quick Enablement. When Ariba Network receives documents from buying organizations, it maps these IDs to NetworkID. Suppliers cannot use this domain. This domain is applicable to buyers using on-demand Ariba Procurement and Invoicing Solutions and buyers using Invoice Automation or Purchase Order Automation with the Oracle Fusion Middleware adapter.
- **VendorSiteID**—Supplier location site ID and supplier location contact ID. Includes two IdReference domains, SiteID and SiteAuxID. These values come from the ERP system. This alphanumeric value is part of the vendor compound key. (See VendorID for more information.) Suppliers cannot use this domain. This domain is applicable to buyers using on-demand Ariba Procurement and Invoicing Solutions and buyers using Invoice Automation or Purchase Order Automation with the Oracle Fusion Middleware adapter.
- **SystemID**—(For Supplier Connectivity only) A unique alphanumeric value defined by a buying organization for a particular business application; for example, SAP1. When Ariba Network receives documents containing this ID domain, it associates them with the business application and Bill To addresses configured in the buyer's account.
- **SSPPrivateID**—(On-demand Ariba Procurement and Invoicing Solutions only) An automatically generated supplier location private ID (called the Ariba Network Private ID in Ariba Procurement and Invoicing Solutions). This ID is generated in Ariba Procurement and Invoicing Solutions for each supplier location. It uniquely identifies a supplier location. Prior to release 12s2, this value was sent using the PrivateID domain. To ensure proper routing of legacy documents that use PrivateID for specifying this auto-generated value, Ariba Network uses both SSPPrivateID and PrivateID to find matching suppliers in documents sent from the Ariba solution. Outbound documents do not include the SSPPrivateID domain. They use the PrivateID domain to include the supplier location private ID value. Suppliers cannot use this domain.

For more information about PrivateID, ProviderID, VendorID, and VendorSiteID, see “[Quick Enablement](#)” on page 43.

## 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 D-U-N-S number and a NetworkID number.

For example:

```
<To>
  <Credential domain="partition-oracle107">
    <Identity>1001134</Identity>
  </Credential>
  <Credential domain="DUNS">
    <Identity>123456789</Identity>
  </Credential>
  <Credential domain="NetworkID">
    <Identity>AN1000000123</Identity>
  </Credential>
</To>
```

Here is an example of the To credential from Ariba Procurement and Invoicing Solutions, which includes VendorID and VendorSiteID:

```
<To>
  <Credential domain="NetworkId">
    <Identity>AN01000252747</Identity>
  </Credential>
  <Credential domain="PrivateID">
    <Identity>v2034</Identity>
  </Credential>
  <Credential domain="SSPPPrivateID">
    <Identity>sid498__1000195__1000027</Identity>
  </Credential>
  <Credential domain="VendorID">
    <Identity>v2034</Identity>
  </Credential>
  <Credential domain="VendorSiteID">
    <Identity>
      <IdReference domain="SiteID" identifier="MainSite">
        <IdReference domain="SiteAuxID" identifier="contact2">
          </Identity>
        </IdReference>
      </IdReference>
    </Identity>
  </Credential>
</To>
```

The receiving system 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 receiving system 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.

## Case Sensitivity

The case sensitivity of credential data depends on the credential domain. NetworkID and DUNS values are case-insensitive; AribaNetworkUserId, PrivateId, ProviderId, VendorID, VendorSiteID, and SSPPriateID values are case sensitive.

For example, applications should not distinguish between the NetworkID values “an1234” and “AN1234.”

Applications should treat the domain names themselves (NetworkID, DUNS, and AribaNetworkUserId, PrivateId, ProviderId, VendorID, VendorSiteID, and SSPPriateID) as case insensitive, so for example, DUNS and duns are the same.

## Domain Changes

Ariba Network might change credential domains in documents routed to suppliers. In any cXML document routed to suppliers, Ariba Network changes AribaNetworkUserId to NetworkID in From credentials.

For example, as a PunchOutSetupRequest document routes from a buyer to a supplier, Ariba Network changes

```
<From>
  <Credential domain="AribaNetworkUserId">
    <Identity>tom@abcd.com</Identity>
  </Credential>
```

to

```
<From>
  <Credential domain="NetworkID">
    <Identity>AN01000000123</Identity>
  </Credential>
```

## NetworkID and Older Versions of Ariba Buyer

Some suppliers might not have D-U-N-S numbers. If they have customers using earlier versions of Ariba Buyer, those suppliers must be able to interpret **NetworkID** values used in the **DUNS** credential domain in PunchOutSetupRequest and OrderRequest documents.

### For Buying Organizations

Administrators using earlier versions of Ariba Buyer must load new suppliers using the **NetworkID** value in the **DUNS** domain, if that is the only value provided by the suppliers.

### For Suppliers

cXML-enabled suppliers can receive a mismatched domain value pair, where the domain is always DUNS, but the value could be NetworkId or DUNS, depending on what buying organizations have configured for suppliers.

cXML-enabled suppliers without D-U-N-S numbers that have customers on earlier versions of Ariba Buyer must be able to interpret NetworkID values in credentials, regardless of the domain. (For the greatest level of compatibility, cXML-enabled suppliers should interpret NetworkID and DUNS values, regardless of the domain in the To Credential.)

For example:

```
<To>
  <Credential domain="DUNS">
    <Identity>AN01000000123</Identity>
  </Credential>
</To>
```

or

```
<To>
  <Credential domain="NetworkID">
    <Identity>AN01000000123</Identity>
  </Credential>
</To>
```

or

```
<To>
  <Credential domain="DUNS">
    <Identity>123456789</Identity>
  </Credential>
</To>
```

To avoid having to implement this workaround, cXML-enabled suppliers should obtain D-U-N-S numbers, enter them into their Ariba Network accounts, and communicate them to their customers for use within Ariba Buyer.

## DigitalSignature Element

The cXML specification defines an element named `DigitalSignature` in the `Sender` element. Ariba applications do not use this element. Do not insert this element in cXML documents.

To authenticate documents, use either shared secrets or digital certificates. For information about authentication of documents passed between applications and Ariba Network, see “[cXML Document Authentication](#)” on page 27.

## Test Accounts

cXML applications should support both Ariba Network production accounts and Ariba Network test accounts. Test accounts are used by organizations during development of their applications to keep test data separate from production data, which helps prevent confusion. Ariba applications considers test accounts to be completely separate from production accounts.

To denote organization IDs for test accounts, append “-T” to the ID string. For example:

```
<Credential domain="DUNS">
  <Identity>942888711-T</Identity>
</Credential>

<Credential domain="NetworkID">
  <Identity>AN6565656565-T</Identity>
</Credential>
```

Applications should not distinguish between “-T” and “-t”; they should be case-insensitive.

## Required Credentials

Every cXML document routed to a cXML server must contain specific credentials in the Header section. Credentials are specified in the From, To and Sender elements. They allow receiving systems to identify, authenticate, and authorize parties.

As cXML documents travel to their destinations, intermediate nodes (such as Ariba Network) change the Sender element. The Sender element always specifies the most recent node in the transmission chain.

The following sections list the credentials in documents sent by cXML applications:

- [Ariba Buyer](#)
- [Ariba Marketplace, Standard Edition](#)
- [Service Providers](#)
- [Suppliers](#)

### Ariba Buyer

The following credentials are required in transactions between Ariba Buyer, suppliers, and Ariba Network.

#### Buyer to Supplier

- OrderRequest
- PunchOutSetupRequest

Element	Content
From	Ariba Buyer
To	Supplier
Sender	Ariba Buyer

For more information about credential limitations of older versions of Ariba Buyer, see “[NetworkID and Older Versions of Ariba Buyer](#)” on page 35.

#### Example

```
<From>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@acme.com</Identity>
  </Credential>
</From>
<To>
  <Credential domain="DUNS">
    <Identity>942888711</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@acme.com</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Ariba Buyer 7.0.6</UserAgent>
</Sender>
```

### Buyer to Ariba Network

- GetPendingRequest
- SubscriptionContentRequest
- SubscriptionListRequest
- SubscriptionStatusUpdateRequest
- SupplierListRequest
- SupplierDataRequest
- OrderStatusSetupRequest

### Example

```
<From>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@acme.com</Identity>
  </Credential>
</From>
<To>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@ariba.com</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@acme.com</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Ariba Buyer 7.1</UserAgent>
</Sender>
```

### Supplier PunchIn

Ariba Buyer 8.1 and later with the Supplier PunchIn Portal accepts cXML ProfileRequest documents sent by Ariba Network.

- ProfileRequest

Element	Content
From	Ariba Network
To	Ariba Buyer
Sender	Ariba Network

### Example

```
<From>
  <Credential domain="NetworkID">
    <Identity>AN10000001</Identity>
  </Credential>
</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN100000123</Identity>
  </Credential>
</To>
<Sender>
```

```

    <Credential domain="NetworkID">
      <Identity>AN10000001</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>
    <UserAgent>Ariba Network</UserAgent>
  </Sender>

```

## Ariba Marketplace, Standard Edition

The following credentials are required in transactions between Ariba Marketplace, Standard Edition (AM-SE), suppliers, and Ariba Network.

### Supplier to Marketplace

- ProviderSetupRequest during configuration
- OrderStatusSetupRequest

Element	Content
From	Supplier
To	Ariba Network
Sender	Marketplace

### Marketplace to Supplier

- PrivateOrganizationRequest

Element	Content
From	Marketplace
To	Ariba Network
Sender	Marketplace

### Example

```

<From>
  <Credential domain="NetworkID">
    <Identity>AN65656565</Identity> <!-- identity of the marketplace -->
  </Credential>
</From>
<To>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@ariba.com</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN65656565</Identity>
    <SharedSecret>abracadabra</SharedSecret>
    <!-- shared secret of marketplace -->
  </Credential>
  <UserAgent>Ariba Marketplace 7.5</UserAgent>
</Sender>

```

The order is forwarded to the supplier by Ariba Network using PunchOut.

## Service Providers

The following credentials are required in transactions between service providers, suppliers, and Ariba Network.

### Supplier to Service Provider

- ProviderSetupRequest during configuration

Element	Content
From	Supplier
To	Service provider
Sender	Ariba Network

### Example

```
<From>
  <!-- Supplier's identity -->
  <Credential domain="NetworkID">
    <Identity lastChangedTimestamp="2000-03-12T18:39:09-08:00">
      AN0133333333
    </Identity>
  </Credential>
</From>
<To>
  <!-- Service provider's identity -->
  <Credential domain="NetworkID">
    <Identity>AN0122222222</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@ariba.com</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Ariba Network v20</UserAgent>
</Sender>
```



## Suppliers

The following credentials are required in transactions between suppliers and Ariba Network.

### Supplier to Ariba Network

- GetPendingRequest
- ProfileRequest
- CatalogUploadRequest

Element	Content
From	Supplier
To	Ariba Network
Sender	Supplier

### Example

```
<From>
  <Credential domain="NetworkID">
    <Identity>AN10000000123</Identity>
  </Credential>
</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN01000000001</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN10000000123</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Our order-receiving system 3.0</UserAgent>
</Sender>
```

### Ariba Network to Supplier

- ProfileRequest initiated by Ariba Network

Element	Content
From	Ariba Network
To	Supplier
Sender	Ariba Network

**Example**

```

<From>
  <Credential domain="NetworkID">
    <Identity>AN01000000001</Identity>
  </Credential>
</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN10000000123</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN01000000001</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Ariba Network</UserAgent>
</Sender>

```

**Buyer to Supplier:**

- ProfileRequest initiated by buyer
- OrderRequest
- StatusUpdateRequest
- PaymentRemittanceRequest
- TimeCardRequest

Element	Content
From	Buyer
To	Supplier
Sender	Ariba Network

**Example**

```

<From>
  <Credential domain="NetworkID">
    <Identity>AN01000000456</Identity>
  </Credential>
</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN10000000123</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN01000000001</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Ariba Network</UserAgent>
</Sender>

```

**Supplier to Buyer:**

- ConfirmationRequest
- ShipNoticeRequest
- InvoiceDetailRequest
- TimeCardRequest

Element	Content
From	Supplier
To	Buyer
Sender	Supplier

**Example**

```

<From>
  <Credential domain="NetworkID">
    <Identity>AN1000000123</Identity>
  </Credential>
</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN0100000456</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN1000000123</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Our order-receiving system 3.0</UserAgent>
</Sender>

```

## Quick Enablement

Ariba Network Quick Enablement allows buying organizations and invoice conversion service (ICS) providers to create new Ariba Network accounts on behalf of suppliers. The buyers create these accounts by specifying the supplier's company information in a **Correspondent** element in the header of purchase orders, invoices, payment proposals, CC invoices (invoices sent from the Ariba procurement solution or the ERP system), and collaboration requests to the supplier or invoices from the supplier. Ariba Network uses this information to create the supplier's account.

Buying organization or service provider accounts must be enabled for Quick Enablement. ICS quick enablement must be enabled by Ariba Customer Support. Ariba Network ignores the **Correspondent** element for accounts that have not been enabled.

The **Correspondent** element is available in cXML 1.2.016 or later. It specifies the supplier's company information with a **Contact** element and it can optionally use a **preferredLanguage** attribute to specify the supplier's preferred language; any documents Ariba Network sends to the supplier will be in that language, if it is supported. If **preferredLanguage** is not specified or if it specifies a language that is not supported, Ariba Network sends documents to the supplier in English.

Following are the types of documents that can be used for quick enablement:

- Purchase orders (see “[Using Purchase Orders](#)” on page 44)
- Invoices (see “[Using ICS Invoices](#)” on page 45)
- Payment proposals (see “[Using Payment Proposals](#)” on page 48)
- CC Invoices (invoices sent from the Ariba procurement solution or the ERP system) (see “[Using CC Invoices](#)” on page 50)

## Using Purchase Orders

Procurement applications add the Correspondent element in purchase orders to create Ariba Network accounts for suppliers and to send them the purchase orders through email or fax.

The following example shows the header of a purchase order for Quick Enablement:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM
"http://xml.cxml.org/schemas/cXML/1.2.016/cXML.dtd">
<cXML payloadID="125xyzkjlkw" timestamp="2006-04-20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN20000000123</Identity> <!-- ID of buyer -->
      </Credential>
    </From>
    <To>
      <Credential domain="PrivateId">
        <Identity>SupplierABC</Identity> <!-- ID of new supplier-->
      </Credential>
      <Correspondent preferredLanguage="en-US">
        <Contact role="correspondent">
          <Name>ACME Supply, Inc.</Name>
          <PostalAddress name="default">
            <Street>123 Main Street</Street>
            <Street>Suite 101</Street>
            <City>Beamont</City>
            <State>TX</State>
            <PostalCode>77705</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="routing">orders@acme.com</Email>
            <!-- Email address for routing the PO -->
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>650</AreaOrCityCode>
              <Number>1234567</Number>
            </TelephoneNumber>
          </Phone>
          <Fax name="routing"> <!--Fax number for routing the PO -->
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>650</AreaOrCityCode>
              <Number>5555555</Number>
            </TelephoneNumber>
          </Fax>
        </Contact>
      </Correspondent>
    </To>
  </Header>
  <Sender>
```

```
<Credential domain="NetworkID">
  <Identity>AN20000000123</Identity> <!-- ID of buyer -->
  <SharedSecret>abracadabra</SharedSecret>
</Credential>
<UserAgent>Our Procurement App V2.0</UserAgent>
</Sender>
</Header>
```

## Supplier IDs

Buying organizations use the PrivateId domain in the To Credential to identify the supplier. Buying organizations using Ariba Procurement and Invoicing Solutions also use the VendorID, VendorSiteID, and SSPPPrivateID domains. Ariba Network assigns the supplier a NetworkID (AN-ID), but the buying organization can continue to use the other domains in subsequent documents.

## Routing

Buying organizations include either the supplier's email address or fax number in the Contact element to instruct Ariba Network how to route the purchase order. The Email or Fax element must have a name="routing" attribute; Ariba Network rejects purchase orders that do not have this attribute.

**Note:** If buying organizations provide both Email and Fax elements, Ariba Network rejects the purchase orders due to conflicting routing information.

## Taking Ownership of Accounts

Ariba Network routes the purchase order to the supplier along with an invitation to log in and complete the registration process. Ariba Network encourages suppliers to take ownership of these accounts, which changes them to regular supplier accounts.

Ariba Network uses the data in the Contact element to populate fields in the supplier's account. The Name, Street, City, and Country elements in the Contact element are required.

After the supplier logs in and takes ownership of the account, Ariba Network ignores any Correspondent element in subsequent purchase orders sent to that supplier, because the supplier might have set account values such as preferred routing method, company name, or company address.

Quick Enablement purchase orders are subject to the same Supplier Membership Program (SMP) thresholds as purchase orders sent to registered suppliers. If suppliers do not register and subscribe to the Supplier Membership Program after they reach the membership thresholds, Ariba Network rejects purchase orders addressed to them. For more information about SMP thresholds, see [www.ariba.com/supliermembership](http://www.ariba.com/supliermembership).

## Using ICS Invoices

Invoice conversion service providers add the Correspondent element in invoices to create Ariba Network accounts for suppliers.

**Note:** For detailed information on ICS invoices and the invoice conversion process, see the *Ariba Guide to Invoice Conversion*.

The following example shows the header of an ICS invoice for Quick Enablement:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM
"http://xml.cxml.org/schemas/cXML/1.2.016/InvoiceDetail.dtd">
<cXML payloadID="125xyzkjlkw" timestamp="2006-04-20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="ProviderId">
        <Identity>supplier123</Identity> <!-- ID of new supplier -->
      </Credential>
      <Correspondent preferredLanguage="en-US">
        <Contact role="correspondent">
          <Name>ACME Supply, Inc.</Name>
          <PostalAddress name="default">
            <Street>123 Main Street</Street>
            <Street>Suite 101</Street>
            <City>Beaumont</City>
            <State>TX</State>
            <PostalCode>77705</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="routing">sales@acme.com</Email>
            <!-- Email address for routing the Invoice-->
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>650</AreaOrCityCode>
              <Number>1234567</Number>
            </TelephoneNumber>
          </Phone>
          <Fax name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>650</AreaOrCityCode>
              <Number>1234567</Number>
            </TelephoneNumber>
          </Fax>
        </Contact>
      </Correspondent>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN20000000123</Identity> <!-- ID of buyer -->
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN200000777</Identity> <!-- ID of service provider -->
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Nifty Invoice Scanning V2.0</UserAgent>
    </Sender>
  </Header>
```

## Supplier IDs

Service providers identify suppliers with the PrivateId or ProviderId domain in the From element:

- The PrivateId domain is used if the supplier already exists.
- The ProviderId domain is used if the service provider cannot match the supplier on the paper invoice to a supplier on the buying organization's PO report or vendor master file. The first time the service provider is uses a particular provider ID, Ariba Network sends the invoice to the **Unassigned Invoices** page of the buying organization's Ariba Network account. Ariba Network uses the Quick Enablement process to create a private supplier account for the supplier.

## Taking Ownership of Accounts

Suppliers must contact Ariba Network Support to take ownership of Quick Enablement accounts created through invoicing.

Ariba Network uses the data in the Contact element to populate fields in the supplier's account. The Name, Street, City, and Country elements in the Contact element are required.

## Using Payment Proposals

Procurement applications use the Correspondent element in payment proposals to create Ariba Network accounts for suppliers and to send them the payment proposals through email.

The following example shows the header of a payment proposal for Quick Enablement:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM
"http://xml.cxml.org/schemas/cXML/1.2.016/cXML.dtd">
<cXML payloadID="125xyzkjlkw" timestamp="2006-04-20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN20000000123</Identity> <!-- ID of buyer -->
      </Credential>
    </From>
    <To>
      <Credential domain="PrivateId">
        <Identity>SupplierABC</Identity> <!-- ID of new supplier-->
      </Credential>
      <Correspondent preferredLanguage="en-US">
        <Contact role="correspondent">
          <Name>ACME Supply, Inc.</Name>
          <PostalAddress name="default">
            <Street>123 Main Street</Street>
            <Street>Suite 101</Street>
            <City>Beaumont</City>
            <State>TX</State>
            <PostalCode>77705</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="routing">payment@acme.com</Email>
            <!-- Email address for routing the Payment Proposal -->
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>650</AreaOrCityCode>
              <Number>1234567</Number>
            </TelephoneNumber>
          </Phone>
        </Contact>
      </Correspondent>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN20000000123</Identity> <!-- ID of buyer -->
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Procurement App V2.0</UserAgent>
    </Sender>
  </Header>
```

### Supplier IDs

Buying organizations identify suppliers with the PrivateId domain in the To Credential. Buying organizations using Ariba Procurement and Invoicing Solutions also use the VendorID, VendorSiteID, and SSPPPrivateID domains. Ariba Network assigns the supplier a NetworkID (ANID), but the buying organization can continue to use the other domains in subsequent documents. For more information about Credential domains, see [“Supported ID Domains”](#) on page 33.



## Routing

Buying organizations can include only the supplier's email address in the `Contact` element to instruct Ariba Network how to route the payment proposal. The `Email` element must have a `name="routing"` attribute; Ariba Network rejects payment proposals that do not have this attribute.

### Notes:

- Ariba Network does not support the `fax` element for payment proposal.
- If buying organizations provide both the `Email` and `Fax` elements, Ariba Network rejects the payment proposals due to conflicting routing information.
- If the supplier declines to take ownership of the account, the buying organization cannot send additional payment proposals to this supplier through Ariba Network.
- Ariba Network support only cXML version 1.2.016 and later for Quick Enablement payment proposals.
- Ariba Network does not support attachments in payment proposals. If a payment proposal contains an attachment, Ariba Network ignores it, but processes the payment proposal.

## Taking Ownership of Accounts

Ariba Network routes an invitation to the private supplier to log in and complete the registration process. Ariba Network encourages private suppliers to take ownership of these accounts, which changes them to regular supplier accounts.

Once the private supplier logs in and takes ownership of the account, the supplier can view the payment proposal from the Inbox. All subsequent notifications on the payment proposals are sent to the supplier through email.

Ariba Network uses the data in the `Contact` element to populate fields in the supplier's account.

Quick enablement payment proposals are not subject to the Supplier Membership Program threshold calculation. If suppliers do not register and subscribe to the Supplier Membership Program after they reach the membership thresholds, Ariba Network rejects payment proposals addressed to them. For more information about SMP thresholds, see [www.ariba.com/supliermembership](http://www.ariba.com/supliermembership).

## Using CC Invoices

This Ariba Network feature is available only for buyer systems that are enabled to allow quick enablement through CC Invoices sent to private suppliers.

A CC invoice is a cXML copy request document that contains an attached invoice sent from the Ariba procurement solution or the ERP system.

The CC invoice feature allows Ariba Network to store a copy of the invoice originating from these ERP systems. Suppliers can then log into their Ariba Network account and view the status of their invoices and monitor the progress of the invoice through your invoice reconciliation process.

External systems use the Correspondent element in CC invoices to create Ariba Network accounts for suppliers who are not yet registered on the Ariba Network and send them the welcome letters through email.

The following example shows the header of a CC invoice for Quick Enablement:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM
"http://xml.cxml.org/schemas/cXML/1.2.016/cXML.dtd">
<cXML payloadID="125xyzkjlkw" timestamp="2006-04-20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN20000000123</Identity> <!-- ID of buyer -->
      </Credential>
    </From>
    <To>
      <Credential domain="PrivateId">
        <Identity>SupplierABC</Identity> <!-- ID of new supplier-->
      </Credential>
      <Correspondent preferredLanguage="en-US">
        <Contact role="correspondent">
          <Name>ACME Supply, Inc.</Name>
          <PostalAddress name="default">
            <Street>123 Main Street</Street>
            <Street>Suite 101</Street>
            <City>Beamont</City>
            <State>TX</State>
            <PostalCode>77705</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="routing">payment@acme.com</Email>
            <!-- Email address for routing the CC Invoice-->
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>650</AreaOrCityCode>
              <Number>1234567</Number>
            </TelephoneNumber>
          </Phone>
        </Contact>
      </Correspondent>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN20000000123</Identity> <!-- ID of buyer -->
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Procurement App V2.0</UserAgent>
    </Sender>
  </Header>
```

## Supplier IDs

Buying organizations identify suppliers with the `PrivateId` domain in the `To Credential`. Buying organizations using Ariba Procurement and Invoicing Solutions also use the `VendorID`, `VendorSiteID`, and `SSPPPrivateID` domains. Ariba Network assigns the supplier a `NetworkID (ANID)`, but the buying organization can continue to use the other domains in subsequent documents. For more information about `Credential` domains, see “[Supported ID Domains](#)” on page 33.

## Routing

Buying organizations can include only the supplier's email address in the `Contact` element to instruct Ariba Network how to route the CC invoice. The `Email` element must have a `name="routing"` attribute; Ariba Network rejects CC invoices that do not have this attribute.

### Notes:

- Ariba Network does not support the `fax` element for CC invoices sent from the Ariba procurement solution or the ERP system.
- If buying organizations provide both the `Email` and `Fax` elements, Ariba Network rejects the CC invoices due to conflicting routing information.
- If the supplier declines to take ownership of the account, the buying organization cannot send additional CC invoices to this supplier through Ariba Network.
- Ariba Network support only cXML version 1.2.016 and later for Quick Enablement CC Invoice.
- Ariba Network does not support attachments in CC invoices sent from the Ariba procurement solution or the ERP system. If a CC invoice contains an attachment, Ariba Network ignores it, but processes the CC invoice.

## Taking Ownership of Accounts

Ariba Network routes an invitation to the private supplier to log in and complete the registration process. Ariba Network encourages private suppliers to take ownership of these accounts, which changes them to regular supplier accounts.

Once the private supplier logs in and takes ownership of the account, the supplier can view the CC invoices sent from the Ariba procurement solution or the ERP system from the `Inbox`. All subsequent notifications on the CC invoices are sent to the supplier through email.

Ariba Network uses the data in the `Contact` element to populate fields in the supplier's account.

Quick enablement CC invoices sent from the Ariba procurement solution or the ERP system are subject to the Supplier Membership Program (SMP) calculation only when the private supplier has taken ownership of the account on Ariba Network. Private suppliers who have not yet taken ownership of their account can continue to receive CC invoices and these are not calculated for SMP. For more information about SMP thresholds, see [www.ariba.com/suppliermembership](http://www.ariba.com/suppliermembership).



---

## Chapter 4 Profile Transaction

- “Using the Profile Transaction” on page 53
- “ProfileRequest Document” on page 55
- “ProfileResponse Document” on page 56
- “Implementation Hints and Limitations” on page 57

---

### Using the Profile Transaction

Every cXML server must accept the Profile transaction. This transaction allows cXML clients to find out which cXML requests servers support and their URLs. It enables applications to obtain the latest transaction URLs automatically without requiring manual intervention.

Ariba Network, suppliers, and service providers send and receive ProfileRequest documents. Buying organizations send ProfileRequest documents, and they receive them if they use the Ariba Collaboration PunchIn site or Ariba Sourcing.

### Sending ProfileRequest Documents

All cXML applications obtain Ariba Network’s URLs by sending a ProfileRequest document to a predetermined URL. To find out Ariba Network’s URLs, post a ProfileRequest document to one of the following URLs:

<https://service.ariba.com/service/transaction/cxml.asp>  
(for shared secret authentication)

<https://certservice.ariba.com/service/transaction/cxml.asp>  
(for digital certificate and shared secret authentication)

Each trading partner could have a different set of URLs on Ariba Network. For each of your trading partners, send a ProfileRequest document to Ariba Network to obtain Ariba Network’s URLs for that organization. Cache these URLs for up to 24 hours and use them for all communication to that trading partner. You can reduce communication overhead by querying for a trading partner’s URLs only if you have documents to send to that organization.

To obtain Ariba Network’s URLs, use Ariba Network’s ID (NetworkID: AN01000000001) in the To credential of the ProfileRequest document. To obtain a trading partner’s URLs on Ariba Network, use that partner’s ID in the To credential of the ProfileRequest document.

## Receiving ProfileRequest Documents

Ariba Network queries cXML applications to find out their supported transactions and their URLs. The following table lists where Ariba Network sends ProfileRequest documents for each type of application:

Application	Ariba Network Queries its cXML Profile by...
Supplier cXML Sites	Posting ProfileRequest documents to suppliers' cXML-enabled sites (such as PunchOut sites) by using the URL suppliers enter in the <b>Profile URL</b> field in their Ariba Network accounts.
Ariba Buyer 8.0 and earlier	Not supported.
Ariba Buyer 8.1 and later	Posting ProfileRequest documents to the Ariba Collaboration PunchIn site using the URL that buying organizations enter in the <b>Profile URL</b> field in their Ariba Network accounts.
Ariba Sourcing	Posting ProfileRequest documents to Ariba Sourcing using the URL that sourcing organizations enter in the <b>Profile URL</b> field in their Ariba Network accounts.

## From Credential Element

In ProfileRequest documents sent by Ariba Network, the From credential identifies the initiating trading partner, not Ariba Network.

cXML servers (such as supplier PunchOut sites) should not use this From credential to decide which services and URLs to provide. Ariba Network caches only one profile per cXML server, and it uses that profile for all requests to that server, regardless of the requesting organization. For example, Ariba Network stores only one profile per PunchOut site, not one profile for each buying organization that uses it.

## Query Frequency

Ariba Network queries cXML applications for their profiles when users request services from those sites. Ariba Network caches profiles for up to 24 hours. cXML applications should query Ariba Network for its profile once every 24 hours.

For example, when Ariba Buyer initiates a PunchOut session, Ariba Network immediately queries the supplier's PunchOut site for its profile and Ariba Network uses this profile information for all subsequent transactions it initiates to that site over the next 24 hours.

Each account on Ariba Network has a **Reset Profile** button that clears the organization's cXML profile cached on Ariba Network. The next time Ariba Network needs to send the organization a cXML document, it queries the site for its cXML profile instead of using the cached profile, which can be up to 24-hours old. This function is helpful when integrating applications with Ariba Network.

## ProfileRequest Document

cXML applications' transaction URLs can periodically change due to requirements of their internal configuration, reliability, and load balancing. Use the Profile transaction to look up Ariba Network's and your trading partners' current URLs.

Your application posts a ProfileRequest document to Ariba Network with the To credential set to Ariba Network's ID (NetworkID: AN01000000001), and Ariba Network responds with a ProfileResponse document that lists all the cXML transactions it supports and the URLs for those transactions. For each of your trading partners, your application also posts a ProfileRequest document to Ariba Network with the To credential set to the ID of the trading partner, and Ariba Network responds with a ProfileResponse document that lists all the cXML transactions that partner supports and the URLs for those transactions.

The following listing shows an example ProfileRequest you send to Ariba Network to obtain Ariba Network's URLs. It is a simple Request that contains an empty ProfileRequest element.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="456778-199@acme.com" xml:lang="en-US"
  timestamp="2001-03-12T18:39:09-08:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN0100000123</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN0100000123</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Download Application, v1.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <ProfileRequest />
  </Request>
</cXML>
```

Use your ID in the From and Sender credentials and Ariba Network's ID in the To credential. The NetworkID AN01000000001 always represents Ariba Network.

If your customer uses different procurement applications for their business organizations, specify the procurement application's System ID in the To credential. This ID helps Ariba Network identify the procurement application using the Bill To address associated to it.

The following listing shows an example ProfileRequest containing the System ID. This is a simple Request that contains an empty ProfileRequest element.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://svcdev.ariba.com/schemas/cXML/1.2.017/cXML.dtd">
<cXML timestamp="2007-09-12T04:52:36-07:00"
  payloadID="1189597956670-4363355987563302335@10.10.13.240">
```

```

<Header>
  <From>
    <Credential domain="NetworkID">
      <Identity>AN01000000123</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="NetworkID">
      <Identity>AN01000000001</Identity>
    </Credential>
    <Credential domain="SystemID">
      <Identity>SAP001</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkID">
      <Identity>AN01000000123</Identity>
      <SharedSecret>abracadabra</SharedSecret>
    </Credential>
    <UserAgent>Supplier</UserAgent>
  </Sender>
</Header>
<Request>
  <ProfileRequest />
</Request>
</XML>

```

## ProfileResponse Document

The following listing shows an example ProfileResponse returned by Ariba Network.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="9949494-189@ariba.com" xml:lang="en-US"
  timestamp="2001-03-12T18:39:10-08:00">
  <Response>
    <Status code="200" text="OK"/>
    <ProfileResponse effectiveDate="2001-03-12T18:39:10-08:00">
      <Transaction requestName="ConfirmationRequest">
        <URL>https://service.ariba.com/service/transaction/cxml.asp</URL>
      </Transaction>
      <Transaction requestName="PunchOutSetupRequest">
        <URL>https://service.ariba.com/service/transaction/cxml.asp</URL>
      </Transaction>
      <Transaction requestName="InvoiceDetailRequest">
        <URL>https://svccoms.ariba.com/service/transaction/cxml.asp</URL>
      </Transaction>
      .
      .
      .
    </ProfileResponse>
  </Response>
</cXML>

```

This example lists only three transactions (ConfirmationRequest, PunchOutSetupRequest, and InvoiceDetailRequest); the actual ProfileResponse lists many more transactions. Your application caches this information and uses these URLs for all communication with Ariba Network or the trading partner for the next 24 hours.



Possible status codes in the response are:

- 200 – Success
- 4xx – Unrecoverable Error
- 5xx – Temporary Error. Client should retry this request

## Implementation Hints and Limitations

---

The following implementation information can help you successfully create ProfileResponse documents:

- [HTTPS URLs](#)
- [Options for OrderRequest](#)
- [Ariba Network Test URLs](#)

### HTTPS URLs

Ariba Network requires your ProfileResponse documents to return URLs with “https://” prefixes.

For more information about HTTPS, see “[HTTPS Connections](#)” on page 26.

### Options for OrderRequest

The cXML specification describes the following options for OrderRequest in ProfileResponse documents:

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

The ProfileRequest generated by Ariba Network does not contain these options, although Ariba Network does accept order attachments and change orders.

Similarly, Ariba Network does not interpret these options if they appear in ProfileRequest documents from suppliers.

### Ariba Network Test URLs

Ariba Network has three test URLs you can use to exercise your cXML application’s handling of permanent errors (4xx) and temporary errors (5xx). Use these URLs as negative test cases.

- <https://service.ariba.com/ANCXMLDispatcher.aw/ad/cxml400Test>  
This URL returns a random 4xx status, regardless of your cXML document. Use this URL to test your cXML application’s handling of permanent errors.
- <https://service.ariba.com/ANCXMLDispatcher.aw/ad/cxml500Test>  
This URL returns a random 5xx status, regardless of your cXML document. Use this URL to test your cXML application’s handling of temporary errors.
- <https://service.ariba.com/ANCXMLDispatcher.aw/ad/cxmlUniqueTest>  
This URL returns a 200 status half of the time and a 5xx status half of the time. Use this URL to test your cXML application’s retry capability.



---

## Chapter 5 PunchOut Site Planning

- “PunchOut Versus Static Catalogs” on page 59
- “Implementation Methodology” on page 60
- “Retrofitting an Existing Website” on page 69
- “PunchOut for Services” on page 70
- “Writing a PunchOut Deployment Guide” on page 71

This chapter describes how to plan PunchOut sites and promote them to your customers. Use this chapter as a guide to help avoid potential pitfalls that can cause delays in production. By following the methodology suggested here, you can create a solution for one customer that can scale to additional customers.

### PunchOut Versus Static Catalogs

---

PunchOut allows you to offer products and services with live pricing and company-specific customization. PunchOut offers a Web shopping experience that static catalogs cannot offer, but it requires more development effort.

Ariba Buyer displays both static and PunchOut catalog items in the local catalog hierarchy, enabling users to choose items and insert them in requisitions. However, static catalog content resides completely in Ariba Buyer, but PunchOut catalog content is hosted by your website.

### Catalog Hierarchy and Requisition Line Items

While static catalogs populate both the catalog hierarchy and requisition line items, PunchOut uses separate vehicles to populate the catalog hierarchy and requisition line items. PunchOut first requires you to provide a PunchOut index catalog that your customer loads into Ariba Buyer. Second, your PunchOut site sends cXML messages to Ariba Buyer to populate requisitions with line items.

### Deciding Whether to Use PunchOut

It is important to determine whether PunchOut is right for you, because development of a PunchOut site can be time-consuming and costly. If you retail only a small number of products, a static catalog might be more appropriate than a PunchOut site.

Also, consider whether your customers will use PunchOut. Customers can configure Ariba Buyer to allow or disallow PunchOut; consult with your customers to see whether they allow it.

---

## Implementation Methodology

---

The process of implementing a PunchOut site spans from initial evaluation of your existing e-commerce system, to development, to deployment. The more effort you spend in the early stages of implementation, the more likely you will be able to deploy your PunchOut site on time with the desired functionality.

The steps in creating a PunchOut site are:

- **Planning**
- **Design**
- **Development**
- **Testing**
- **Deployment**
- **Becoming Ariba Ready Certified**

### Planning

In the planning phase, you create a high-level vision of the integration between Ariba Buyer, Ariba Network, your PunchOut site, and your order processing system. In this step, you develop an estimate of the project plan and required resources.

#### Analysis of Current and Future State

Before developing a PunchOut site, perform an in-depth analysis of your current e-commerce and order processing systems to find out how you need to modify them.

##### Current Site

Some of the factors to consider for your current e-commerce site are:

- Do you have an existing e-commerce site?
- Does your site have XML integration enabled?
- Do you currently receive orders electronically through XML or EDI?
- How is integration performed with your accounting systems?
- Do you use your customers' control data (for example, ship location IDs) to identify pricing and shipping of orders?
- Does your site currently have the ability to deliver custom pricing or content based on the customer ID?

##### Planned PunchOut Site

Some of the factors to consider for the planned PunchOut site are:

- What is the process flow from shopping through order placement and fulfillment?
- How much customization per company do you want to provide?
- Can a single product line be selected for an initial pilot of your PunchOut site?
- Will you need to interface with multiple XML-based procurement applications?
- How will you model a "shopping cart" in requisition line items?

## Outsourcing Versus Internal Development

Next, review the resources you have available for implementing your PunchOut site. Building a PunchOut site requires programming expertise, which you can develop within your organization or outsource. Some key factors to take into account are:

- Does your technical staff have the skills to implement a PunchOut site?
- Do you have an existing Web infrastructure that can be leveraged to use cXML?
- Do you have an approved budget for e-commerce initiatives, in particular, PunchOut enablement?
- Have you evaluated the process flow with PunchOut to verify that it fits into your business model? PunchOut can be used for both products and services.

## Key Participants

The Ariba PunchOut integration process is a collaborative effort that leverages your in-house and external resources. There are three potential key participants: technical developer, integration manager, and Ariba Supplier Consulting.

### Technical Developer

Identify a team member who will be the PunchOut site technical developer. This person acts as the primary owner of the PunchOut process and assumes responsibility for a number of tasks. The following is a role description for the technical developer:

- Reads PunchOut documentation available from the following websites:  
[supplier.ariba.com](http://supplier.ariba.com)  
[www.cXML.org](http://www.cXML.org)
- Becomes familiar with Ariba Network methodology; develops or possesses a strong working knowledge of catalog formats, including:
  - CIF (Catalog Interchange Format)
  - cXML
  - PunchOut
- Tests the PunchOut site with Ariba Network and with customers
- Manages the Ariba Network supplier account for all PunchOut-related matters
- Performs ongoing maintenance of the PunchOut site

### **Integration Manager**

Identify a team member who will be the PunchOut site integration manager. This person is the point of contact for non-technical, business issues relating to PunchOut. The following is a role description for the Supplier Integration Manager:

- Verifies the ANID (Ariba Network ID) or Dun & Bradstreet D-U-N-S (Data Universal Numbering System) number for your company
- Manages PunchOut relationships with customers, resolving issues such as:
  - Identifying targeted commodities for PunchOut
  - Supplying commodity codes for PunchOut products to the customer
  - Defining the purchase order and invoice processes with the customer
- Creates a project plan and implementation time line
- Defines additional resource requirements and makes appropriate assignments

### **Ariba Supplier Consulting**

Ariba Supplier Consulting is a consulting group that provides targeted support with specific tasks such as Ariba Network registration, catalog creation, PunchOut site development, and integration testing. Ariba Supplier Consulting supports suppliers at any level of technical expertise.

For more information, see [www.ariba.com/suppliers/supplier\\_consulting.cfm](http://www.ariba.com/suppliers/supplier_consulting.cfm).

### **Determining the Level of Support Needed**

Your business requirements, order volume, amount of desired automation, and team expertise determine the level of support you need. Answering the following questions can help determine what you need.

- **Does your team have a solid understanding of XML and cXML?**

XML provides the building blocks of all cXML documents. cXML documents provide a way for buyers, suppliers, and Ariba Network to communicate with each other across the Internet. cXML documents are constructed based on Document Type Definition (DTD) files, which are used to define a content model for a cXML document. DTDs include the specifications for the allowed elements, their order, and attributes data types.

- **Do you currently have any transactive, XML-enabled Web-based e-commerce applications?**

Technical developers must have a fundamental understanding of how to create, receive, transmit, parse, and query XML data to and from a remote source. An XML parser is the basic tool that developers use to process XML messages. Free tools are available to familiarize the technical team with XML and cXML.

- **Do you currently have a catalog index file?**

A catalog index file is a cXML or CIF document you create that the customer loads into Ariba Buyer to create entries in the catalog hierarchy. The file defines how you and your products appear in the Ariba Buyer catalog user interface. The catalog entry for PunchOut items is similar to static items, except that it contains a clickable URL link to your PunchOut site instead of a unit price. When users select a PunchOut item, Ariba Buyer formats and sends a cXML PunchOutSetupRequest document to Ariba Network, which initiates the PunchOut session.

## Design

During the design phase, you analyze the overall design of your PunchOut site and decide how to integrate it with your order processing system. CIF, cXML, or PunchOut catalog formats might be better for certain commodities and customer business rules, so discuss their use with your customers to understand their requirements. You must analyze the following areas:

- **Supplier Requirements**
- **Customer Requirements**
- **Learning About cXML and PunchOut**

### Supplier Requirements

The PunchOut protocol is flexible enough to allow you to differentiate your PunchOut site from those from other suppliers. Several factors, such as schedule, budget, and desired appearance will influence how you approach these supplier specifics and you should review all of them as part of your design process.

### Branding

Branding is the process you use to make your PunchOut site unique. You can personalize the appearance and functionality of the site based on customer authentication.

You are encouraged to personalize your PunchOut site and leverage the look and feel of your existing websites. However, PunchOut sites do not require typical e-commerce site functions, such as input fields for manual login, links to exit the current shopping session, or access to external sites.

### Publishing the Index Catalog

PunchOut index catalogs are published on Ariba Network like other catalog types. You can make the catalog public (available to all customers) or private (available only to specific customers). Customers select catalogs based on their description and on details about the supplier. Publishing a PunchOut index catalog on Ariba Network is the quickest way to let customers know that you have a PunchOut site available.

### Customer Requirements

Meet with your customers to determine their business and technical requirements. Ideally, you collect this information from each customer that wants to use your PunchOut site.

### Business Requirements

Talk with your customers about which products to make available on your PunchOut site. For the best user experience, you should understand the high-level business requirements of your customers.

- Analyze your customers' current and future procurement practices.
- Determine your customers' content-specific requirements by commodity.
- Identify any reporting considerations or requirements.
- Assign ownership across your team to resolve open issues.

## Technical Requirements

Find out your customers' technical requirements for product content and transactions. Develop the processes for addressing the issues that arise when two organizations enter into a trading relationship. Some of the issues that should be discussed are:

- Content and pricing, including national versus regional contracts
- Commodity codes and Unit of Measure (UOM) encoding
- International issues, such as multi-language and currency
- Freight, shipping methods, and taxes
- How to initiate payment, such as PCard or invoice
- Issuing credits and returns
- Non-catalog (ad-hoc) purchase orders
- Changed and cancelled orders
- Additional information required for documents and cXML requests, such as cost center, department, requester, and supplier account code
- How to handle conflicts with existing sales channels, such as distributors
- Updating order status on Ariba Network

## Security

Your PunchOut site must communicate through HTTPS (Hyper Text Transfer Protocol Secure). For more information, see “[HTTPS Connections](#)” on page 26. HTTPS protects all parties in PunchOut sessions: your customer, Ariba Network, and your PunchOut site.

## Learning About cXML and PunchOut

You will need to document the transaction process flow into and out of your PunchOut site and identify which messages need to be coded. Ariba has documentation available to assist in defining the process. The technical developer should read the following guides, available on Ariba Network.

Guide	Describes
<i>Configuring Document Routing</i>	How to set up advanced configuration, including cXML configuration
<i>Ariba Network Catalog Administration Guide for Suppliers</i>	How to upload, test, and publish static and PunchOut catalogs
<i>Ariba Catalog Format Reference</i>	Catalog features and the syntax of CIF and cXML catalogs

## Development

Development involves the implementation of each step in the PunchOut process. This process can be described through message flow.



## Message Flow

A PunchOut session is comprised of cXML messages that pass between Ariba Buyer, Ariba Network, and your PunchOut site:

### 1 User Login

A user at a buying organization first logs in to Ariba Buyer and creates a requisition. This step is important, because it means the user has been authenticated by the buying organization. During PunchOut, Ariba Network authenticates the buying organization, not the user.

### 2 PunchOut Site Selection

Next, the user searches for products and services in the procurement application and selects your PunchOut item. Depending on what you include in your PunchOut index catalog, users experience store-, aisle-, shelf- or product-level PunchOut. If you offer aisle-, shelf-, or product-level PunchOut, the user punches out to a page on your site that describes the aisle, shelf, or product. If you offer store-level PunchOut, the user punches out to see all your products by selecting your company name. Store-level PunchOut usually requires your site to have a search mechanism so users can find items they need.

For more information, see “[punchoutLevel Attribute](#)” on page 78.

### 3 PunchOutSetupRequest

Ariba Buyer generates a cXML PunchOutSetupRequest document and sends it through an HTTP Post to Ariba Network. Ariba Network authenticates it and forwards it through an HTTP Post to your PunchOut site.

When the buying organization registers on Ariba Network, it configures a SharedSecret. PunchOutSetupRequest documents sent to Ariba Network identify the customer based on the Identity element in the From element and populate the Credential domain with the customer's NetworkID. Each buying organization has its own NetworkID.

When Ariba Network determines who the request is from and who it is to, it deletes the customer's shared secret and uses the one from your Ariba Network account. This shared secret allows the PunchOutSetupRequest to effectively log in to your PunchOut site. You never see your customer's SharedSecret and do not have to maintain a separate password/login for each user or customer. The end user can be identified in Contact and Extrinsic elements.

In addition to the authentication and identification parameters, the PunchOutSetupRequest document contains a BuyerCookie. The BuyerCookie changes between concurrent PunchOut sessions, thereby allowing you to track which screen a particular user is on during the shopping process. An edit operation on an existing order usually results in a new buyer cookie for that particular session. To be more specific, Ariba Buyer guarantees that the buyer cookie is unique among all values used by simultaneously initiated PunchOut sessions. The value might, for example, correspond to a session identifier in that application. To link a specific order to a user, use the SupplierPartAuxiliaryID (supplier cookie) element.

### 4 PunchOut authentication

When your PunchOut site receives the PunchOutSetupRequest document, it performs the following tasks:

- Authenticates Ariba Network based on the Sender and SharedSecret
- Verifies the From identification

You can now initiate a session because the user's organization is a certified Ariba Network member. Your PunchOut site can generate a shopping page for the PunchOut session.

### 5 PunchOutSetupResponse

Your PunchOut site redirects the user. It issues a `PunchOutSetupResponse` document to Ariba Network with your `StartPage URL`, which is the shopping page of your PunchOut site. Ariba Network forwards the `PunchOutSetupResponse` to Ariba Buyer.

## 6 Shopping

Ariba Buyer opens your PunchOut site in a new window using the `StartPage URL` you supplied. The user selects and configures products or services. Selecting an item adds it to a shopping cart or basket on your site.

## 7 PunchOutOrderMessage

When done selecting items on your PunchOut site, the user clicks a **Transfer Basket to Ariba Buyer** link. Your site issues a `PunchOutOrderMessage` document to Ariba Buyer (in an HTML hidden form field) that lists the contents of the user's shopping cart.

The window displaying your PunchOut site disappears and the description of the PunchOut items appears in the user's requisition. This information acts as a quote, not an actual order. When the quote is approved in Ariba Buyer, it generates a purchase order.

To alleviate user confusion, your checkout process should use the following sequence of buttons:

1 Add item to basket

2 Transfer basket to Ariba Buyer

The checkout process should not require the user to enter credit card information or ship-to address details. This data is maintained in Ariba Buyer.

In Ariba Buyer 6.1 and earlier, because your site appears in a new browser window, there is a final screen that the user sees containing the requisition number and a **Close Browser** button. This returns the user to Ariba Buyer.

To allow users to return to the PunchOut site and make changes to it, the `PunchOutOrderMessage` document should have the `operationAllowed="edit"` attribute.

## 8 Requisition Approval

Ariba Buyer submits the requisition for approval within the buying organization. It does not update you on the progress of the requisition until after it has received all required approvals and has been turned into a purchase order.

If managers in the approval chain deny a requisition, they can use PunchOut to go to your site to remove line items or delete the requisition. You should reach an agreement with customers about how canceled requisitions should be handled.

## 9 OrderRequest

Upon approval of the requisition, Ariba Buyer generates an `OrderRequest` document and transmits it to you through Ariba Network. This document contains the purchase order details required for processing. For more information, see Chapter 7, "[Purchase Orders](#)."

## Extrinsics and Supplier Cookies

Ariba Buyer uses extrinsic data to further identify a user to you. The standard extrinsics sent from Ariba Buyer in `PunchOutSetupRequest` documents are `User` and `CostCenter`. Customers determine the naming and population of all extrinsic elements, so indicate any additional data you need. Reliance on extrinsic data is discouraged, because it makes using your PunchOut site with other customers more difficult.

The `SupplierPartAuxiliaryID` element, or “supplier cookie,” allows you to transmit additional data, such as a quote number from your PunchOut site to the purchase order. Ariba Buyer passes it back to you in any subsequent `PunchOutSetupRequest` “edit” or “inspect” sessions, and any resulting `OrderRequest` document. Suppliers often use the cookie to associate items in a requisition with the corresponding items in a PunchOut session shopping cart.

## Testing

The PunchOut testing phase ensures that your site is configured properly and that it effectively communicates with Ariba Buyer.

### Self-Testing on Ariba Network

Your Ariba Network account has a parallel account called a *test account*. Test accounts have a built-in catalog tester that allows you to check static catalogs, PunchOut sites, and order routing. Use the catalog tester to test your PunchOut site and to send simple purchase orders to yourself. It has a single-step feature, allowing you to edit each PunchOut document before transmission.

**Important:** The catalog tester is useful for debugging your PunchOut site and demonstrating the site to potential customers. For information about using your test account and the catalog tester, see the *Ariba Network Using Test Accounts* topic in the Product Documentation > For Administrators section of the Learning Center.

### Testing with Customers

The final phase of testing is to exercise your PunchOut site with your customers. You should confirm that your customers have enabled their Ariba Network test accounts and have added your test account as a supplier. Then, you can publish your index catalog to them to confirm that it meets their criteria, allow them to exercise your PunchOut site, receive test orders, and you can observe those orders in your order-entry system.

Specific scenarios to run through when testing with customers are suggested below.

### Authentication

Your PunchOut site must perform authentication through the domain, buyer identity, and shared secret. You cannot deploy it if it performs authentication any other way, for example with a user ID or with a user-entered password.

### Basic Tests from Ariba Buyer

The following scenarios comprise a basic test of your PunchOut site.

- Testing the PunchOut item

Create a requisition in Ariba Buyer. Select your PunchOut item from the catalog hierarchy.

**Expected behavior:** Ariba Buyer connects to your PunchOut site. The user can shop, place items in a cart, and return the cart to the Ariba Buyer requisition.

- Simulating a lost connection

Create a requisition in Ariba Buyer. Select a PunchOut item. Ariba Buyer displays your PunchOut site. Close the PunchOut site before “checking out.”

**Expected behavior:** The user session returns to Ariba Buyer and displays the Ariba Buyer first page.

Select the PunchOut item again.

**Expected behavior:** Return to your PunchOut site, where the shopping cart is empty.

- Testing multiple line items on a requisition

Create a requisition with two line items from the same PunchOut site shopping cart. After the cart returns to the requisition, select one item to initiate the edit functionality.

**Expected behavior:** Both items appear in the cart upon return to your PunchOut site.

Remove both items from your PunchOut site’s cart and check out again.

**Expected behavior:** The requisition must not contain any of the items selected during the PunchOut session.

- Testing a non-catalog purchase

You must agree with your customer how to handle and route non-catalog (ad hoc) purchases. Create a requisition with non-catalog items and see how it appears in your order entry system.

- Testing the basic functionality of service items

If your PunchOut site lists services, test it like a user would:

- 1 Go to your PunchOut site and provide configuration data to purchase a service. The PunchOut site returns a line item.
- 2 Use a PunchOut edit session to bring back a line item with pricing. After generating a purchase order, try to PunchOut with both edit and inspect.

- Testing contract pricing

If your pricing is determined by the buying organization’s ID, make sure the correct price is displayed.

- 1 Go to your PunchOut site and provide configuration data. You receive a line item back including contract pricing.
- 2 Submit the requisition and initiate the workflow and approval process.
- 3 After the requisition is approved, use an edit PunchOut session to select the product. You can also use an inspect PunchOut session.

## Deployment

For deployment, move the tested site to production. You should coordinate this action with your customers.

## Configuration Management

To make the build-and-deploy process manageable and repeatable, it is recommended that you use a configuration management system. You can script the process of deployment, including creating new websites, moving files, and generating customer-specific content and pricing.

### Customer Service

Confirm that your Customer Service organization is ready to support the new PunchOut site and any new policies and procedures.

### Sanity Check

Immediately after going live, have the buying organization perform a few test PunchOut sessions and create several test orders to validate connectivity in the production environment. Finally, closely monitor your Ariba Network account to ensure that purchase orders route properly.

## Becoming Ariba Ready Certified

*Ariba Ready* is an Ariba Network service that certifies your organization has followed the guidelines for Ariba Buyer and Ariba Network integration. When customers look for new suppliers, they see which ones are Ariba Ready, and know they can integrate with you easily.

After you have enabled and tested your PunchOut site, you should apply for Ariba Ready certification. The Ariba Ready team places your PunchOut site in a queue to be tested through script testing. When the site passes the scripts, you receive an Ariba Ready logo to notify potential customers that the site has met Ariba's PunchOut requirements, expediting the addition of new customers.

For more information, see [www.ariba.com/suppliers/supplier\\_validation.cfm](http://www.ariba.com/suppliers/supplier_validation.cfm).

## Retrofitting an Existing Website

---

You might have an existing website that you want to modify to support PunchOut. For business or technical reasons, it might make sense to use an existing e-commerce site.

### Leveraging an Existing Site

When users interact with a typical B2B site, they log in directly, search for items, configure commodities, select shipping options, enter credit card information, and place orders. However, these sites are really B2C applications: they treat users as consumers.

B2B sites, however, have dynamic workflow, approvals, saved shopping carts, and contract pricing. Your customer should calculate payment and shipping information. Existing processes, such as dynamic workflow or collection of shipping and payment information, will not be needed for PunchOut.

The model to adopt when creating a PunchOut site is *quote* and *purchase*. Users perform PunchOut for product selection, configuration, and to obtain a quote. Ariba Buyer generates the purchase order separately. The process flow is as follows:

- 1 A requisitioner in Ariba Buyer uses PunchOut to go to your PunchOut site and select products.
- 2 At checkout, the PunchOut site brings back the fully configured items to Ariba Buyer.
- 3 In Ariba Buyer, the user selects logistic information, including bill-to, ship-to, shipping method, and need-by date.

- 4 The user submits the requisition for approval. Parties in the buying organization can inspect, edit, approve, or deny the requisition depending on each approver's role and their permissions.
- 5 When the requisition is fully approved, Ariba Buyer submits a purchase order, with shipping, billing, and need-by date.

## Quick Retrofitting

Use the following steps to modify an e-commerce site to support PunchOut as rapidly as possible:

- 1 Remove all non-configuration related processes.

Ariba Buyer authenticates users and Ariba Network authenticates buying organizations.

- 2 Adapt a model where you provide a quote and receive a purchase order.

If your current site does not support the purchase order model, leverage the existing code base and build a new site with new processes.

- 3 Remove or deactivate payment, shipping, and workflow.

Ariba Buyer handles payment, shipping, and workflow according to the customers' specific business processes.

- 4 Clean up the user interface.

Remove all links to outside websites. A PunchOut user should not be able to exit the PunchOut site through site navigation.

## PunchOut for Services

---

Your PunchOut site can offer permanent or temporary services. Developing a PunchOut site for services requires a good understanding of the PunchOut process, and possibly, coordinating services procurement with your customers.

Procuring services through Ariba Buyer is very different from procuring commodities. For commodities, the approval workflow and access control lists are in Ariba Buyer; there is no preliminary product configuration required and offerings do not dynamically change based on market conditions. Any commodity that does not follow the above principles lends itself to PunchOut, where the catalog is maintained at your site.

## Services Exchange

You can create a PunchOut site that supplies services or contract work. This is called a *services exchange*, because requisitioners request services and configure settings that describe the work they need done. Users follow two steps to procure services from services exchanges:

- 1 Open a position
- 2 Engage a candidate to fulfill the opened position

You can model these steps with two PunchOut operations. The first step is a "create" session that partially populates a requisition line item. The second step is an "edit" session that adds more detail to form a complete line item. The create session opens a contract worker position; the edit session engages a candidate to fill the opened position.

## Create Session

The create session defines a contract worker. This session might be subject to an approval flow in Ariba Buyer if it returns estimates for requested services.

Your PunchOut site can temporarily stop the workflow by withholding a key field, such as Unit of Measure (UOM). It can send an email notification to the user to return to the site and continue with the process. These messages cannot automatically trigger an event in Ariba Buyer.

If opening a position requires pre-approval in Ariba Buyer, then you need to arrange a custom double approval chain for the service-specific commodity. The buying organization must send you a notification of approval or denial of a request through email notifications.

## Edit Sessions

The edit session further describes the service. The user enters missing data (such as the UOM), making the requisition complete and ready for the start of the approval process in Ariba Buyer.

After the approval process is complete, Ariba Buyer sends the purchase order to your services exchange. The contract worker starts on the negotiated start date and enters time worked into time sheets. The accounting system linked to Ariba Buyer processes the invoices and pays against them accordingly.

## Writing a PunchOut Deployment Guide

---

After implementing a PunchOut site, provide Ariba Buyer implementors with a *PunchOut Deployment Guide* that explains your policies, capabilities, and processes. This guide will help your customers integrate with your site and adhere to your business policies. The availability of this information will make the rollout to subsequent customers easier.

Your *PunchOut Deployment Guide* should contain the following sections:

- **Connectivity Overview**
- **Authentication and Identification**
- **Required Extrinsic**s
- **Content Requirements/Specification**
- **Address Information**
- **Accounting Structure**
- **Commodity Codes**
- **Transactions Supported**

Each of these sections is described below.

### Connectivity Overview

Describe the PunchOut process flow and the integration to Ariba Buyer. You can copy the explanation from the PunchOut Event Sequence section of the *cXML User's Guide*.

Include any application-specific processes in the integration. For example, document RFQ (Request For Quote) or service requisitioning where a second PunchOut might be required to receive the pricing of selected services.

## Authentication and Identification

Explain how you perform authentication of PunchOut sessions.

Include any additional information used to identify the user, such as `PunchOutSetupRequest` extrinsics, or `Contact` or `Address` elements. Describe your PunchOut site's ability to use this information to present custom content.

## Required Extrinsics

Indicate whether your site requires extrinsic information to initiate customized PunchOut sessions. Keep the use of extrinsic elements to a minimum, because they increase implementation lead time.

Different versions of Ariba Buyer use different default extrinsic elements:

- Ariba Buyer 6.1 and earlier (cXML 1.0) use `User` and `CostCenter` elements. However, customers might name these extrinsics differently, so find out the exact names.
- Ariba Buyer 7.0 and later (cXML 1.1) use the `Contact` element, obsoleting the extrinsic elements `User` and `CostCenter`.

In purchase orders, extrinsics are often used to send additional information from the customer at the line item level, such as `Company Code` and `Contract Number`. Describe any line item extrinsics you accept.

## Content Requirements/Specification

Describe your content specification process and capabilities. Describe your process for selecting the categories of products shown to users. Describe whether you have the capability to limit access to items to certain users in the organization.

If you display both contract and non-contract items, describe how these are shown and who can see them. For instance, you might allow typical users to see only contract items, but allow purchasing agents to see all your items. In this case, describe how your PunchOut site determines users' roles.

Provide your PunchOut index catalog (either CIF or cXML) to customers so the appropriate links appear in Ariba Buyer. Also, if you support the `SelectedItem` attribute (available in cXML 1.1), describe how you use it.

## Address Information

It is important to discuss address format so Ariba Buyer can integrate with your order processing application.

Some customers use the `addressID` attribute of the `Address` element to identify a predefined ship-to or bill-to address. If you use this attribute, describe the process of loading and maintaining your customer's address data. Also, discuss how you handle exceptions, such as the user drop-shipping the delivery to a location not already in Ariba Buyer, or the user adding a new address not known to you. In the first case, the `addressID` is null, in the second case the `addressID` might be a number you do not have.

If you do not use the `addressID` attribute, tell customers what to send you in their address elements. This data includes the `DeliverTo` elements, which are often the most problematic, because some customers implement them differently. Most commonly, they contain a person's name and their building, floor, or mailstop.



Typically, there are two occurrences of that element in the order, but some customers send only one. Collect the information your customer plans to send with these before implementation so you can map them into your system.

Similarly, you must also note the format and content of the **Street** elements. Determine the data format your customers put in these elements before implementation. The format varies on a number of factors, including the ERP system used. The following table is useful for capturing this information for discussion.

Element	#	Customer		Supplier	
		Description	Max Length	Description	Max Length
<b>DeliverTo</b>	<b>1</b>				
<b>DeliverTo</b>	<b>2</b>				
<b>Street</b>	<b>1</b>				
<b>Street</b>	<b>2</b>				
<b>Street</b>	<b>3</b>				
<b>Street</b>	<b>4</b>				

## Accounting Structure

Customers have different formats for accounting information they send in purchase orders and receive in invoices. This difference is due to a number of factors, including the ERP system used and the customer's general ledger design.

Collect this information during implementation and describe your ability to capture and return this information for invoicing and Peard reconciliation. You can use the following table for collecting this information from your customers:

Parameter	Customer Data			Return on Invoice?
	Segment Type	Format / Length	Description	
<b>Reporting Center</b>	Cost Center	Alpha / 5	ID	Y
<b>Reporting Account</b>	Account	Integer / 6	Account	N
<b>Reporting Business</b>	Company	Alpha / 4	ID	N

## Commodity Codes

Describe the commodity code standard you support for PunchOut and the level of granularity of the data that you return. Include an appendix that lists all the distinct commodity codes you send, so customers can perform any required mapping. Ask your customers to specify commodity codes that they use to control workflow or approvals.

The UNSPSC (United Nations Standard Product and Service Code) standard is preferred. For more information, see “**Commodity Codes**” on page 241.

## Transactions Supported

Describe the electronic transactions that you support, the method you use to process orders, how you handle orders not supported electronically, and how you process exceptions. The following are specific order types you might need to describe:

### Change/Cancel Orders

Ariba Network provides a separate method for routing change and cancel orders. Describe how you route and process these orders. For example, you might route change/cancel orders directly to your shipping department.

For more information, see “[Cancel Orders and Change Orders](#)” on page 128.

### Non-Catalog (Ad-Hoc) Items

Non-catalog orders sent to you come with a `SupplierPartID` of “Not Available” or with an `isAdHoc` flag. Describe whether you process these line items.

For more information, see “[Non-Catalog Items](#)” on page 133.

### Orders Containing Static and PunchOut Items

If you have both static catalogs and a PunchOut catalog, you might receive purchase orders that contain both types of items. Your order-receiving system needs to be able to process them.

One difference between these two line item types is the content of the `SupplierPartAuxiliaryID` element. For PunchOut line items, you receive the value your PunchOut site generates. For static catalog orders, you receive the value from the static catalog, or no data for catalogs without this information.

### Copied Requisitions

Ariba Buyer allows users to make copies of their requisitions for repeat orders. When it copies requisitions, it copies the contents of entire line items, except for the `SupplierPartAuxiliaryID` element. This exception preserves the integrity of your PunchOut site, because that element is in your control. Because the `supplierPartID`, quantity, and price are all in the copied line item, you should be able to process it. Describe in this section whether you will process purchase orders generated from copied requisitions.

### Changes in Price

Prices can change between the time the PunchOut session occurs and the time the purchase order is sent. Describe how you process these exceptions.

### Third-Party Suppliers

If you are a product aggregator, your PunchOut site lists other suppliers’ products, and you route purchase orders you receive to those suppliers. Use the third-party suppliers’ NetworkIDs or D-U-N-S numbers to populate the `SupplierID` element at the line item level.

These IDs must be registered on Ariba Network and in Ariba Buyer. Before sending a purchase order to Ariba Network, Ariba Buyer determines which supplier the OrderRequest is for and populates the To element in the header with the ANID or D-U-N-S number of that supplier (extracted from the NetworkID or D-U-N-S number at the item or requisition level).

In your *PunchOut Deployment Guide*, list the third-party suppliers and their IDs.

### **Quotes Split into Multiple Orders**

Customers that pay with PCards or assign your products to multiple general ledger accounts might split quotes among several orders. While customers always approve or reject the entire quote as a unit, this split might cause exceptions in your order entry system.

Describe whether this situation will cause exceptions. If you cannot process these orders, describe the process for these exceptions.

**Note:** The frequency of these situations depends on the products you offer, how they are classified, and your customers' accounting processes.



---

## Chapter 6 PunchOut Transaction

- “PunchOut Index Catalog” on page 77
- “PunchOutSetupRequest Document” on page 79
- “PunchOutSetupResponse Document” on page 84
- “PunchOutOrderMessage Document” on page 85
- “PunchOut URL” on page 92
- “URL vs. SelectedItem Elements” on page 94
- “Level 2 PunchOut” on page 94
- “PunchOut Session Timeout” on page 103
- “ASP Examples” on page 103

This chapter discusses how to set up PunchOut sites that integrate with Ariba Network and procurement applications.

### PunchOut Index Catalog

---

A PunchOut index catalog is a file created by the supplier that the buying organization loads into their procurement application to create a PunchOut link in the catalog hierarchy. The file defines how the PunchOut catalog or the PunchOut items appear in the procurement application’s catalog. Ariba Buyer and Procure-to-Pay accept both CIF and cXML PunchOut index catalogs.

### SupplierID Element

This element identifies the supplier that hosts the PunchOut catalog. Suppliers use NetworkID or Dun & Bradstreet D-U-N-S® numbers to identify themselves.

- Your NetworkID appears on the Home page of your Ariba Network account
- You can find information on D-U-N-S numbers at [www.dnb.com](http://www.dnb.com)

### URL Element

This element must exist, but it is no longer used by Ariba Network.

In previous releases of Ariba Network, this URL identified the supplier’s PunchOut website. Now, the URL comes from the supplier’s cXML profile or from the supplier’s Ariba Network account. For more information, see “PunchOut URL” on page 92.

Ariba Buyer passes this value in the PunchOutSetupRequest document, so the supplier can use it to further identify the PunchOut item.

## Classification Element

PunchOut items in the procurement application must be classified to map to the catalog hierarchy. The recommended commodity classification system is UNSPSC (United Nations Standard Products and Services Code). For more information, see “[Commodity Codes](#)” on page 241.

## punchoutLevel Attribute

Use the optional `punchoutLevel` attribute to tell Procure-to-Pay where to place your offerings in the catalog displayed to end-users. This attribute is available only for Procure-to-Pay (cXML 1.2.016 or later). It is available for both cXML and CIF PunchOut index catalogs.

The `punchoutLevel` attribute can have the following values:

Value	Use if your PunchOut site has...	PunchOut Item Example
store	A search page that offers all of your products or services. Procure-to-Pay users punch out directly without displaying item details. These items display at the top of categories, above products.	ACME Hardware Store
aisle	Pages that group your items into a small number of categories. Procure-to-Pay users punch out directly without displaying item details.	Cutting Tools
shelf	Pages for similar products from which customers would choose when shopping. Procure-to-Pay displays item details and allows users to punch out.	Handsaws
product	A page for each of your offered items or SKUs. Procure-to-Pay displays item details and allow users to punch out.	Cuts-a-Lot 14-inch Handsaw

You can provide a mixture of items that have different `punchoutLevel` values. For example, you might offer users one store-level PunchOut item and a product-level PunchOut item for each of your 50 most popular products. If you do not include this attribute, Procure-to-Pay considers the item to be a store-level item.

Your PunchOut site can interpret the `SelectedItem` element in the `PunchOutSetupRequest` document to determine which aisle, shelf, or product users select.

Shelf-level and product-level PunchOut are especially useful if you implement Level 2 PunchOut. For more information, see “[Level 2 PunchOut](#)” on page 94.

## Example PunchOut Index File

The following example shows a PunchOut index file in cXML format. To keep the file size low, use cXML instead of CIF for Level 2 index files.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Index SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.016/cXML.dtd">
<Index>
  <SupplierID domain="DUNS">1234567</SupplierID>
  <Comments xml:lang="en-US">Sample cXML Index Catalog</Comments>
  <IndexItem>
    <IndexItemPunchout>
      <ItemID>
        <SupplierPartID>123456</SupplierPartID>
      </ItemID>
```

```

    <PunchoutDetail punchoutLevel="store">
      <Description xml:lang="en-US">Workchairs, Inc.</Description>
      <URL>http://www.workchairs.com</URL>
      <Classification domain="UNSPSC">88888889</Classification>
    </PunchoutDetail>
  </IndexItemPunchout>
</IndexItem>
</Index>

```

## PunchOutSetupRequest Document

To initiate PunchOut sessions, requisitioners select PunchOut items in their procurement application. The procurement application generates a cXML PunchOutSetupRequest document and sends it to Ariba Network, which forwards it to the PunchOut site. The PunchOutSetupRequest document authenticates the buyer for the supplier.

This section discusses:

- [Credential Elements](#)
- [UserAgent Element](#)
- [PunchOutSetupRequest operation Attribute](#)
- [BuyerCookie Element](#)
- [Extrinsic Element](#)
- [BrowserFormPost Element](#)
- [SupplierSetup Element](#)
- [SelectedItem Element](#)
- [Example PunchOutSetupRequest Document](#)

## Credential Elements

Like other cXML documents, PunchOutSetupRequest documents contain From, To, and Sender credentials.

### From Credential

This element identifies the originator of the PunchOutSetupRequest (the buying organization). For example:

```

<From>
  <Credential domain="NetworkID">
    <Identity>AN01000002792</Identity>
  </Credential>
</From>

```

### To Credential

This element identifies the supplier (the destination of the PunchOutSetupRequest.) For example:

```

<To>
  <Credential domain="DUNS">
    <Identity>942888711</Identity>
  </Credential>
</To>

```

For more information about credential limitations of older versions of Ariba Buyer, see “[NetworkID and Older Versions of Ariba Buyer](#)” on page 35.

### Sender Credential

When a procurement application creates the PunchOutSetupRequest document, the Sender credential specifies the identity and shared secret of the buying organization. When Ariba Network forwards the document to the supplier, it changes the Sender credential to specify the identity of Ariba Network and uses the supplier’s SharedSecret.

This example shows a PunchOutSetupRequest that has passed through Ariba Network.

```
<Sender>
  <Credential domain="AribaNetworkUserId">
    <Identity>sysadmin@ariba.com</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Ariba Buyer 7.1</UserAgent>
</Sender>
```

**Note:** If the PunchOutSetupRequest document comes from Ariba procurement solutions, the Sender credentials is NetworkID.

### UserAgent Element

This element identifies the originating application. It consists of the software company name, product name and version. Version details can appear in parentheses.

Example:

```
<UserAgent>Ariba Buyer 8.2</UserAgent>
```

### PunchOutSetupRequest operation Attribute

The operation attribute specifies the type of PunchOut session to initiate: create, edit, or inspect:

operation	Description
create	Punch out to add a new PunchOut item to a requisition
edit	Punch out to modify an existing PunchOut item
inspect	Punch out to inspect (read-only) an existing PunchOut item

For example:

```
<PunchOutSetupRequest operation="create">
```

Suppliers can restrict the editing of later PunchOut sessions by using the operationAllowed attribute in the PunchOutOrderMessage document. For more information, see “[operationAllowed Attribute](#)” on page 85.



## BuyerCookie Element

The BuyerCookie element transmits information that is opaque to the PunchOut site, but must be returned to the originator for all subsequent PunchOut operations. It enables the procurement application to match multiple, outstanding PunchOut requests. It is unique per PunchOut session.

**Note:** When this element passes through Ariba Network from Ariba Buyer 7.0 (cXML 1.1), the supplier actually receives a session ID number, not the alphanumeric value in the PunchOutSetupRequest from the procurement application to Ariba Network.

Example:

```
<BuyerCookie>1TF5JRP11S3W9</BuyerCookie>
```

## Extrinsic Element

This optional element is used for any additional data that the requestor wants to pass to the PunchOut site. In cXML 1.0, the extrinsics User and CostCenter elements often provided contact information. In cXML 1.1, the Contact element obsoletes the User Extrinsic and a few others commonly included in cXML 1.0 documents.

Ariba Buyer 6.1 and 7.0 include a different set of extrinsics. By default, Ariba Buyer 6.1, includes the extrinsics User and CostCenter. By default, Ariba Buyer 7.0 includes the extrinsics UniqueName, UserEmail, and CostCenter. Any additional extrinsics would have to be agreed upon between the buying organization and the supplier.

The following example passes the department of the user initiating the PunchOut operation.

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

<Extrinsic name="UniqueName">jsmith</Extrinsic>
```

In cXML 1.1 or later, use the Contact element in the body of the request to uniquely identify the user (for example, John Smith in Finance at acme.com) instead of using extrinsic elements. The buying organization can configure their procurement application to insert Contact and extrinsic data.

Example:

```
<Contact role="endUser">
  <Name xml:lang="en-US">Mr. Burns</Name>
  <Email>mburns@springfield.com</Email>
  <Phone name="Office">
    ...
  </Phone>
</Contact>
```

## BrowserFormPost Element

This element contains the URL where the PunchOut site returns the PunchOutOrderMessage at the end of the PunchOut session.

Example:

```
<BrowserFormPost>  
  <URL>http://procure.bigbuyer.com:3377/punchout</URL>  
</BrowserFormPost>
```

## SupplierSetup Element

In early cXML releases, the SupplierSetup element provided the only way to specify a PunchOut site's URL. However, beginning with cXML 1.1, suppliers configure Ariba Network with the URLs of their PunchOut sites and use the new SelectedItem element to specify store-, aisle-, shelf- or product-level PunchOut.

The SupplierSetup element has been deprecated. However, PunchOut sites must accept it until all Ariba Buyer installations recognize and send the SelectedItem element. For more information, see “**PunchOut URL**” on page 92.

## SelectedItem Element

This element contains the item selected by the user from a catalog and identifies the supplier's part number. Starting with cXML 1.1, procurement applications use the SelectedItem element to specify store-, aisle-, shelf, or product-level PunchOut.

Example:

```
<SelectedItem>  
  <ItemID>  
    <SupplierPartID>55555</SupplierPartID>  
  </ItemID>  
</SelectedItem>
```

## Example PunchOutSetupRequest Document

The following example shows a PunchOutSetupRequest document as it travels from Ariba Network to the supplier's PunchOut site. It contains the supplier's shared secret.

```
<?xml version="1.0"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="958075346970@www.bigbuyer.com " timestamp="2005-06-14T12:57:09-07:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN01000002792</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="DUNS">
        <Identity>12345678</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>sysadmin@ariba.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ariba Buyer 8.2</UserAgent>
    </Sender>
  </Header>
  <Request>
    <PunchOutSetupRequest operation="create">
      <BuyerCookie>1J3YVWU9QWMTB</BuyerCookie>
      <Extrinsic name="CostCenter">610</Extrinsic>
      <Extrinsic name="User">jpicard</Extrinsic>
      <BrowserFormPost>
        <URL>http://bigbuyer.com:3377/punchout</URL>
      </BrowserFormPost>
      <SupplierSetup>
        <URL>https://punchout.workchairs.com/PunchOutServlet</URL>
      </SupplierSetup>
      <ShipTo>
        <Address addressID="001">
          <Name xml:lang="en">BigBuyer Headquarters</Name>
          <PostalAddress>
            <DeliverTo>Jean Picard</DeliverTo>
            <Street>1565 Pine, MS A.2</Street>
            <City>New York</City>
            <State>NY</State>
            <PostalCode>01043</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
        </Address>
      </ShipTo>
      <Contact>
        <Name>jpicard</Name>
      </Contact>
      <SelectedItem>
        <ItemID>
          <SupplierPartID>54543</SupplierPartID>
        </ItemID>
      </SelectedItem>
    </PunchOutSetupRequest>
  </Request>
</cXML>
```

## PunchOutSetupResponse Document

---

After suppliers receive PunchOutSetupRequest documents, they return PunchOutSetupResponse documents to Ariba Network through the same HTTPS connection as the PunchOutSetupRequest.

This section discusses:

- [Overview of the PunchOutSetupResponse Documents](#)
- [Status Element](#)
- [StartPage URL Element](#)
- [Example PunchOutSetupResponse Document](#)

### Overview of the PunchOutSetupResponse Documents

The PunchOutSetupResponse document serves two functions:

- It indicates whether PunchOut initiation was successful
- It provides the procurement application with a redirect URL to the supplier's website start page

Ariba Network has a 30-second timeout for receiving PunchOutSetupResponse documents. If suppliers do not send a response within that time limit, the PunchOut session initiation fails.

### Status Element

This element conveys the success or failure of a request operation. It is comprised of a code attribute and a text attribute, and an optional `xml:lang` attribute.

The code attribute follows the HTTP status code model. In general, a 2xx series code indicates a successful client-server communication, a 4xx series code indicates a client error status code, and a 5xx series code indicates a server error code. The text attribute and optional `xml:lang` attribute allow for a text description of the status returned in a response. Suppliers should put the XML parsing or application error in the body of the Status element.

### StartPage URL Element

The PunchOutSetupResponse document contains a URL element that specifies the start page URL to pass to the user's browser for the interactive browser session.

## Example PunchOutSetupResponse Document

The following is a sample PunchOutSetupResponse document.

```
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="958074700772@www.workchairs.com" timestamp="2005-06-14T12:59:09-07:00">
  <Response>
    <Status code="200" text="success"/>
    <PunchOutSetupResponse>
      <StartPage>
        <URL>https://punchout.workchairs.com/Servlet/sessionid=7006</URL>
      </StartPage>
    </PunchOutSetupResponse>
  </Response>
</cXML>
```

## PunchOutOrderMessage Document

The PunchOutOrderMessage document sends the contents of the PunchOut site's shopping basket to the procurement application user's requisition. It provides product details and prices to procurement applications. Suppliers can also send supplier cookies to associate items with a specific shopping session.

It provides a quote for the requested items—but suppliers have not yet received a purchase order, so the order cannot yet be booked.

This section discusses:

- [operationAllowed Attribute](#)
- [cxml-base64 and cxml-urlencoded](#)
- [BuyerCookie Element](#)
- [Total Element](#)
- [ItemIn Element](#)
- [ItemID Element](#)
- [ItemDetail Element](#)
- [Example PunchOutOrderMessage Document](#)

### operationAllowed Attribute

This attribute controls whether procurement applications can initiate a later PunchOut session containing data from this PunchOutOrderMessage document. This attribute can be assigned a value of create, inspect, or edit.

If operationAllowed="create", procurement applications can generate only an OrderRequest document. Otherwise, the procurement application can inspect or edit the shopping cart, initiating subsequent PunchOutSetupRequest documents with the appropriate operations and the ItemOut elements corresponding to the ItemIn list returned in a PunchOutOrderMessage document. Support for edit implies support for inspect.

## cxml-base64 and cxml-urlencoded

This attribute to the HTML input element identifies the hidden form field that stores the cXML PunchOutOrderMessage document. The supplier must encode the PunchOutOrderMessage document using either base64 or URL encoding and indicate which encoding is used in the form field name attribute. For example:

```
<input type="hidden" name="cxml-base64" value=
"Entire text of base64-encoded cXML PunchOutOrderMessage document">
```

or

```
<input type="hidden" name="cxml-urlencoded" value=
"Entire text of URL-encoded cXML PunchOutOrderMessage document">
```

Suppliers can use cxml-urlencoded in cXML 1.1 and later.

## BuyerCookie Element

This element is used by procurement applications to associate a given PunchOutOrderMessage with its originating PunchOutSetupRequest. PunchOut sites must return this element whenever it appears. The PunchOutOrderMessage document must contain the same BuyerCookie that was used in the PunchOutSetupRequest document for this PunchOut session.

Do not use the BuyerCookie to track PunchOut sessions, because it changes for every session, from create, to inspect, to edit.

Example:

```
<BuyerCookie>1TF5JRP11S3W9</BuyerCookie>
```

Ariba Buyer discards it after it is used.

**Note:** The BuyerCookie value might expire while the user is navigating your site. Suppliers might want to support re-creation of a specific user's last shopping cart. If provided, this must be an option and for only that user.

## Total Element

This element contains the total value for order: quantity \* price without shipping and tax.

Example:

```
<Total>
  <Money currency="USD">100.23</Money>
</Total>
```

For more information about the Money element, see “[Money Element](#)” on page 18.

## ItemIn Element

This element passes PunchOut item details back to the requisition. The following example shows a single line item passed back:

```
<ItemIn quantity="1">
  <ItemID>
    <SupplierPartID>1234</SupplierPartID>
    <SupplierPartAuxiliaryID>ARB-65-1</SupplierPartAuxiliaryID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">2.10</Money>
    </UnitPrice>
    <Description xml:lang="en">3M POST-IT Cube, printed logo in 2 colors on
      side. 1 3/8 x 2 3/4 post-it cube of white paper. Approximately 345
      sheets. (Refill)</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="UNSPSC">14111514</Classification>
  </ItemDetail>
</ItemIn>
```

This element can also contain the `ItemType` and `parentLineNumber` attributes for an item group having child line items. For more information, see “[ItemOut Element](#)” on page 118.

For complete information on catalog items (`ItemID`, `UnitPrice`, `Description`, `UnitOfMeasure`, and `Classification`), see the *Ariba Catalog Format Reference*.

## ItemID Element

This element contains `SupplierPartID` and `SupplierPartAuxiliaryID`.

Suppliers can use the `SupplierPartAuxiliaryID` element as a supplier cookie to transport complex configuration and bill-of-goods information to identify the item when presented in the future. If `SupplierPartAuxiliaryID` contains special characters, such as additional XML elements not defined in the cXML protocol, they must be escaped properly.

Buying organizations do not display the `SupplierPartAuxiliaryID` element; instead they pass it back in `OrderRequest` documents for use by suppliers.

Example:

```
<ItemID>
  <SupplierPartID>1234</SupplierPartID>
  <SupplierPartAuxiliaryID>ARB-65-1</SupplierPartAuxiliaryID>
</ItemID>
```

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

## ItemDetail Element

This element contains detailed properties of a line item, such as unit price, description, UOM (Unit Of Measure), and PriceBasisQuantity.

Example:

```
<ItemDetail>
  <UnitPrice>
    <Money currency="USD">0.10</Money>
  </UnitPrice>
  <Description xml:lang="en">ACME Push Pins; Clear; Box Of 20</Description>
  <UnitOfMeasure>EA</UnitOfMeasure>
  <PriceBasisQuantity quantity = "2" conversionFactor = "0.10"
    <UnitOfMeasure>BX</UnitOfMeasure>
    <Description xml:lang = "en">This field specifies that 1 Box is equivalent to 10 EA
    and the unit price is for 2 Boxes</Description>
  </PriceBasisQuantity>
  <Classification domain="UNSPSC">4412210601</Classification>
  <ManufacturerPartID>ISBN-23455634</ManufacturerPartID>
  <ManufacturerName>Acme Manufacturing USA, Inc.</ManufacturerName>
  <LeadTime>4</LeadTime>
</ItemDetail>
```

## UnitPrice Element

This element contains the unit price of the item.

UnitPrice is required by the PunchOutOrderMessage for further processing by procurement applications. The user should be able to “edit” a requisition even if UnitPrice is not passed back.

## Description Element

This element contains the item description.

## UnitOfMeasure Element

This element contains the item Unit of Measure code as defined by the United Nations UOM standard. For more information, see Appendix A, “[Recommended Coding Systems](#).”



## PriceBasisQuantity Element

This element contains the quantity-based pricing for a line item. Quantity-based Pricing is commonly also referred to as Price-Based Quantity or PBQ. Quantity-based pricing allows the unit price of an item to be based on a different price unit quantity than 1. In addition to quantity-based pricing, Unit Conversion Pricing allows unit of measure conversion in the pricing calculation, when the unit of measure on the order differs from the pricing unit of measure.

Example:

```
<ItemOut isAdHoc = "yes" lineNumber = "1" quantity = "10" requestedDeliveryDate =
"2012-03-07">
  <ItemID>
    <SupplierPartID>N160INSTLL</SupplierPartID>
  </ItemID>
  \<ItemDetail>
    <UnitPrice>
      <Money currency = "USD">14.00000</Money>
    </UnitPrice>
    <Description xml:lang = "en">N160INSTLL</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <PriceBasisQuantity quantity = "2" conversionFactor = "0.10">
      <UnitOfMeasure>BX</UnitOfMeasure>
      <Description xml:lang = "en">This field specifies that 1 Box is equivalent to 10 EA
and the unit price is for 2 Boxes</Description>
    </PriceBasisQuantity>
  </ItemDetail>
</ItemOut>
```

## Classification Element

This element contains the UNSPSC commodity code. Ariba Buyer 8.0 and later and Procure-to-Pay recognize an optional UNSPSC version number. For more information, see Appendix A, “**Recommended Coding Systems.**”

Examples:

```
<Classification domain="UNSPSC">4412210601</Classification>
<Classification domain="UNSPSC_V7.1">4412210601</Classification>
```

## LeadTime Element

Ariba Buyer 8.2 (cXML 1.2.011) and later and Procure-to-Pay can interpret a LeadTime element in the ItemDetail element. PunchOut sites can include LeadTime to indicate how many business days it will take from receiving the purchase order to product delivery.

## Example PunchOutOrderMessage Document

The following is an example PunchOutOrderMessage document, which is hidden inside a cxml-ur1encoded form field.

```
<input type="hidden" name="cxml-ur1encoded" value=
"<!DOCTYPE cXML SYSTEM &#34;http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd&#34;>
<cXML payloadID=&#34;958074737352&amp;www.workchairs.com&#34;
timestamp=&#34;2004-06-14T12:59:09-07:00&#34;>
  <Header>
    <From>
      <Credential domain=&#34;DUNS&#34;>
        <Identity>12345678</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain=&#34;NetworkID&#34;>
        <Identity>AN01000002792</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain=&#34;www.workchairs.com&#34;>
        <Identity>PunchoutResponse</Identity>
      </Credential>
      <UserAgent>Our PunchOut Site V4.2</UserAgent>
    </Sender>
  </Header>
  <Message>
    <PunchOutOrderMessage>
      <BuyerCookie>1J3YVWU9QWMTB</BuyerCookie>
      <PunchOutOrderMessageHeader operationAllowed=&#34;edit&#34;>
        <Total>
          <Money currency=&#34;USD&#34;>14.27</Money>
        </Total>
      </PunchOutOrderMessageHeader>
      <ItemIn quantity=&#34;2&#34;>
        <ItemID>
          <SupplierPartID>3171 04 20</SupplierPartID>
          <SupplierPartAuxiliaryID>ContractId=1751 ItemId=417714
        </SupplierPartAuxiliaryID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency=&#34;USD&#34;>1.22</Money>
          </UnitPrice>
          <Description xml:lang=&#34;en&#34;>ADAPTER; TUBE; 5/32&#34;; MALE; #10-32 UNF;
FITTING</Description>
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Classification domain=&#34;UNSPSC&#34;>21101510</Classification>
          <ManufacturerName>Dogwood</ManufacturerName>
        </ItemDetail>
      </ItemIn>
      <ItemIn quantity=&#34;1&#34;>
        <ItemID>
          <SupplierPartID>3801 04 20</SupplierPartID>
          <SupplierPartAuxiliaryID>
            ContractId=1751 ItemId=417769
          </SupplierPartAuxiliaryID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency=&#34;USD&#34;>11.83</Money>
          </UnitPrice>
```

```

      <Description xml:lang="en">ADAPTER; TUBE; 5/32"; 2 PER PACK; MALE
#10-32 UNF; STAINLESS STEEL; FITTING</Description>
      <UnitOfMeasure>EA</UnitOfMeasure>
      <Classification domain="UNSPSC">21101510</Classification>
      <ManufacturerName>Legris</ManufacturerName>
      <LeadTime>2</LeadTime>
    </ItemDetail>
    <SupplierID domain="DUNS">022878979</SupplierID>
  </ItemIn>
</PunchOutOrderMessage>
</Message>
</cXML>
"
```

Below is the same example, but base64 encoded. Suppliers should use base64 encoding if there are any non-US-ASCII characters in the PunchOutOrderMessage document.

[illegible]

## PunchOut URL

---

Ariba Network receives PunchOut requests from buyers and routes them to suppliers' PunchOut sites.

There are three methods suppliers can use to specify the URLs of their PunchOut sites. In order of preference, they are:

- [URL Specified on the Supplier's PunchOut Site](#)
- [URL Specified on Ariba Network](#)
- [URL Specified in the Supplier's Index Catalog \(Deprecated\)](#)

### URL Specified on the Supplier's PunchOut Site

All cXML 1.1 (and later) PunchOut sites should specify their PunchOut URL through the cXML Profile transaction.

Suppliers specify their ProfileRequest URL in the **Configuration** area of their Ariba Network accounts and Ariba Network periodically sends ProfileRequest documents to query these sites. The sites respond with a ProfileResponse document listing URLs for all the cXML requests they support, including the URL for PunchOut requests. Ariba Network forwards PunchOutSetupRequest documents to that URL. This method is recommended, because suppliers can change their URLs without having to log in to Ariba Network. For more information about cXML profiles, see Chapter 4, "[Profile Transaction](#)."

If suppliers do not specify a profile URL on Ariba Network, it uses the method described below ([URL Specified on Ariba Network](#)) to determine their PunchOut URLs.

### URL Specified on Ariba Network

If suppliers do not support the cXML Profile transaction, Ariba Network checks whether they have entered a URL in their account. Suppliers can specify the URL of their PunchOut site in the **Configuration** area of their Ariba Network accounts.

If no URL is available from either the supplier's cXML profile or the supplier's Ariba Network account, the PunchOut request fails.

### URL Specified in the Supplier's Index Catalog (Deprecated)

This method for specifying a PunchOut URL is no longer supported. PunchOut index catalogs must continue to specify this URL, but Ariba Network does not use it as the PunchOutSetupRequest destination.

This URL is generated from the storefrontURL or the PunchoutDetail URL, depending on where the requisitioner is when they punch out. In either case, this value appears in the PunchOutSetupRequest.

Previous releases of Ariba Network forwarded cXML PunchOutSetupRequest documents to the URL contained in those documents. For improved security, Ariba Network now ignores these URLs. Ariba Network now forwards PunchOutSetupRequest documents to the URL specified in the receiving organization's cXML Profile or in its Ariba Network account.

The URL from the index catalog appears in the SupplierSetup element for the PunchOutSetupRequest document. The following example shows both cXML and CIF index catalogs and the resulting PunchOutSetupRequest:

cXML index catalog:

```
<PunchoutDetail>  
  <Description xml:lang="en-US">Desk Chairs</Description>  
  <URL>https://www.workchairs.com/punchout.asp</URL>  
  <Classification domain="UNSPSC">5136030000</Classification>  
</PunchoutDetail>
```

CIF index catalog:

```
CIF_I_V3.0  
CODEFORMAT: UNSPSC  
COMMENTS: This is an example of a PunchOut catalog item  
FIELDNAMES: Supplier ID, Supplier Part ID, Manufacturer Part ID, Item Description, SPSC Code, Unit  
Price, Unit of Measure, Lead Time, Manufacturer Name, Supplier URL, Manufacturer URL, Market Price,  
PunchOut Enabled  
CURRENCY: USD  
DATA  
762311901,A2C-311F,C-311F,"Desk Chairs",11116767,,,,,https://www.workchairs.com/punchout.asp,,,t  
ENDOFDATA
```

Resulting PunchOutSetupRequest segment:

```
<SupplierSetup>  
  <URL>https://www.workchairs.com/punchout.asp</URL>  
</SupplierSetup>
```

---

## URL vs. SelectedItem Elements

---

Depending on the cXML version used by PunchOut sites, the `PunchOutSetupRequest` might also contain a `SelectedItem` element specifying the item the user is punching out for:

- Ariba Buyer 6.1 and earlier (cXML 1.0) does not use `SelectedItem`. So, URLs from PunchOut index catalogs are the only ways to specify items to punch out for.
- Ariba Buyer 7 and later and Procure-to-Pay (cXML 1.1 and later) use `SelectedItem` to specify items to punch out for. `SelectedItem` uses Supplier Part ID and Supplier Part Auxiliary ID to identify the item. PunchOut sites can ignore the URL in the `PunchOutSetupRequest`, so suppliers can use made-up URLs in their index catalogs.

---

## Level 2 PunchOut

---

Buying organizations want improved catalog searching capability and quality when using the search interface in Ariba Buyer and Ariba Procure-to-Pay to find PunchOut catalog items. They want the best results when searching local CIF catalogs as well as supplier-managed PunchOut catalogs.

Level 2 PunchOut enables buying organizations to search for and find PunchOut items within their procurement application, instead of having to search each suppliers' site directly.

Level 2 makes suppliers' products more accessible to buying organizations, increases the visibility of suppliers' products, and increases visits to suppliers' PunchOut sites. It is supported in Ariba Buyer and Ariba Procure-to-Pay.

**Note:** If your customers use Ariba Procure-to-Pay, use the `punchoutLevel` attribute in your index catalogs to indicate where to place your items in the procurement application catalog. For more information, see “[punchoutLevel Attribute](#)” on page 78.

For details on the different levels of PunchOut items (for example, store, aisle, shelf, and product), see “[punchoutLevel Attribute](#)” on page 78 and the section on the `punchoutLevel` attribute in Chapter 2 of the *Ariba Catalog Format Reference*.

## User Experience

To understand the benefits of Level 2 PunchOut, you need to know how buying organizations find products.

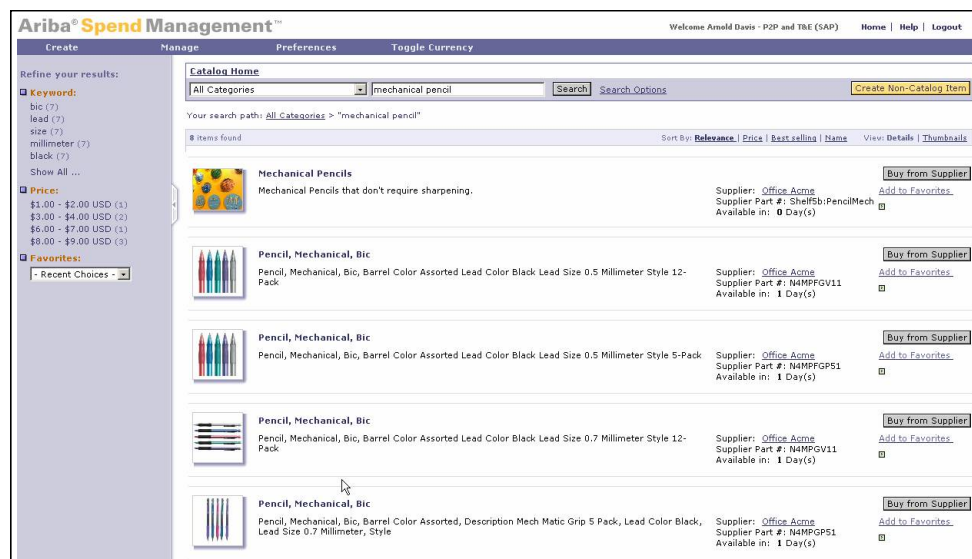
- **Item display and search**

With Level 2 PunchOut, users learn the semantics of the basic search mechanism in their procurement application, instead of having to learn the best means to search in each supplier's PunchOut site.

If the buying organization uses an Ariba procurement application, suppliers can optionally include hierarchical attributes to specify how to display the PunchOut item in the procurement system catalog. The `punchoutLevel` attributes `store`, `aisle`, `shelf`, or `product` cause the item to be displayed differently in the catalog. For example, the attributes `aisle` and `shelf` point to groups of related items. These items display at the top of categories, above products. If the supplier does not specify a `punchoutLevel` attribute, procurement applications consider the item to be a store-level item.

Users can find the PunchOut items using a keyword or supplier search. The following keywords are searchable:

- Item Description
- Short Name
- Supplier Name
- Buyer Part ID
- Supplier Part ID
- Manufacturer Part ID



### • Level 2 PunchOut Mechanism

Once an item has been found, the user accesses the supplier's PunchOut site to get more detailed product information, current pricing, and to configure products. The user must be presented with the option (a one-click button) to add the viewed item to the requisition and immediately be returned to the Ariba application. Suppliers should also add a **Browse** button to allow users to shop for additional items at the supplier's site.

The following example shows a typical user experience:

- 1 A user creates a requisition and searches for products. The procurement system displays all matching content.
- 2 The user chooses one of the supplier's products and clicks **Buy from Supplier**.
- 3 The supplier's PunchOut site displays more product details and current pricing for the specific item.
- 4 The user enters the quantity on the supplier's PunchOut site and clicks either **Add to Requisition and Return to Ariba** or **Browse** to continue shopping on the PunchOut site.

Depending on the user's choice, the PunchOut site either immediately returns the user and the selected item to the requisition in the Ariba application, or the user continues browsing the site.

## Level 2 PunchOut Format

These are the field requirements for a static catalog, a store-level PunchOut index catalog, and a Level 2 PunchOut index catalog.

The following fields should be provided, but only the indicated fields are required. See the *Ariba Catalog Format Reference* for additional information.

	Field Name	CIF	Store Level	Level 2	Field Requirements
1	Supplier ID	X	X	X	String: 255 chars
2	Supplier Part ID	X	X	X	String: 255 chars
3	Manufacturer Part ID				String: 255 chars
4	Item Description	X	X	X	String: 1000 chars
5	SPSC Code	X	X	X	String: 40 chars
6	Unit Price	X			Decimal
7	Units of Measure	X			String: 32 chars
8	Lead Time				Integer
9	Manufacturer Name				String: 255 chars
10	Supplier URL		X	X	String: 255 chars
11	Manufacturer URL				String: 255 chars
12	Market Price				Decimal
13	Short Name				String: 50 chars
14	Image				String: 255 chars
15	Thumbnail				String: 255 chars
16	PunchOut Enabled		X	X	Boolean: TRUE, FALSE
17	PunchOutLevel			X	store, aisle, shelf, or product
18	Supplier Part Auxiliary ID				String: 255 chars
19	Parametric Name				String: 255 chars
20	Parametric Data				String: 255 chars



## Level 2 PunchOut Requirements

### CIF and cXML Index Content Requirements

#### Required Fields

- PunchOutLevel attribute (store-, aisle-, shelf-, or product-level items)
- Unique Supplier Part ID/Supplier Part Auxiliary ID combination (key identifier for the selected index item; values are included in the PunchOutSetupRequest document)
- Rich Item Description (searchable product name, product attributes, and so on; for store-, aisle-, and shelf-level items, only the first 50 characters are displayed)
- UNSPSC code (use detailed levels for shelf- and product-level items)
- Supplier URL (required for compatibility with Ariba Buyer 8.x; a dummy URL is sufficient)

Additional information that is also recommended:

- Manufacturer and Manufacturer Part ID
- Parametric data (as applicable)

#### Optional Fields

- Short Name (if you use this field, then Item Description is not displayed except for product-level items)
- Image link (image is displayed as a thumbnail only for product-level items)
- Unit Price or Market Price provides an estimated price only for product-level items (cannot be used for other item levels)

### Level 2 PunchOut Site Requirements

#### Minimum Requirements

- Direct access to the item or shelf specified in the Index catalog
- Immediate one-click **Return to Ariba** button to post the items from the supplier site to the Ariba application
- Support for PunchOut edit to allow the user to return to the shopping session and modify an item quantity, remove an item, or add an item
- Support for PunchOut inspect so that requisitioners and approvers can view the shopping cart but will not have the ability to modify the items in the cart
- Option to continue shopping in PunchOut site, rather than return immediately with item selected

For Shelf Level—additional refinement should not be required to select an item after the associate reaches the site. The site should present a short list of items immediately.

### Other Requirements

- You must be able to provide timely updates for the PunchOut index as changes occur (updates mostly depend upon frequency of changes in index; once a quarter is best practice).
- Optionally, you can upload an updated PunchOut index to Ariba Network programmatically, using the CatalogUploadRequest transaction (see “[CatalogUploadRequest Element](#)” on page 220 for more details). This is a good way to reduce workload.
- You must be able to support orders from multiple PunchOut sessions to the site. (The supplier cookie stored in the SupplierPartAuxiliaryID method traditionally used to match PunchOut sessions to orders is more difficult to support with Level 2 PunchOut than site-level PunchOut).

## Example Index Catalogs

You can specify Level 2 PunchOut items in CIF or cXML index catalogs. However, for the smallest file, create the index catalog in cXML format instead of CIF. The first index catalog example below shows store-level PunchOut with one PunchOut item. The second index catalog example shows Level 2 PunchOut, with expanded product descriptions.

### Store-Level PunchOut Index Catalog

```
CIF_I_V3.0
LOADMODE: F
CODEFORMAT: UNSPSC_V13.5
COMMENTS: Store-Level PunchOut
SUPPLIERID_DOMAIN: NETWORK_ID
ITEMCOUNT: 1
TIMESTAMP: 2006-02-18 00:00:00
FIELDNAMES: Supplier ID, Supplier Part ID, Manufacturer Part ID, Item Description, SPSC Code, Unit
Price, Unit of Measure, Lead Time, Manufacturer Name, Supplier URL, Manufacturer URL, Market Price,
PunchOut Enabled
DATA
AN100000123,A2C,C-311F,"ACME Garden Supply",27112000,,,,,,,,t
ENDOFDATA
```

## Level 2 PunchOut Index Catalog

```

CIF_I_V3.0
LOADMODE: F
CODEFORMAT: UNSPSC_V13.5
COMMENTS: Level 2 PunchOut
SUPPLIERID_DOMAIN: NETWORK_ID
ITEMCOUNT: 198
TIMESTAMP: 2006-02-18 00:00:00
FIELDNAMES: Supplier ID, Supplier Part ID, Manufacturer Part ID, Item Description, SPSC Code, Unit
Price, Unit of Measure, Lead Time, Manufacturer Name, Supplier URL, Manufacturer URL, Market Price,
PunchOut Enabled, PunchoutLevel, Image
DATA
AN100000123,A2C,C-311F,"Lawn Maid 4 HP side discharge lawnmower. Briggs and Stratton 4.0 hp engine
delivers the power for any property. Side Discharge model with a 22"" Quick Mulch Deck. 6"" rear
wheels for easy mowing. For flat
terrain.",27112014,,,,,,,,t,product,https://www.acme.com/images/A2C.jpg
AN100000123,A3C,C-312F,"Lawn Maid 6 HP self-propelled side discharge lawnmower. Briggs and
Stratton 6.0 hp engine delivers the power for any lawn. Side Discharge model with a 24"" Quick
Mulch Deck. 6"" rear wheels for easy
mowing.",27112014,,,,,,,,t,product,https://www.acme.com/images/A3C.jpg
AN100000123,A4C,C-316F,"Lawn Maid Post Hole Digger. Drills down to 4 feet.
Chromemolly",27112013,,,,,,,,t,product,https://www.lawnsRUs.com/images/A4C.jpg
AN100000123,X7H,52429,"Precision Touch 12-amp Electric Blower. 2-speed blower offers the
lightweight airpower you need to blow away debris. Double
insulated.",27112701,,,,,,,,t,product,https://www.acme.com/images/X7H.jpg
.
.
.
ENDOFDATA

```

The Level 2 PunchOut index catalog provides much more content. It lists each PunchOut item and the descriptions contain enough data so that users can find specific products. The more complete your descriptions, the more likely users will find your items when they search.

Each index item also gives requisitioners enough information to decide whether to punch out to view your product page. For example, it lists a catalog image file, which displays a picture of the product in Ariba Buyer 8.2.2 or later and Procure-to-Pay. It also uses the `punchoutLevel` attribute to indicate that the item is a product-level PunchOut item in Procure-to-Pay.

## Resulting PunchOutSetupRequest

If a user selects the first item from the Level 2 PunchOut index catalog above, you would receive the following PunchOutSetupRequest document:

```
<?xml version="1.0"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="958075346970@www.bigbuyer.com" timestamp="2006-03-19T12:57:09-07:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN000002792</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN100000123</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>sysadmin@ariba.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ariba Buyer 8.2.2</UserAgent>
    </Sender>
  </Header>
  <Request>
    <PunchOutSetupRequest operation="create">
      <BuyerCookie>1J3YVWU9QWMTB</BuyerCookie>
      <Extrinsic name="CostCenter">610</Extrinsic>
      <Extrinsic name="User">jpicard</Extrinsic>
      <BrowserFormPost>
        <URL>http://bigbuyer.com:3377/punchout</URL>
      </BrowserFormPost>
      <SupplierSetup>
        <URL></URL>
      </SupplierSetup>
      <ShipTo>
        <Address addressID="001">
          <Name xml:lang="en">BigBuyer Headquarters</Name>
          <PostalAddress>
            <DeliverTo>Jean Picard</DeliverTo>
            <Street>1565 Pine, MS A.2</Street>
            <City>New York</City>
            <State>NY</State>
            <PostalCode>01043</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
        </Address>
      </ShipTo>
      <Contact>
        <Name>jpicard</Name>
      </Contact>
      <SelectedItem>
        <ItemID>
          <SupplierPartID>A2C</SupplierPartID>
        </ItemID>
      </SelectedItem>
    </PunchOutSetupRequest>
  </Request>
</cXML>
```

Your PunchOut site uses the `SupplierPartID` element to determine which product to display. If you need to differentiate variations of a product for size, color, or language, use the `SupplierPartAuxiliaryID` field in your index catalog; procurement systems copy it to the `SelectedItem` element.

## Level 2 PunchOut Index Catalog

The following example is an item-level PunchOut index catalog:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Index SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.012/cXML.dtd">
<Index>
  <SupplierID domain="duns">611429481</SupplierID>
  <Comments xml:lang="en-US">
    Sample cXML/Index
  </Comments>
  <IndexItem>
    <IndexItemPunchout>
      <ItemID>
        <SupplierPartID>1-57231-805-8</SupplierPartID>
      </ItemID>
      <PunchoutDetail punchoutLevel="product">
        <Description xml:lang="en-US">Whiteboard markers, one dozen</Description>
        <URL>http://www.whitebd.com/cXML/PunchoutSetup/Punchoutshop.asp</URL>
        <Classification domain="UNSPSC">55101524</Classification>
      </PunchoutDetail>
    </IndexItemPunchout>
  </IndexItem>
  <IndexItem>
    <IndexItemPunchout>
      <ItemID>
        <SupplierPartID>VTS-4976-200</SupplierPartID>
      </ItemID>
      <PunchoutDetail punchoutLevel="product">
        <Description xml:lang="en-US">Whiteboard Eraser, felt</Description>
        <URL>http://www.whitebd.com/cXML/PunchoutSetup/Punchoutshop.asp</URL>
        <Classification domain="UNSPSC">43232005</Classification>
      </PunchoutDetail>
    </IndexItemPunchout>
  </IndexItem>
  <IndexItem>
    <IndexItemPunchout>
      <ItemID>
        <SupplierPartID>GS3600</SupplierPartID>
      </ItemID>
      <PunchoutDetail punchoutLevel="product">
        <Description xml:lang="en-US">Whiteboard Cleaner, non-toxic</Description>
        <URL>http://www.whitebd.com/cXML/PunchoutSetup/Punchoutshop.asp</URL>
        <Classification domain="UNSPSC">52161512</Classification>
      </PunchoutDetail>
    </IndexItemPunchout>
  </IndexItem>
</Index>
```

## Shelf-Level PunchOut

Shelf-level PunchOut is a way of offering similar products through one PunchOut index item. Consider using this type of PunchOut if you have similar products from multiple manufacturers or a single product available in multiple configurations. You create a product-selector page on your PunchOut site that enables requisitioners to choose a specific configuration or SKU.

For example, you might offer lawnmowers from several manufacturers:

Lawn Maid 4-HP gas 20-inch deck self-powered lawnmower, side discharge  
 Lawn Maid 6-HP gas 25-inch deck self-powered lawnmower, side discharge  
 Precision Touch 4-HP gas 12-inch deck lawnmower, side discharge  
 Precision Touch 5-HP gas 15-inch deck lawnmower, rear discharge

The product-selector page on your PunchOut site could display these models with links to datasheets. You would add the following item to your PunchOut index catalog so requisitioners could find this product category and punch out to your product-selector page:

```
AN100000123,lawnmowers,XXX,"Powerful and Efficient Gas
Lawnmowers",27112014,,,,,,,,t,https://www.acme.com/images/lawnmower.jpg,shelf
```

When users punch out to your product-selector page, they do not have to perform a search to find the specific product or configuration. Instead, your page displays all available products of that type.

Use shelf-level PunchOut for specific products, not for groups of related products. For example:

Example Supplier	Specific Products	Groups of Products
Computer Store	Computer Mice	Computer Peripherals
Hardware Store	Handsaws	Tools
Grocery Store	Cereals	Breakfast Foods

## Uploading Index Catalogs

Level 2 PunchOut requires you to upload your index catalogs to Ariba Network more often than for store-level PunchOut to ensure that your customers have your latest product offerings. Depending on the volatility of your offerings, you might want to update your catalogs monthly, weekly, or even daily. Each time you upload a catalog, your customer's procurement system automatically downloads it and incorporates it into the local search index.

**Note:** For Level 2 PunchOut index catalogs, cXML is the preferred format because the file size is smaller than CIF.

You have two options for uploading catalogs:

- **Manual Upload.** Manual upload requires you to log in to your Ariba Network account to upload and publish your cXML index catalogs. For more information about manually uploading catalogs, see the *Ariba Network Catalog Administration Guide for Suppliers*.
- **Automatic Upload.** Automatic upload uses the cXML CatalogUpload transaction to upload and publish your cXML index catalogs. You generate a CatalogUploadRequest document and include your cXML catalog as a MIME attachment. For more information about automatic upload, see Chapter 11, "[Catalog Upload Transaction](#)."

Your customers automatically poll Ariba Network for updated catalogs, usually once per day. You might want to contact them to find out what time this polling takes place so you can have your updated index catalogs ready.

## PunchOut Session Timeout

If PunchOut sessions do not end with a PunchOutOrderMessage document, Ariba Buyer and Procure-to-Pay terminate the session after six hours.

## ASP Examples

The following ASP examples can be used as a starting point to rapidly deploy a PunchOut site. They demonstrate how to receive PunchOutSetupRequest documents and generate PunchOutSetupResponse documents.

These are basic examples and are not ready for production. For example, credential variables attached in the query string are not acceptable for production code. These examples use variables that have not been declared. Because a global variable file has not been included with this example, change the following variables in the appropriate locations:

- receivePunchoutSetupRequest.asp
 

```
myAribaPassword = "welcome"
myURL = http://supersupplier.com/punchoutasp
```
- resolveXML.asp
 

```
olddtdvalue = ""cXML.dtd""
newdtdvalue = ""http://supersupplier.com/cXML.dtd""
```

### receivePunchoutSetupRequest.asp

This file receives the PunchOutSetupRequest document and generates a valid PunchOutSetupResponse document containing the first page URL:

```
<?@LANGUAGE = VBScript?>
<%
    Dim myAribaPassword
    Dim myURL
%>
<%
' *****Comments*****
' Include file that loads http post cXML and parses with MS DOM
' *****
%>
<!--#include file="resolveXML.asp" --->
<%
' *****Comments*****
' Variables needed for example. Very simple validation for asp
' Set sharedsecret password expected from Ariba Network
' *****
    myAribaPassword = "welcome"
    myURL = "http://workchairs.com/punchoutaspv2/"
%>
```

```

<%
'*****Comments*****
' Basic Shared Secret validation.
'
' Please note: You will need to fix the timestamp and payloadID. The
' f() now is currently invalid and will fail on some builds of Ariba
' Buyer as well as other product lines. Please use the format
' specified in the cXML User's Guide available at http://www.cXML.org.
' See file TCcXMLFormatDTime.asp for a good start on formatting the
' timestamp in ISO 8601 format.
'*****

if (myAribaPassword <> sharedSecret) then %>
<?xml version="1.0" ?>
<!DOCTYPE cXML SYSTEM "cXML.dtd">
<cXML payloadID="&lt;%= Now &amp;quot;@&quot;&amp;
Request.ServerVariables(&quot;LOCAL_ADDR&quot;)%&gt;" timestamp="&lt;%= Now %&gt;">
<Response>
<Status code="500" Text="Invalid document" />
</Response>
</cXML>
<% else %>
<?xml version="1.0" ?>
<!DOCTYPE cXML SYSTEM "cXML.dtd">
<cXML payloadID="&lt;%= Now &amp;quot;@&quot;&amp;
Request.ServerVariables(&quot;LOCAL_ADDR&quot;)%&gt;" timestamp="&lt;%= Now %&gt;">
<Response>
<Status code="200" text="Success">
</Status>
<PunchOutSetupResponse>
<StartPage>
<URL>
<%=myURL%>
<p>b2bsite/shop.asp?fromIdentity=<%= fromIdentity%>&amp;operation=<%=
operation%>&amp;buyerCookie=<%= buyerCookie%>&amp;BrowserFormPost=<%=
BrowserFormPost%></URL></StartPage></PunchOutSetupResponse></Response></cXML> <%end if%> </p>

```



## resolveXML.asp

This file loads the HTTP POST into a Microsoft DOM object and extracts data for the PunchOutSetupResponse document:

```

<%
Dim xml
Dim xdoc
Dim xml2
'*****Comments*****
' MSDOM cannot resolve local DTDs, so replace definition with URL.
' Get DTDs from http://www.cXML.org and store them locally on your
' server for performance.
'*****
Dim oldtdvalue
Dim newtdvalue
oldtdvalue = ""cXML.dtd""
newtdvalue = ""http://workchairs.com/cxml1.2/cXML.dtd""

if (Request.ServerVariables("REQUEST_METHOD") = "POST") then
'*****Comments*****
' This command reads the incoming HTTP cXML Request
' Note: this does not currently handle cXML documents sent with
' content-type: text/xml or any transfer that specifies the MIME
' charset attribute on that header.
'*****
    totalBytes = Request.TotalBytes
    IF totalBytes > 0 THEN
        xml = Request.BinaryRead( totalBytes )
        for i = 1 to totalBytes
            xmlstr = xmlstr + String(1,AscB(MidB(xml, i, 1)))
        Next
        xml2 = xmlstr
        xml2 = Replace(xml2,oldtdvalue,newtdvalue)
        xml2 = Replace(xml2,"utf-8","utf-16")
        xml2 = Replace(xml2,"UTF-8","UTF-16")
    END IF

'*****Comments*****
' Create MSDOM object and set load values.
'*****
Set xdoc = Server.CreateObject("Microsoft.XMLDOM")
xdoc.ValidateOnParse = False
xdoc.async = False
xdoc.resolveExternals = False
loadStatus = xdoc.loadXML(xml2)

'*****Comments*****
' Create MSDOM object and set load values.
' Note: these XML retrievals handle only the first of any list of
' credential elements
'*****
If loadStatus = True then
    Set fromIdentity = xdoc.getElementsByTagName("Header/From/Credential/Identity")
    fromIdentity = (fromIdentity.item(0).text)
    Set toSuppCred = xdoc.getElementsByTagName("Header/To/Credential/Identity")
    toSuppCred = (toSuppCred.item(0).text)
    Set senderCred = xdoc.getElementsByTagName("Header/Sender/Credential/Identity")
    senderCred = (senderCred.item(0).text)
    Set sharedSecret = xdoc.getElementsByTagName("Header/Sender/Credential/SharedSecret")
    sharedSecret = (sharedSecret.item(0).text)
    Set fromUserAgent = xdoc.getElementsByTagName("Header/Sender/UserAgent")
    fromUserAge = (fromUserAgent.item(0).text)

```

```

        Set operation =
xdoc.documentElement.childNodes(1).childNodes(0).attributes.getNamedItem("operation")
        operation =
xdoc.documentElement.childNodes(1).childNodes(0).attributes.getNamedItem("operation").text
        Set buyerCookie = xdoc.getElementsByTagName("Request/PunchOutSetupRequest/BuyerCookie")
        buyerCookie = (buyerCookie.item(0).text)
        Set buyExtrinsics = xdoc.getElementsByTagName("Request/PunchOutSetupRequest/Extrinsic")
        For i = 0 To (buyExtrinsics.length -1)
            BuyExtrinsicVars = (buyExtrinsics.item(i).text) & "," & BuyExtrinsicVars
        Next
        Set BrowserFormPost =
xdoc.getElementsByTagName("Request/PunchOutSetupRequest/BrowserFormPost")
        BrowserFormPost = (BrowserFormPost.item(0).text)

'*****Comments*****
' Some nice MSDOM error logging for a failed parse or load.
'*****
Else
    Response.Write " <P> xml @ supplier site failed to load using MSDOM: "
    Dim strErrText
    Dim xPE

    Set xPE = xdoc.parseError
    strErrText = "Your XML Document failed to load due the following error: " & "Error #: " &
xPE.errorCode & ": " & "Line #: " & xPE.Line & "Line Position: " & xPE.linepos & "Position In
File: " & xPE.filepos & "Source Text: " & xPE.srcText & "Document URL: " & xPE.url
    Response.Write strErrText
    End If
Else
'*****Comments*****
' ASP page was called using a GET. Functions expect a post.
'*****
    Response.Write " <P> Wrong Method Get: Post supported only"
End if
%>

```

## TCcXMLFormatDTime.asp

This file contains a function to format a timestamp in ISO 8601 format.

```

'*****
' This function does not do the time adjustment but provides a great
' start in helping you out.
'*****
<%
private function TCcXMLFormatDTime(strDTimeIn)
'*****
' Purpose:
' This function returns a date/time value formatted for cXML messaging
' cXML uses the ISO 8601 date/time standard.
'*****
'Input:
' strDTimeIn      Input date/time to be formatted.
,
'Ouptut:   n/a
,
'Return: String value containing the formatted date/time.
,
    const conRoutine = "TCcXMLFormatDTime"
    dim strTempOutDtime
    dim strDay

```

```
dim strMonth
dim strTime

TcXMLFormatDTime = ""
strTime = FormatDateTime(strDTimeIn, 4) ' Short time.

' Set the year.
strTempOutDtime = Year(strDTimeIn) & "-"
' Set the month.
strMonth = Month(strDTimeIn)
if len(strMonth) = 1 then strMonth = "0" & strMonth
strTempOutDtime = strTempOutDtime & strMonth & "-"
' Set the day and the date/time delimiter.
strDay = Day(strDTimeIn)
if len(strDay) = 1 then strDay = "0" & strDay
strTempOutDtime = strTempOutDtime & strDay & "T"
' Set the time.
strTempOutDtime = strTempOutDtime & strTime

TcXMLFormatDTime = strTempOutDtime

exit function
end function
%> </p>
</body>
</html>
```



---

## Chapter 7 Purchase Orders

- “[OrderRequestHeader Element](#)” on page 109
- “[ItemOut Element](#)” on page 118
- “[Cancel Orders and Change Orders](#)” on page 128
- “[Implementation Hints and Limitations](#)” on page 131
- “[Response Documents](#)” on page 138

cXML OrderRequest documents represent purchase orders. These documents consists of one OrderRequestHeader element and one or more ItemOut elements. Ariba Buyer sends these documents to Ariba Network. cXML-enabled suppliers respond with OrderResponse documents.

---

### OrderRequestHeader Element

The OrderRequestHeader element contains contact and shipping information that applies to the entire order.

#### orderID Attribute

This attribute is a user-visible number generated by Ariba Buyer unique to the buying organization. Buying organizations that subscribe to the vPayment service must use a value shorter than 16 characters.

For information about orderID and change orders, see “[Change Orders](#)” on page 130. For information about the vPayment service, see “[vPayment](#)” on page 135.

#### orderDate Attribute

This attribute is the date and time the order was created by the buying organization. Dates are in ISO 8601 format: YYYY-MM-DDThh:mm:ss-hh:mm.

```
<OrderRequest>
  <OrderRequestHeader orderID="D0123" orderDate="2000-09-09T22:54:59-06:00">
    ...
  </OrderRequestHeader>
</OrderRequest>
```

#### orderType Attribute

This attribute describes the order type: regular, release (a release against a master agreement), or blanket (a blanket purchase order, or BPO).

## releaseRequired Attribute

If the order is a blanket purchase order, the `releaseRequired` attribute indicates whether the blanket order requires releases (purchase orders). If “no” is specified, the blanket order does not require purchase orders and can be directly billed against. The default is “yes.”

## effectiveDate Attribute

If the order is a blanket purchase order, the `effectiveDate` attribute indicates the date that the order is available for ordering. This attribute is currently used only with blanket purchase orders.

## expirationDate Attribute

If the order is a blanket purchase order, the `expirationDate` attribute indicates the date that the order is no longer available. If no value is defined, the end date can be indefinite. This attribute is currently used only with blanket purchase orders.

## addressID Attribute

This attribute identifies the buying organization’s shipping or billing address defined in the `ShipTo` and `BillTo` elements of `OrderRequest`. This is also the buying organization’s Extrinsic “UniqueName” for `ShipTo/BillTo`.

The `Address` element contains a `PostalAddress` and is similar to the `Contact` element, except that `Contact` can describe multiple methods for contacting a particular person.

```
<Address isoCountryCode="US" addressID="001">
```

## Name Element

For ship to, this value should be the company or organization of the employee receiving ordered products. For bill to, this is the department or group responsible for payment. `Name` is not as specific as the location referenced in the second `DeliverTo` line.

```
<Name xml:lang="en">Workchairs, Inc.</Name>
```

## DeliverTo Element, Line One

This element is the name of the person receiving the ordered products. Avoid empty or white space elements and attributes. Missing values might affect EDI and cXML suppliers.

The suggested implementation name format template is “firstname lastname.”

**Note:** Both `DeliverTo` lines in the `BillTo` element should be in the same format for consistency. This logic is not in the existing templates; it is recommended that it be added manually.

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

## DeliverTo Element, Line Two

This element is the location (building, city, office, or mailstop) of the person receiving the ordered products. Locations should always be complete enough for a mailing label.

There should be consistency between uses of the same element, particularly ShipTo and BillTo.

```
<DeliverTo>Mailstop M-543</DeliverTo>
```

## Street Element

This element is the street address of the ShipTo or BillTo location where ordered products are to be delivered and billed. If there are multiple lines in the street address, use up to four Street elements instead of inserting new lines.

```
<Street>123 Anystreet</Street>  
<Street>M/S 450</Street>
```

## City Element

This element is the city where ordered products are to be shipped or billed.

```
<City>Sunnyvale</City>
```

## State Element

This element is a two-letter state, province, or territory code for the location where the goods are to be shipped and billed. For more information, see “[Country, State, and PostalCode Elements](#)” on page 22.

```
<State>CA</State>
```

## PostalCode Element

This element is the postal or zip code where goods are to be shipped and billed. Do not use a dash (-) in US extended zip codes. For more information, see “[Country, State, and PostalCode Elements](#)” on page 22.

```
<PostalCode>90489</PostalCode>
```

## CarrierIdentifier Element

This element contains the carrier name of the shipment. For example:

```
<ShipTo>
  <Address>
    <Name xml:lang="USD">Acme</Name>
    <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>
  <<CarrierIdentifier domain="companyName">UPS</CarrierIdentifier>
```

## TransportInformation Element

This element contains the transport information for a purchase order or ship notice. This element is specified only at the header-level. The TransportInformation element contains the following elements:

Element	Description
Route	<p>Shipping method for the shipment. This is required if you select a carrier.</p> <p>This element has the following attribute:</p> <ul style="list-style-type: none"> <li>Method</li> </ul> <p>The possible values for shipping method are:</p> <ul style="list-style-type: none"> <li>air</li> <li>motor</li> <li>rail</li> <li>ship</li> </ul>
ShippingContractNumber	The contract number associated to the contract for sale.
ShippingInstructions	Information for the shipment

Here is an example for the TransportInformation element:

```
<OrderRequestHeader orderDate="2010-03-26T16:40:53" orderID="POw4401" orderType="regular" type="Update">
  <Total>
    <Money currency="USD">10.00</Money>
  </Total>
  <ShipTo>
    <Address>
      <Name xml:lang="USD">Acme</Name>
      <PostalAddress name="default">
        <DeliverTo>Joe Smith</DeliverTo>
        <DeliverTo>Mailstop M-543</DeliverTo>
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>AL</State>
        <PostalCode>35762</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Address>
  </ShipTo>
</OrderRequestHeader>
```



```

    </PostalAddress>
  </Address>
  <CarrierIdentifier domain="companyName">UPS</CarrierIdentifier>
  <TransportInformation>
    <Route method="motor"/>
    <ShippingContractNumber>145</ShippingContractNumber>
    <ShippingInstructions>
      <Description xml:lang="en-US">As per the contract</Description>
    </ShippingInstructions>
  </TransportInformation>

```

## Country isoCountryCode

This attribute is the country code from the ISO 3166 standard. The content for Country is a human-readable or printable name. For more information, see [“Country, State, and PostalCode Elements”](#) on page 22.

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

## TelephoneNumber Element

This element is the telephone number of the person or department where ordered products are shipped to or billed.

For international dialing, the CountryCode contains the ITU dialing code for a country after any escape codes. The ITU dialing code is the access code for a particular country. For more information, see Appendix A, [“Recommended Coding Systems.”](#)

United States:

```

<TelephoneNumber>
  <CountryCode isoCountryCode="US">1</CountryCode>
  <AreaOrCityCode>650</AreaOrCityCode>
  <Number>9308410</Number>
</TelephoneNumber>

```

London:

```

<TelephoneNumber>
  <CountryCode isoCountryCode="UK">44</CountryCode>
  <AreaOrCityCode>20</AreaOrCityCode>
  <Number>78628500</Number>
</TelephoneNumber>

```

## Fax Element

This element is the fax number of the person or department where ordered products are to be shipped or billed.

United States example:

```

<Fax name="work">
  <TelephoneNumber>
    <CountryCode isoCountryCode="US">1</CountryCode>
    <AreaOrCityCode>408</AreaOrCityCode>
    <Number>3582100</Number>
  </TelephoneNumber>

```

```
</Fax>
```

London example:

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

## Email Element

This element is the email address of a person or a department where ordered products are to be shipped or billed. Ariba Network sends status information to the ShipTo Email address when the order status is updated, or as the result of a ConfirmationRequest or ShipNoticeRequest.

Ariba Network uses the optional preferredLang attribute (produced by Ariba Buyer 8.2 or later) in the Email element to specify the recipient's language.

```
<Email>wsmithers@springfield.com</Email>
```

```
<Email preferredLang="ja-JP">mburns@springfield.com</Email>
```

## TermsofDelivery Element

This element specifies the terms of delivery in a purchase order or ship notice. The TermsofDelivery element can appear at the header-level or line-item level. To add at line-item level, include this element to the ItemOut element. For more information, see “[TermsofDelivery Element](#)” on page 125.

**Note:** You can also add this element to the ShipNoticeHeader to specify terms at the header level. To add at the line-item level, include it to the ShipNoticeItem element. For more information, see “[TermsofDelivery Element](#)” on page 153.

The TermsofDelivery element contains the following elements:

Element	Description
TermsofDeliveryCode	Standard delivery terms and Incoterms.
ShippingPaymentMethod	Denotes the mode of shipping payment. <ul style="list-style-type: none"> <li>Account: When shipping charges are charged to an account.</li> <li>Collect: When the consignee pays the freight charges.</li> <li>Prepaid by Seller: When the seller makes the payment to the carrier for freight charges prior to a shipment.</li> <li>Mixed: When the consignment is partially Collect and partially Prepaid.</li> <li>Other: Any other shipping payment method or the third-party pays the shipment charges. You can enter additional information for the payment method.</li> </ul>

Element	Description
TransportTerms	The terms of transportation: <ul style="list-style-type: none"> <li>• Free-Carrier</li> <li>• CostAndFreight</li> <li>• DeliveredAtFrontier</li> <li>• Other: When you specify this option, you can additionally enter a description.</li> </ul>
Address	The Deliver To address for the ship notice.
Comments	Additional information for the delivery terms.
Comments	Additional information when “Other” Transport Term is selected.

Here is an example of the TermsOfDelivery element:

```

<TermsOfDelivery>
  <TermsOfDeliveryCode value="PriceCondition"/>
  <ShippingPaymentMethod value="AdvanceCollect"/>
  <TransportTerms value="Other">Contract Terms</TransportTerms>
  <Address>
    <Name xml:lang="en-US">SNV</Name>
    <PostalAddress name="default">
      <Street>123 street</Street>
      <City>Sunnyvale</City>
      <State>AL</State>
      <PostalCode>35762</PostalCode>
      <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
  </Address>
  <Comments xml:lang="en-US" type="Transport">Transport instructions</Comments>
  <Comments xml:lang="en-US" type="TermsOfDelivery">Delivery at the doorstep</Comments>
</Terms Of Delivery>

```

## URL Element

This element is the ship to URL. This element is not required to be populated.

```
<URL>https://www.workchairs.com</URL>
```

## Tax Element

This element is the tax associated with the purchase 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 level tax amounts.

```

<Tax>
  <Money currency="USD">1.34</Money>
  <Description xml:lang="en">Sales Tax</Description>
</Tax>

```

Two extrinsic elements can be added to the Tax element:

- withholdingTax (for total withholding taxes), and
- withholdingTaxTotal (for total taxes minus withholding tax)

The Money element in Tax element is used to capture totals for all tax amounts and is not used for invoice display. The use of negative rates ensures backward compatibility of existing cXML applications.

The following is an example of how withholding tax data in an invoice is displayed when exported to cXML. A withholding tax rate of -2% was applied to the invoice that was submitted via Ariba Network:

```
<Tax><Money currency="USD">1273.35</Money><Description xml:lang="en-US"></Description>
  <TaxDetail category="withholdingTax" percentageRate="-2">
    <TaxableAmount><Money currency="USD">25466.90</Money></TaxableAmount>
    <TaxAmount><Money currency="USD">-509.34</Money></TaxAmount>
    <Description xml:lang="en-US"></Description>
  </TaxDetail>
  <TaxDetail category="vat" percentageRate="7" taxPointDate="2011-03-11T00:00:00-08:00">
    <TaxableAmount><Money currency="USD">25466.90</Money></TaxableAmount>
    <TaxAmount><Money currency="USD">1782.69</Money></TaxAmount>
    <Description xml:lang="en-US"></Description>
  </TaxDetail>
  <Extrinsic name="withholdingTaxTotal"><Money currency="USD">-509.34</Money></Extrinsic>
  <Extrinsic name="taxTotal"><Money currency="USD">1782.69</Money></Extrinsic>
</Tax>
```

## Modifications Element

The Modifications element can be used to include the total landed cost of an item. This element can store:

- Any modification to the original price or shipping price of the item
- A set of one or more Modification elements

You can add the Modifications element to the Shipping element.

The Modification element contains details of the allowances and charges applicable at the header-level and line-item level. It has the following elements:

Element	Description
OriginalPrice	<p>The original price of the item. The allowances and charges are applied on the original price.</p> <p>This element has the Money element. It also contains an optional type attribute.</p> <p>For example, the type value can be the MSRP, ListPrice, Actual, AverageSellingPrice, CalculationGross, BaseCharge, AverageWholesalePrice, ExportPrice, AlternatePrice, ContractPrice, etc</p>
AdditionalDeduction	<p>The details of the deductions available for the item. Used only when allowances are applicable.</p> <p>This element can have any one of the following elements that defines the deduction value:</p> <ul style="list-style-type: none"> <li>• DeductionAmount: This element has the Money element.</li> <li>• DeductionPercent: This has a percent attribute.</li> <li>• DeductedPrice: This element has the Money element. It contains the final price of the item. This price overrides the price of the item.</li> </ul> <p>The AdditionalDeduction element has an optional type attribute. It contains details on the type of deductions available for the item</p>

Element	Description
AdditionalCost	<p>The details of the additional charges applied on an item. This element can be specified when the AdditionalDeduction element is not specified.</p> <p>It contains the following elements:</p> <ul style="list-style-type: none"> <li>• Money: Enter the money value in the value attribute. This is a mandatory attribute.</li> </ul> <p><b>Note:</b> Do not use this element for shipping, special handling, freight, etc.</p> <ul style="list-style-type: none"> <li>• Percentage: Enter the percentage value in the percent attribute. This is a mandatory attribute.</li> </ul>
ModificationDetail	<p>The details of any information for the AdditionalDeduction or AdditionalCost element.</p> <p>This ModificationDetail element has the following elements:</p> <ul style="list-style-type: none"> <li>• Description</li> <li>• Extrinsic</li> </ul> <p>This element has the following attributes:</p> <ul style="list-style-type: none"> <li>• Name: This is a mandatory attribute.</li> <li>• startDate: This is an optional attribute.</li> <li>• endDate: This is an optional attribute.</li> </ul>

```

<OrderRequestHeader
  orderDate = "2012-11-19T10:15:00-08:00"
  orderID = "1_2482012_5"
  orderType = "regular"
  type = "new">
  <Total>
    <Money currency="USD">65.00</Money>
  </Total>
  <Modifications>
    <Modification>
      <OriginalPrice>
        <Money currency = "USD">40.00</Money>
      </OriginalPrice>
      <AdditionalCost>
        <Money currency = "USD">10</Money>
      </AdditionalCost>
      <ModificationDetail
        endDate = "2013-11-30T10:15:00-08:00"
        name = "Access Charges"
        startDate = "2012-11-19T10:15:00-08:00">
        <Description xml:lang = "en-US">Access Charges</Description>
      </ModificationDetail>
    </Modification>
  </Modifications>
  </Total>
  <ShipTo>

```

## ItemOut Element

ItemOut elements describe individual ordered items.

This element can also contain the `ItemType` and `parentLineNumber` attributes to specify if the line item is an item group having child line item or an independent line item.

The `parentLineNumber` attribute is a mandatory field and applicable only for a line item with `itemType="item"`. This attribute refers to the line number of the parent line item.

The `ItemType` attribute can contain two values: "composite" to identify an item group, "item" to identify an independent line item. Buyers can group child line items under an item group. For example:

```
<ItemOut lineNumber="1" quantity="4" itemType="composite">
  <ItemID>
    <SupplierPartID>AD1513</SupplierPartID>
    <SupplierPartAuxiliaryID></SupplierPartAuxiliaryID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">30</Money>
    </UnitPrice>
    <Description xml:lang="en">
      Dining Set
    </Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="unspsc">432118</Classification>
  </ItemDetail>
  <Distribution>
    <Accounting name="DistributionCharge">
      <AccountingSegment id="100">
        <Name xml:lang="en">Percentage</Name>
        <Description xml:lang="en">
          Percentage
        </Description>
      </AccountingSegment>
    </Accounting>
    <Charge>
      <Money currency="USD"> 120</Money>
    </Charge>
  </Distribution>
</ItemOut>
<ItemOut lineNumber="2" quantity="4" parentLineNumber="1" itemType="item">
  <ItemID>
    <SupplierPartID>AD1514</SupplierPartID>
    <SupplierPartAuxiliaryID></SupplierPartAuxiliaryID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">2.50</Money>
    </UnitPrice>
    <Description xml:lang="en">
      Fork
    </Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="unspsc">432118</Classification>
  </ItemDetail>
</ItemOut>
```

## Distribution Element

The Distribution element facilitates the buying organization's reconciliation process. It allows requisitioners to split line item charges among multiple groups or accounts. It can contain any relevant accounting code used by a buying organization.

For example, if a line item has a two-way split (for example 60/40), there should be two Distribution elements.

## Accounting Element

The Accounting element groups segments to identify who, or what group, is responsible for a line item's costs. The name attribute identifies the accounting combination. The default value for name is DistributionCharge:

```
<Accounting name="DistributionCharge">
```

The data in the Accounting element is combined to form the actual accounting code, for example, a-b-c where a, b, and c are the IDs of accounting segments.

```
<Distribution>
  <Accounting name="split 60%">
    <AccountingSegment id="2911">
      <Name xml:lang="en">Department</Name>
      <Description xml:lang="en">Development</Description>
    </AccountingSegment>
    <AccountingSegment id="345">
      <Name xml:lang="en">Account</Name>
      <Description xml:lang="en">Computer Accessories</Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">288</Money>
  </Charge>
</Distribution>
<Distribution>
  <Accounting name="split 40%">
    <AccountingSegment id="2911">
      <Name xml:lang="en">Department</Name>
      <Description xml:lang="en">Development</Description>
    </AccountingSegment>
    <AccountingSegment id="123">
      <Name xml:lang="en">Account</Name>
      <Description xml:lang="en">Office Supplies</Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">192</Money>
  </Charge>
</Distribution>
```

In this example, each Distribution/Accounting has two AccountingSegment elements, with the same name patterns. The whole charge is to the same department, but the "Account" segment is split (60% for Computer Accessories and 40% for Office Supplies). The Charge elements for an item should add up to the item's extended price.

In this example, the number of digits in the id attributes are consistent where the Name element is the same. That is, the Department segment has a four digit number and the Account segment has a three digit number. The structure of the accounting segments must be consistent.

Buying organizations do not need to reveal all of their accounting information; just accounting information for this purchase order. Because the segments are named and are not dependant on position in the accounting structure, Buying organizations can provide only the ones they wish to represent.

**Note:** Some Ariba Buyer configurations generate accounting data in awkward ways. For example, they put the split percentage in an AccountingSegment element, resulting in accounting codes such as:

75%-123-45-ABC

Where 123, 45, and ABC make sense, such as division, project, and department, but the 75% is difficult to process.

## Segment Element

**Note:** The Segment element was deprecated and replaced by the AccountingSegment element in cXML 1.2.005. However, suppliers must process both Segment and AccountingSegment, because Ariba Buyer 6, 7, and 8 continues to generate documents that use Segment.

Segment has three attributes: type, id, and description. type identifies the segment relative to others in the Accounting element; id is the unique identifier within a Segment type (for example, the actual accounting code); and description describes the id value.

Segment can contain any relevant accounting code used by a buying organization. Ariba Buyer populates these elements from a line item's Cost Center and Account Name. Examples of possible values are: asset number, billing code, cost center, G/L account, and department.

```
<Distribution>
  <Accounting name="DistributionCharge">
    <Segment type="G/L Account" id="456" description="Travel"/>
    <Segment type="Cost Center" id="23" description="European Projects"/>
  </Accounting>
  <Charge>
    <Money currency="USD">4688.00</Money>
  </Charge>
</Distribution>
```

In the above example Travel describes the G/L Account charge, and corresponds to the 456 account.



## AccountingSegment Element

AccountingSegment has one attribute, id, which is the unique identifier within an AccountingSegment Name (for example, the actual accounting code).

It contains two elements: Name and Description. The Name element identifies the accounting segment relative to others in the Accounting element; and Description describes the id value.

AccountingSegment can contain any relevant accounting code used by a buying organization. Ariba Buyer populates these elements from a line item's Cost Center and Account Name. Examples of possible values are: asset number, billing code, cost center, G/L account, and department.

```
<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">European Projects</Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">4688.00</Money>
  </Charge>
</Distribution>
```

In the above example Travel describes the G/L Account charge, and corresponds to the 456 account.

## Accounting Fields in Ariba Invoice Professional

For purchase orders sent to Invoice and Payment Professional, the expected cXML mappings for accounting fields and split accounting fields are as follows.

```
<ItemOut quantity="2.0" lineNumber="0000000001" requestedDeliveryDate="2009-03-12T10-03-39+01:00">
  <ItemID><SupplierPartID></SupplierPartID></ItemID>
  <ItemDetail><UnitPrice><Money currency="EUR">50.0</Money></UnitPrice>
    <Description xml:lang="en">TESTING<ShortName>TEST_07</ShortName></Description>
    <UnitOfMeasure>EA</UnitOfMeasure><Classification domain="UNSPSC">10401501</Classification>
    <ManufacturerPartID></ManufacturerPartID>

    <Extrinsic name="AccountCategory">K</Extrinsic>
    <Extrinsic name="ItemCategory">M</Extrinsic>
    <Extrinsic name="CompanyCode">0100</Extrinsic>
    <Extrinsic name="PurchaseGroup">0001</Extrinsic>

  </ItemDetail>

  <Distribution>
    <Accounting name="Accounting Information">
      <AccountingSegment id="100">
        <Name xml:lang="en">Percentage</Name>
        <Description xml:lang="en">Percentage</Description>
      </AccountingSegment>
      <AccountingSegment id="0000015300">
        <Name xml:lang="en">GeneralLedger</Name>
        <Description xml:lang="en">ID</Description>
      </AccountingSegment>
      <AccountingSegment id="07801042A0">
        <Name xml:lang="en">CostCenter</Name>
        <Description xml:lang="en">ID</Description>
      </AccountingSegment>
    </Accounting>
    <Charge>
      <Money currency="EUR">50.0</Money>
    </Charge>
  </Distribution>
</ItemOut>
```

## ReceivingTypes in Ariba Invoice Professional

Many ERP systems have a concept of a two-way match between an invoice and the associated orders for items that do not require a receipt. These ERP systems typically include an indicator on order line level if receiving is required for an item or not.

The Extrinsic ReceivingType indicates if a receipt is required or not. The following values are supported for the ReceivingType Extrinsic:

ReceivingType	Result
4	No receipt required - reconcile by quantity. Invoice line items are reconciled to order lines based on the quantity.
10	No receipt required - reconcile by amount. Invoice lines are reconciled based on the value in the Amount field only.
any other value	Triggers three-way invoice exception handling between the invoice, receipts and order.

The expected cXML mapping for ReceivingType is as follows:

```
<ItemOut quantity="2.0" lineNumber="0000000001" requestedDeliveryDate="2009-03-12T10-03-39+01:00">
  <ItemID><SupplierPartID></SupplierPartID></ItemID>
  <ItemDetail><UnitPrice><Money currency="EUR">50.0</Money></UnitPrice>
    <Description xml:lang="en">TESTING<ShortName>TEST_07</ShortName></Description>
    <UnitOfMeasure>EA</UnitOfMeasure><Classification domain="UNSPSC">10401501</Classification>
    <ManufacturerPartID></ManufacturerPartID>
    <Extrinsic name="ReceivingType">4</Extrinsic>
  </ItemDetail>
</ItemOut>
```

If ReceivingType is any other value than 4 or 10, or if ReceivingType is not present in the ItemDetail element, the invoice reconciliation will include matching of receipts to invoices and orders.

### Impact on Ariba Network

Supplier will see the receiving type when reviewing order details, for example:

LINE ITEMS				
Line #	Part ID	Quantity	Unit	Description
• :		1210		
	Charge Amount:			\$100.00USD
OTHER INFORMATION				
	Requester:	adavis		
	ReceivingType:	4		

### Integration with External Systems

Integration with your external ERP system depends on how the external system handles receipt requirements. For example, some ERP systems only indicate on header level if a receipt is required. You will need to build your own business logic to set the receipt required indicator on line level based on your business practices and needs.

In SAP, the **GR Based IV** check box indicates if a receipt is required for invoice verification. You must set up your custom mapping to set the `Extrinsic ReceivingType` to 4 for items where that box is not checked in SAP. If GR Based IV is checked in SAP, you can omit setting the `Extrinsic ReceivingType` entirely, or set `ReceivingType` to any other value than 4 (including just leaving it blank).

## BlanketItemDetail Element

The `BlanketItemDetail` element provides details specific to blanket purchase order items, and should be used only for items in a blanket purchase order. There are three levels of blanket purchase orders:

- **Supplier level:** Contains only one line item defining the terms of the supplier level agreement, and can also contain a maximum amount.
- **Commodity level:** Contains one or more line items that correspond to different commodities that the blanket purchase order captures. This type of line item has supplier, commodity, and limits. `ItemID` in this case should include an empty `SupplierID` tag.
- **Item level:** Contains similar information to the `ItemDetail` element, except that all elements are primarily optional.

The following extrinsics are not visible to the supplier, but are required by Ariba Network for internal implementation purposes:

- `Ariba.availableAmount`—A numeric value sent only on creation of a BPO.
- `Ariba.invoicingAllowed`—A Boolean value used to determine if Ariba Network allows suppliers to create PO-Flip invoices for purchase orders or BPOs. Yes indicates a no release required BPO. No indicates a release required BPO.
- `Ariba.collaborationAllowed`—A Boolean value used to determine if invoices can be created by supplier punch in.

**Note:** The `BlanketItemDetail` element can also contain the quantity-based pricing for a line item. For more information, see “[PriceBasisQuantity Element](#)” on page 126.

## Masking Values in Blanket Purchase Orders

Ariba Network allows buying organizations to mask the quantity, amount, and price while sending a blanket purchase order to the supplier. These values are also not displayed when a supplier creates an invoice for the blanket purchase order containing the masked values.

To mask these values, Ariba Network has introduced the following extrinsic that buying organizations must use to mask values in the **Amount**, **Quantity**, and **Unit Price** fields in a blanket purchase order:

- `hideAmount`—An extrinsic that allows you to mask values for the amount.
- `hideUnitPrice`—An extrinsic that allows you to mask values for the quantity and unit price

A buying organization can always view all the values available in a blanket purchase order from their Ariba Network buyer account. In case the blanket purchase order contains a masked value, the string “The purchase order contains masked values for the supplier.” appears. When the blanket purchase order is sent through cXML or exported through cXML, the masked values display a zero amount in the respective fields.

To enable this feature, configure your adapter (either an Ariba Network Adapter or your own adapter) to send the required extrinsic in the blanket purchase order. For more information, see the *Ariba Network Buyer Administration Guide*.

### Header-Level and Line-Level Masking

The following table displays the fields that are masked when the extrinsic is specified at a header-level or line-item level for a blanket purchase order. If either or both the extrinsic is specified at the header level, the values are masked in the header-level and all the line items of the blanket purchase order. If the extrinsic is specified at the line item level, the values for that particular line item is masked.:

Values Masked in a Blanket Purchase Order						
Extrinsic Enabled						
Header level - Amount and Unit Price	Amount Available	Subtotal	Maximum Amount	Minimum Amount	Status	Price
Header level - Unit Price	-	-	-	-	Status	Price
Header level - Amount	Amount Available	Subtotal	Maximum Amount	Minimum Amount	Status	-
Line item level – Amount and Unit Price	Amount Available	Subtotal	Maximum Amount	Minimum Amount	Status	Price
Line item level – Unit Price	-	-	-	-	Status	Price
Line item level – Amount	Amount Available	Subtotal	Maximum Amount	Minimum Amount	Status	-

### Money currency Attribute

This attribute is an ISO 4217 standard 3-letter currency code and the item amount at the line item level. For example: “USD” = United States Dollar. For more information, see Appendix A, “[Recommended Coding Systems](#).”

```
<Money currency="USD">1.34</Money>
```

### Modifications Element

The ItemDetail element also stores the Modifications element. The Modification element contains details of the allowances and charges applicable for line items at the line-item level. For more information, see “[Modifications Element](#)” on page 116.

### TermsofDelivery Element

This element allows you to specify the terms of delivery in the purchase order or ship notice. For more information, see “[TermsofDelivery Element](#)” on page 114.

## PriceBasisQuantity Element

This element contains the quantity-based pricing for a line item. Quantity-based Pricing is commonly also referred to as Price-Based Quantity or PBQ. Quantity-based pricing allows the unit price of an item to be based on a different price unit quantity than 1. In addition to quantity-based pricing, Unit Conversion Pricing allows unit of measure conversion in the pricing calculation, when the unit of measure on the order differs from the pricing unit of measure.

```
<ItemDetail>
  <UnitPrice>
    <Money currency="USD">0.10</Money>
  </UnitPrice>
  <Description xml:lang="en">ACME Push Pins; Clear; Box Of 20</Description>
  <UnitOfMeasure>EA</UnitOfMeasure>
  <PriceBasisQuantity quantity = "2" conversionFactor = "0.10">
    <UnitOfMeasure>BX</UnitOfMeasure>
    <Description xml:lang = "en">This field specifies that 1 Box is equivalent to 10 EA
      and the unit price is for 2 Boxes</Description>
  </PriceBasisQuantity>
```

**Note:** This element can also be used in blanket purchase orders.

## Tolerances Element

This feature is available to buyers on Ariba Network.

Buyers can specify line item quantity tolerance for individual purchase orders or different line items in a purchase order they send from their ERP systems. The line item quantity tolerance specified in the purchase order takes a higher priority over the line item quantity tolerance specified on the Default Transaction Rules page in the Ariba Network account. The tolerances specified in the purchase order are applied when a supplier creates ship notices against the purchase order.

When you send a changed purchase order containing line item quantity tolerance, this tolerance takes precedence over the tolerance specified in the original purchase order. When the changed order does not specify any line item quantity tolerance value, then the line item quantity specified in the changed purchase order or the tolerance value specified in the Default Transaction Rules page is applied.

If the line item quantity tolerance specified in the changed purchase order is less than the total line item quantity that can be shipped, then Ariba Network does not allow you to ship any more items for the purchase order.

To send a purchase order with line item quantity and unit price tolerance, the OrderRequest cXML document must contain the Tolerances element:

- The ItemOut element must have the following element:
  - Tolerances: To specify the unit price or line item quantity tolerance for a line item.

The `Tolerances` element has the following elements:

Element	Description
<code>QuantityTolerance</code>	<p>The quantity tolerance for a line item. This element has the following elements:</p> <ul style="list-style-type: none"> <li>• <b>Percentage:</b> The percentage for the quantity tolerance.</li> <li>• <b>Value:</b> The quantity number for the quantity tolerance.</li> </ul> <p>You can specify one these elements in the <code>QuantityTolerance</code> element.</p>
<code>PriceTolerance</code>	<p>The price tolerance for a line item. This element has the following elements:</p> <ul style="list-style-type: none"> <li>• <b>Percentage:</b> The percentage for the price tolerance.</li> <li>• <b>Money:</b> The amount and currency for the price tolerance.</li> </ul> <p>You can specify one these elements in the <code>PriceTolerance</code> element.</p>

```

</ItemOut>
  <Tolerances>
    <QuantityTolerance>
      <Percentage percent = "12"/>
    </QuantityTolerance>
    <PriceTolerance>
      <Money currency="USD">200</Money>
    </PriceTolerance>
  </Tolerances>
</ItemOut>
</OrderRequest>

```

For more information, see the *Ariba Network Buyer Administration Guide*.

## Ariba Network Currency Code Mapping

When Ariba Network receives purchase orders, it converts the following obsolete currency codes to up-to-date codes:

Obsolete Code	Current Code
ARP	ARS
ARA	ARS
BGL	BGN
BRE	BRL
BRR	BRL
CAN	CAD
CDN	CAD
IRL	IEP
MXP	MXN
PLZ	PLN
RUB	RUR

If you use obsolete currency codes in catalogs or in Ariba Buyer, change them to up-to-date codes. For amounts that do not specify a currency, Ariba Network assigns USD.

Ariba Network does not perform currency code conversion for test accounts.

## Cancel Orders and Change Orders

---

Cancel orders and change orders are purchase orders sent by buying organizations that modify previously sent purchase orders.

### Cancel and Change Order Requirements

Before a buying organization can send cancel orders or change orders, specific requirements must be met in terms of buying organizations' capabilities, order status, and purchase order matching.

#### Buying Organizations

The use of cancel and change orders is dependent on the business processes of buying organizations. They can enable or disable the generation of these orders in Ariba Buyer and on Ariba Network.

By default, Ariba Network does not allow buying organizations to cancel or change purchase orders that are marked Shipped or Partially Shipped. However, Ariba Network has buyer-configurable rules for allowing cancel and change orders for fully shipped or partially shipped purchase orders. Suppliers can view these rules.

#### Suppliers

Suppliers can configure their Ariba Network accounts to route cancel and change orders differently from new purchase orders.

#### Cancel and Change Orders and Order Status

When Ariba Network receives change orders, it resets order status to the initial state. It discards any previously set status for those orders.

For information about the interaction between these orders and confirmation and ship notices, see [“Confirmations and Cancel and Change Orders”](#) on page 149 and [“Ship Notices and Cancel and Change Orders”](#) on page 153.

#### Purchase Order Matching

Each cancel and change order refers to the previous version of the order through a `DocumentReference` element containing the `payloadID` of the previous order version.

Ariba Network checks whether this reference is valid; if no existing purchase order matches, it rejects the cancel or change order.



## Cancel Orders

Buying organizations use cancel orders to cancel previously sent purchase orders. Ariba Network displays the status of canceled purchase orders as “Obsolete.”

Cancel orders are the same as regular purchase orders, except:

- The OrderRequestHeader element has the attribute type="delete" instead of "new", and the orderVersion attribute of the original order (Ariba Buyer 8.1 and later).
- The OrderRequestHeader DocumentReference element may contain the payloadID of the previous OrderRequest.

The following example shows a cancel order generated by Ariba Buyer.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="985274930687.1374859706.109.7733@zimbuyer.com"
  timestamp="2003-03-28T13:04:04-08:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN01000002792</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="DUNS">
        <Identity>623331717</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="BuyerNetworkUserId">
        <Identity>sysadmin@buyer.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ariba Buyer 8.1</UserAgent>
    </Sender>
  </Header>
  <Request>
    <OrderRequest>
      <OrderRequestHeader
        orderDate="2003-03-28T13:04:04-08:00"
        orderID="PC066"
        type="delete"
        orderVersion="3">
        <DocumentReference payloadID="985274.155.43@zimbuyer.com"/>
        .
        .
        .
        Order details are included by Ariba Buyer,
        but are ignored by Ariba Network
        .
        .
        .
      </OrderRequest>
    </Request>
  </cXML>
```

For information about the interaction between these orders and confirmation and ship notices, see [“Confirmations and Cancel and Change Orders”](#) on page 149 and [“Ship Notices and Cancel and Change Orders”](#) on page 153.

## Change Orders

Buying organizations use change orders to modify previously sent purchase orders. Each updated OrderRequest document *replaces* the previous version of that document. In other words, the previous document version is discarded; updates are not cumulative.

Change orders are the same as regular purchase orders, except:

- The OrderRequestHeader element has the attribute type="update" instead of "new".
- The OrderRequestHeader DocumentReference element may contain the payloadID of the previous OrderRequest.

Ariba Buyer 8.0 and earlier uses a different mechanism to denote order versions than Ariba Buyer 8.1 and later.

### Ariba Buyer 8.0 and Earlier

By default, Ariba Buyer 8.0 and earlier denotes a change order by appending -V*n* to the orderID of an earlier order version. For example, orderID="PC066-V2" is the first revision of the original purchase order PC066. The version number "1" is implied for the original order, and it increments for each new version of an order.

Ariba Buyer 7.0.6 fixed a minor problem in cXML change orders, for more information, see [“Changes Introduced by Ariba Buyer”](#) on page 136.

### Ariba Buyer 8.1 and Later

Ariba Buyer 8.1 and later inserts order version attributes that enable better version tracking by receiving systems. It does not use the -V*n* notation described above.

Ariba Buyer 8.1 and later uses cXML 1.2.009, which has two OrderRequestHeader attributes for indicating change orders:

- **orderVersion**  
Specifies the buyer system order version and is applicable only when the OrderRequest represents a change order request. The version number for the original purchase order should be 1 and incremented by 1 for each subsequent version.

Ariba Network validates the orderVersion number. This number must be unique per purchase order number, and it must be ascending (from the original order to newest change order). Ariba Network does not allow duplicate or descending order versions.

- **isInternalVersion**  
Indicates whether the change is 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.

For example:

```
<OrderRequestHeader
  orderDate="2005-03-28T13:04:04-08:00"
  orderID="D0166"
  type="update"
  orderVersion="3"
  isInternalVersion="yes">
  <DocumentReference payloadID="985274.155.43@zimbuyer.com"/>
```

This order is a change order (type="update"). It is the second revision of the original order (orderVersion="3"). It is an internal version, so changes in it are not relevant to the supplier (isInternalVersion="yes").

## Implementation Hints and Limitations

---

The following topics can help you send or receive purchase orders successfully.

- [Credential Elements](#)
- [Contact Roles](#)
- [Sold To Address and VAT ID](#)
- [Path Routing](#)
- [Non-Catalog Items](#)
- [Attachments](#)
- [PCards and vPayment](#)
- [Blanket Purchase Orders](#)
- [Maximum Document Size](#)
- [Extrinsic for Clickable Links](#)
- [Extrinsic for Legacy Purchase Orders](#)
- [Changes Introduced by Ariba Buyer](#)

### Credential Elements

For information about the credentials used within OrderRequest documents, see “[cXML Credentials](#)” on page 32.

For information about credential limitations of older versions of Ariba Buyer, see “[NetworkID and Older Versions of Ariba Buyer](#)” on page 35.

### Contact Roles

By default, OrderRequest documents use the BillTo and ShipTo elements to specify bill to and ship to addresses. In some cases trading partners might want to list additional addresses.

When buying organizations generate OrderRequest documents, they can use the optional Contact section to specify multiple roles and addresses.

Ariba Network recognizes the following Contact role values in purchase orders:

Contact Role Value	Displays as
"postDelivery"	Post Address
"cargoDelivery"	Cargo Address
"companyDelivery"	Receiving
"privateEndUser"	Requisitioner Address
"defaultDelivery"	Personal Deliver To Address

Contact Role Value	Displays as
"buyerCorporate"	Buyer Headquarter Address
"supplierCorporate"	Supplier Address
"soldTo"	Customer

## Sold To Address and VAT ID

Ariba Network allows buyers to configure a list of Sold To Addresses and associate them with one or more VAT IDs. When a supplier creates a PO invoice or a non-PO invoice, Ariba Network displays the list and allows the supplier to choose an address. If a VAT ID is associated with the address, Ariba Network displays it on the invoice.

As a best practice, Ariba recommends that buyers using PO Automation or Ariba Invoice Professional include the Sold To address and VATID in the purchase order cXML to eliminate the need for the supplier to provide this information on the invoice. To add the Sold To address, use the `Contact` role element. To add the VAT ID, add the Extrinsic `buyervatID` to the `OrderRequestHeader`:

```
<Extrinsic name="buyerVatID">SE123456789087</Extrinsic>
```

## Path Routing

Supplier aggregators or marketplace hosts that want to receive copies of purchase orders, but are not the primary recipients, should use cXML path routing. These organizations are called *copy organizations*.

Copy organizations should perform the following steps to prepare for path routing:

- 1 They must tell buying organizations to add path routing information to all `PunchOutSetupRequest` and `OrderRequest` documents. For example:

```
<Path>
  <Node type="copy">
    <Credential domain="NetworkID">
      <Identity>AN01000000111</Identity>
    </Credential>
  </Node>
</Path>
```

- 2 They must add the `CopyRequest` transaction to their cXML profile.

When Ariba Network 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. If the organization does not have a cXML profile or if `CopyRequest` does not appear in its profile, Ariba Network attempts to send the copy `OrderRequest` to the URL the organization entered in the "URL to send your orders" field in the Method for Receiving Orders page in the Configuration area of its account.

## Non-Catalog Items

Non-catalog (ad-hoc) purchase orders contain items described manually by requisitioners, not items selected from electronic catalogs. Requisitioners enter these items on an ad-hoc basis or because they could not find them in electronic catalogs.

Often, non-catalog items do not have part numbers. Purchase orders containing non-catalog items usually require special validation and processing, so not all suppliers accept them.

### Special Characteristics of Non-Catalog Orders

Non-catalog items have the following user-entered information:

- Supplier Part ID (optional)
- Description
- Commodity Code
- Quantity
- Unit of Measure
- Unit Price

Suppliers must carefully validate this information, because it comes from users, not from electronic catalogs.

### Enabling or Disabling Non-Catalog Orders

Buying organizations determine whether users can order non-catalog items. Suppliers should contact their customers to determine whether they will receive orders containing non-catalog items.

Suppliers can configure their Ariba Network accounts to route orders that contain non-catalog items differently from regular orders, or to reject those orders.

### Detecting Non-Catalog Orders

To detect non-catalog items, Ariba Network examines incoming orders and looks for either:

- The `isAdHoc` flag (used by Ariba Buyer 8.0 and later)
- Supplier Part IDs that are blank or set to "not available"

Ariba Buyer does not use the `isAdHoc` flag for regular catalog items. That is, it does not include `isAdHoc="false"` for regular catalog items.

### Breaking Requisitions into Multiple OrderRequests

Ariba Buyer should be configured to break requisitions that contain both catalog and non-catalog items into separate purchase orders. 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.

## Attachments

Requisitioners sometimes clarify purchase orders with associated memos or drawings. Ariba Buyer can be configured to allow users to attach files of any type to purchase orders before sending them to Ariba Network. It attaches these files to purchase orders using a MIME envelope as described in the *cXML User's Guide*.

Following describes the order attachment implementation on Ariba Network:

- In the attachment part of the MIME file, Content-Length is optional while Content-Disposition is required with the attachment name.
- In the attachment part of the MIME file, if non-ASCII characters are used in the filename of the Content-Disposition header, the document attached describes the encoding method used by Ariba Network.
- In the cXML part of the MIME file, the attachment tag is optional for orders with attachments. If the attachment tags in the header or line item are indeed provided with the corresponding Content-IDs of the attachments, Ariba Network maintains the Content-IDs of the attachment in the MIME file forwarded to the supplier. A cXML-enabled supplier can use the attachment tags to link the attachments to the corresponding line items.
- In the main POST header of the MIME file, Content-Type can be a multipart/related or multipart/mixed header.

### Attachments on Ariba Network

Ariba Network stores attachments for retrieval by both trading partners. Users can retrieve these files by logging on to their Ariba Network accounts. Ariba Network can also forward attachments to suppliers through email or cXML post.

Attachments expire 18 months after Ariba Network receives them. Expired attachments are not available online.

### Attachment File Names

Ariba Network supports attachments with file names in all supported character sets, including Chinese, Japanese, and Korean. Ariba Network conforms to Multipurpose Internet Mail Extensions (MIME) standards by encoding non-ASCII filenames according to Internet Engineer Task Force (IETF) Request for Comment (RFC) 2047. Filenames in Asian languages can contain up to 15 characters, and filenames in Latin-1 languages can contain up to 21 eight-bit characters or 70 seven-bit characters.

### AttachmentOnline Extrinsic

If suppliers configure their Ariba Network accounts to send orders but not attachments, Ariba Network adds an Extrinsic element named "AttachmentOnline" to the OrderRequestHeader to indicate the existence of attachments. For example:

```
<Extrinsic name="AttachmentOnline">  
  https://service.ariba.com/ad/orderDetail?poID=1234&anp=Ariba  
</Extrinsic>
```

Suppliers can use the URL in this Extrinsic to manually log in and view the purchase order. They can then click on the listed attachments to view or download them.

Ariba Network adds this Extrinsic only for suppliers that have selected “Send order and leave attachment online” in their order routing configuration.

## PCards and vPayment

Buying organizations can include the PCard element in the OrderRequestHeader element to provide purchasing card information to suppliers. Suppliers can charge PCards through their own point-of-sale systems. Buying organizations also use the PCard element for vPayment.

### vPayment

vPayment is a service provided by GE Corporate Payment Services. Ariba Network requests a virtual credit card number from GE for each purchase order. GE responds with a virtual credit card number, which Ariba Network adds to the purchase order before routing it to the supplier. The supplier charges the credit card through a point-of-sale system. Virtual credit card numbers are good for only a specific amount and a limited time.

vPayment purchase orders contain a special Extrinsic element named `paymentServiceName` and an empty PCard element. For example:

```
<OrderRequestHeader orderID="D0123" orderDate="2005-02-01T11:19:34-10:00" type="new">
...
  <Extrinsic name="paymentServiceName">GEvPayment</Extrinsic>
...
  <Payment>
    <PCard number="" expiration=""/>
  </Payment>
</OrderRequestHeader>
```

Buying organizations must configure Ariba Buyer to include these elements. Ariba Buyer should leave the PCard element empty for vPayment, populate it for regular credit card orders, and leave it off entirely for purchase orders that must be invoiced.

vPayment requires the OrderRequestHeader orderID attribute value to be shorter than 16 characters.

## Blanket Purchase Orders

Ariba Procurement and Invoicing Solutions support only item-level BPOs from external ERP systems. When Ariba Procurement and Invoicing Solutions receive any other type of BPO from an ERP system, the BPO is accepted by the Ariba solution, and the contract type specified in the incoming BPO cXML file (in the ContractType extrinsic) is reflected accurately in the Ariba solution. However, the BPO does not function according to its contract type. For example, the pricing terms are shown as item-level pricing terms even if the incoming BPO is a supplier contract.

## Maximum Document Size

The maximum size of cXML documents is 4 MB or 3000 line items. You can add attachments up to 10 MB. The 10 MB limit applies to the *total size* of all attachments associated with the document (for example, a purchase order with attachments or an invoice with attachment), which means that the total size for documents with attachments is 14 MB.

If you attempt to add an attachment that exceeds this limit, Ariba Network displays a warning and does not allow you to add the attachment.

Ariba Network can display purchase orders and invoices that contain up to 1,000 line items or are up to 1 MB in size in suppliers' inboxes and buying organizations' outboxes.

## Extrinsic for Clickable Links

Ariba Network can render URLs as clickable links in online purchase orders and invoices. These URLs are produced by Extrinsic elements of the form:

```
<Extrinsic name="anyname">
  <URL name="click me">http://www.bigcompany.com/info</URL>
</Extrinsic>
```

The previous example produces the following online text:

anyname:[click me](http://www.bigcompany.com/info)

The words “click me” are underlined and are clickable, and the link destination is <http://www.bigcompany.com/info>. If the URL element has no name attribute, Ariba Network displays the URL and makes it clickable.

## Extrinsic for Legacy Purchase Orders

You can upload previously fulfilled purchase orders so that your suppliers can invoice you through Ariba Network. Ariba Network routes these legacy purchase orders to your supplier's online Inbox, even if the use another routing method. The legacy purchase order contains the comment, “This purchase order has already been fulfilled.”

To differentiate legacy purchase orders from new purchase orders, include an Extrinsic element named AribaNetwork.LegacyOrders in the order request header:

```
<OrderRequestHeader orderDate="1975-03-28T13:04:04-08:00" orderID="PC066">
  ...
  <Extrinsic name="AribaNetwork.LegacyOrders"></Extrinsic>
  ..
</OrderRequestHeader>
```

Ariba Network displays a warning message informing you that the order has already been fulfilled on the online purchase order page, and sets the status of the order to “Sent/Unconfirmed.” Do not enable GE vPayment for legacy purchase orders. If vPayment is enabled, Ariba Network attempts to authorize the purchase order and obtain credit card information from GE.

## Changes Introduced by Ariba Buyer

Ariba Buyer 7.0.6 fixed two problems in OrderRequest documents:

- Earlier versions of Ariba Buyer had a minor problems in cXML change orders: lineNumber values for items in change orders and original orders were different. Ariba Buyer 7.0.6 and later maintains the original line item numbers in change orders, which complies with the cXML specification.



- Earlier versions of Ariba Buyer used `requestedDeliveryDate` with an unneeded time value, for example, `requestedDeliveryDate="2005-02-01T00:00:00+09:00"`. Ariba Buyer 7.0.6 and later includes only the date value for this attribute, for example, `requestedDeliveryDate="2005-02-01"`.

---

## Response Documents

---

After Ariba Network forwards an OrderRequest to a supplier, the supplier must respond with a cXML Response in the same HTTP connection within five minutes. The response serves as an acknowledgment.

The Response is not an agreement to ship any items, but simply an acknowledgement that the OrderRequest was received, was authenticated successfully, and that it validates correctly against the DTD. To communicate detailed order status, the supplier can later send ConfirmationRequest documents. For more information, see Chapter 8, “[Order Confirmations and Ship Notices](#).”

### Example Response Document

The following example shows a Response, which is sent in the same HTTP connection as the OrderRequest document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="994994" xml:lang="en" timestamp="2002-03-12T18:39:09-08:00">
  <Response>
    <Status code="200" text="OK"/>
  </Response>
</cXML>
```

There is no reference to the OrderRequest document, because the Response is sent synchronously in the same HTTP connection used to transmit the OrderRequest document.

If Ariba Network does not receive a Response within five minutes, it resends the OrderRequest with the same payloadID. After 10 retries (once per hour), Ariba Network holds the transaction and labels it “Failed.”

There is no element named “OrderResponse,” because the only data that needs to be sent back to Ariba Network is the Status part of the Response.

## Purchase Order Acknowledgments

Suppliers must acknowledge that they received cXML and EDI purchase orders from Ariba Network and that they are syntactically correct. Acknowledgments set purchase order routing status from “sent” to “acknowledged” or “failed.”

Ariba Buyer 8.1 (or later) uses the cXML get pending transaction to retrieve StatusUpdateRequest documents containing receive purchase order acknowledgments from Ariba Network. Users of earlier versions of Ariba Buyer must log in to Ariba Network to see purchase order acknowledgements.

Ariba Network’s purchase order acknowledgment requirements depend on suppliers’ routing methods. The following table lists the period during which suppliers must send acknowledgments and whether Ariba Network resends purchase orders if it receives no acknowledgments:

<b>Routing Method</b>	<b>Supplier Acknowledgment</b>	<b>Time out for Acknowledgment</b>	<b>Ariba Network Resends PO?</b>
cXML	cXML Response in the same HTTP connection as the purchase order	5 minutes	Yes, 10 times (once per hour)
EDI	X12 997 or EDIFACT CONTRL	72 hours	No
Fax	“confirmation of receipt” message from the fax machine	Immediately after sending fax	Yes, 10 times (once per hour)
Email	None	None	No
Online	None	None	No

Email and online routing methods do not support acknowledgments; Ariba Network immediately sets the routing status of these documents to “sent.” If suppliers’ mailboxes reject email purchase orders, Ariba Network sets the routing status of these documents to “failed.”

Suppliers cannot acknowledge failed purchase orders. They must first log in to their Ariba Network accounts, resend the failed purchase orders, and return positive acknowledgments.

## Other Ways to Acknowledge Purchase Orders

Suppliers can acknowledge “sent” or “failed” purchase orders by sending invoices against them. Suppliers can use any invoicing method: online, cXML, or EDI. Ariba Network considers invoices to be acknowledgments and sets purchase order status to “acknowledged.”

If the buying organization downloads purchase order status, Ariba Network sends a cXML StatusUpdateRequest to set purchase order status to “acknowledged.”

## Blanket Purchase Order Acknowledgements

After a blanket purchase order (BPO) is sent to the supplier, updates are sent through Ariba Network about any subsequent “open” or “close” operation on the order, including the available amount. The buying organization sends updates to the supplier through Ariba Network about the amount available on the BPO as a result of invoice reconciliation or the order process.

If the BPO is in a hierarchy and is configured to accumulate spend to a master BPO, then an update for the available amount on the master BPO is also sent through Ariba Network. The `BlanketOrderStatusUpdateRequest` element, used to send status updates, contains the following information:

- `open`: Sent with the available amount when a BPO is opened.
- `close`: Sent with the available amount when a BPO is closed.
- `Update`: Sent when the available amount of a BPO changes as a result of invoice reconciliation or the purchase order process.
- `Comments`: Any comments associated with the “open” or “close” operation for the supplier are included.
- `AvailableAmount`: The amount remaining on the BPO that can still be released or invoiced against.

## Duplicate Documents

If the supplier does not send a Response document within the allotted time discussed above, Ariba Network resends the original document. The duplicate documents have the same `payloadID` value as the original document; it is the supplier’s responsibility to detect duplicates.

If the supplier detects a duplicate, it should discard previous documents and return a Response document. The status in the response should be the same as in the original response.

---

## Chapter 8 Order Confirmations and Ship Notices

- “[Overview](#)” on page 141
- “[ConfirmationRequest Documents](#)” on page 144
- “[ShipNoticeRequest Documents](#)” on page 151
- “[Attachments](#)” on page 155
- “[Serial Numbers and Asset Tags](#)” on page 156

### Overview

---

Suppliers can update purchase order status either manually through their Ariba Network accounts or by submitting cXML documents to Ariba Network.

The cXML ConfirmationRequest and ShipNoticeRequest documents enable suppliers to programmatically inform buying organizations of status changes to purchase orders through Ariba Network. These transactions provide a more detailed item level confirmation and ship notification than the simple acknowledgement of orders provided by StatusUpdateRequest.

This section is an overview of order confirmations and ship notices:

- [Visibility of Order Status](#)
- [Ariba Network Account Configuration](#)
- [Transaction Flow](#)

### Visibility of Order Status

Buying organizations, requisitioners, and suppliers can view order status.

#### Buying Organizations

Authorized procurement personnel within buying organizations can log in to Ariba Network and view order status.

#### Requisitioners

Requisitioners (end users) have access through their procurement application to order status.

The order status location viewed by requisitioners depends on the procurement application:

Application	Order Status Location
Ariba Buyer 7.04 through 8.0	Order status is stored only on Ariba Network, not within Ariba Buyer. Requisitioners view order status by automatically performing order status PunchOut to Ariba Network.
Ariba Buyer 8.1 and later	Order status can optionally be downloaded periodically by Ariba Buyer from Ariba Network. Requisitioners view order status within Ariba Buyer.

## Suppliers

Authorized order-management personnel within supplier organizations can log in to Ariba Network and view order status.

## Ariba Network Account Configuration

Both buying organizations and suppliers perform configuration to enable order status updates.

### Buying Organizations

Ariba Buyer 8.1 or later can download ConfirmationRequest and ShipNoticeRequest documents. Buying organizations must first contact Ariba Network Support to have their accounts enabled to download these document types.

### Suppliers

Before sending ConfirmationRequest or ShipNoticeRequest documents, suppliers must first configure their Ariba Network accounts to set their Preferred Method of Sending Documents. Suppliers can set order status and shipping information manually or automatically; both methods cannot be used simultaneously.

To set programmatic order status updates:

- 1 Suppliers log on to their Ariba Network accounts, go to the Configuration area, and set **Preferred Method of Sending Response Documents** to cXML.
- 2 They determine the URL for posting ConfirmationRequest and ShipNoticeRequest documents by using the Profile transaction. For more information, see Chapter 4, “[Profile Transaction](#).”

These suppliers can still manually update order status by temporarily changing the Preferred Method of Sending Response Documents from cXML to Online. However, after a purchase order has been updated manually, it cannot subsequently be updated through cXML.

Suppliers must also configure their backend system to only use the Ariba Network ANID of their customer in the To Credentials while sending ConfirmationRequest or ShipNoticeRequest documents.

Additionally, they must ensure that they do not use the following in the To Credentials while sending an order confirmation or ship notice through cXML:

- The Network ID (ANID) of your supplier account
- The Network ID (ANID) of Ariba Network - AN01000000001

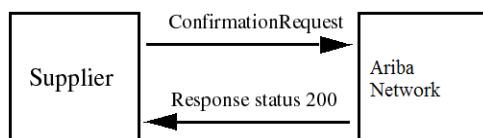
## Transaction Flow

The transaction flow for `ConfirmatioRequest` and `ShipNoticeRequest` depends on which version of Ariba Buyer the buying organization uses.

### Ariba Buyer 7.04 Through 8.0

These applications do not download order confirmations or ship notices; therefore documents travel only between the supplier and Ariba Network.

- 1 The supplier sends a `ConfirmatioRequest` or `ShipNoticeRequest`.
- 2 If the document is valid, Ariba Network responds with status code 200/OK in the Response.

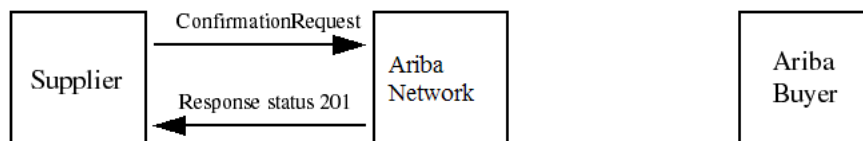


Buying organization can view order status on Ariba Network. For more information, see “[Visibility of Order Status](#)” on page 141.

### Ariba Buyer 8.1 and Later

This application can optionally download order confirmations and ship notices using the `GetPending` transaction. It sends a status update to Ariba Network, which forwards it to the supplier.

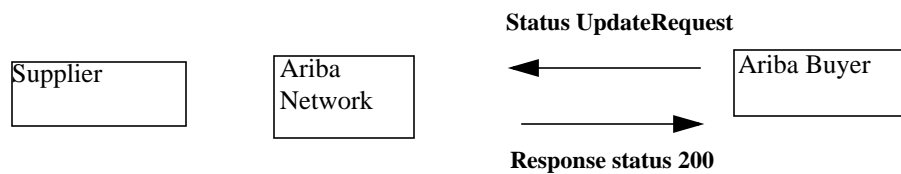
- 1 The supplier sends a `ConfirmatioRequest` or `ShipNoticeRequest`. If the document is valid, Ariba Network responds with status code 201/Accepted in the Response.



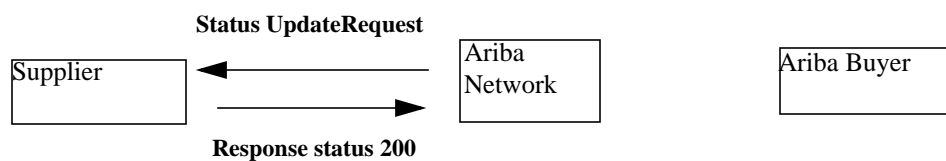
- 2 Some time later, Ariba Buyer uses the `GetPending` transaction to download the `ConfirmatioRequest` or `ShipNoticeRequest`.



- 3 If the document content passes the buying organization’s internal processing, Ariba Buyer sends a `StatusUpdateRequest` to Ariba Network with status code 200/OK. Ariba Network responds with a status code 200/OK in the Response.



- 4 If the supplier's cXML profile indicates the supplier supports StatusUpdateRequest, Ariba Network forwards the StatusUpdateRequest to the supplier. The supplier responds with status code 200/OK in the Response.



## Credentials

For the credential values for order confirmations and ship notices, see “**Required Credentials**” on page 37.

## ConfirmationRequest Documents

Ariba Network receives ConfirmationRequest documents and updates the status of previously ordered items. Requisitioners using Ariba Buyer can view the status of a ordered items. The complete order and status history is visible to both the buying organization and the supplier.

Suppliers can send multiple ConfirmationRequest documents for an order, and each ConfirmationRequest refers to one, not multiple, OrderRequest documents.

**Note:** The ConfirmationRequest document notifies a customer that the supplier might or might not change the customer's purchase order. It does not imply that these changes are acceptable to the customer or that the order can automatically be fulfilled in accordance with such changes by the supplier. How buying organizations interpret ConfirmationRequest documents depends on business processes that exist outside of Ariba Buyer and Ariba Network.

## ConfirmationRequest Element

ConfirmationRequest documents provide detailed status updates of OrderRequest documents.

ConfirmationRequest documents contains three elements: ConfirmationHeader, OrderReference, and an optional ConfirmationItem (included only if type="detail" or "except").



## ConfirmationHeader Element

Optional elements allowed in ConfirmationHeader (for display only) are:

- Total
- Shipping
- Tax
- Contact
- Comments
- Hazard

ConfirmationHeader has the following attributes and elements.

confirmID	Any internal number used by the supplier.
type	<p>Confirmation request type for transaction. Can be: accept, reject, except, detail, backordered, requestToPay, or replace.</p> <ul style="list-style-type: none"> <li>• type="accept", "reject", and "except" can be applied to the entire purchase order. Multiple confirmations sent for a purchase order can reference a line item only one time, but can contain multiple statuses for that line item. Multiple confirmations are acceptable only if type="accept", "except", or "reject".</li> <li>• type="detail" updates portions of a purchase order, such as prices, quantities, delivery dates, reject portions, tax, and shipping information. Multiple ConfirmationRequest documents of this type can refer to a single purchase order, but cannot refer to common line items. At least one of the following elements UnitPrice, Shipping, Tax, ItemIn (item substitution), DeliveryDate, or Contact must be present.</li> <li>• type="backordered" sets the entire order to backordered status. The supplier does not have the items in stock, but will ship them when they are available.</li> <li>• type="replace" replaces all items in the purchase order with new items. ConfirmationItem elements must contain ConfirmationStatus elements only of type "detail" and must include an ItemIn element. The procurement application should cancel and replace the original purchase order or generate a change purchase order that contains these new line items.</li> <li>• type="allDetail" is not supported by Ariba Network.</li> </ul>
operation	<p>Specifies whether the document is new or an update. Can be "new" or "update".</p> <ul style="list-style-type: none"> <li>• operation="new" sets initial status of items or entire orders.</li> <li>• operation="update" revises item or order status set by previously sent order confirmations. Updates are not cumulative; each one must list all quantities of a line item (each document replaces the previous one). Ariba Network rejects updates that do not list all line items previously updated.</li> </ul>
noticeDate	For operation="update", use noticeDate to specify the date of the update (not the original confirmation time).
Total	<p>Total dollar value of all line items in the transaction.</p> <p>The Total element also contains the Modifications element which stores any modification to the original price or shipping price of the item. This element can store a set of one or more Modification elements.</p> <p>The Modification element contains details of the allowances and charges applicable at the header-level. For more information, see “<a href="#">Modifications Element</a>” on page 116.</p>
Shipping	Total shipping charge.
Tax	Total tax.

All values should match the OrderRequest document unless ConfirmationItem updates unit price or quantity.

The type and operation attributes in ConfirmationHeader determine the purpose of the document.

### OrderReference Element

orderID	(Optional) The purchase order number assigned by Ariba Buyer.
orderDate	(Optional) Date the purchase order was sent by the buying organization.
DocumentReference	<p>Exists at the header level if the ConfirmationRequest is the initial response to an OrderRequest and also at the line item level if the ConfirmationHeader operation="update".</p> <p>For operation="new", the DocumentReference at the header level must refer to the purchase order. For operation="update" or "delete", the DocumentReference at the header level must refer to the previous ConfirmationRequest.</p>

### ConfirmationItem Element

lineNumber	Reference line item number from the OrderRequest document.
quantity	Either original quantity if going to confirm in full or updated quantity with type set appropriately.
Comments	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. Valid terms for status information include, shipped and invalid.

## ConfirmationStatus Element

ConfirmationStatus attributes include:

quantity	Sum must match the quantity in the ConfirmationItem.
type	<p>Action taken by supplier for line item number identified. Can be: accept, detail, reject, backordered, requestToPay, or unknown.</p> <ul style="list-style-type: none"> <li>type="accept" accepts particular line item.</li> <li>type="detail" accepts portion of line item 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.</li> <li>type="reject" rejects this portion of the line item.</li> <li>type="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.</li> <li>type="unknown" resets the status to the default state. The item status displays on Ariba Network as if no confirmations had been sent.</li> </ul> <p>type="allDetail" is not supported by Ariba Network.</p>
shipmentDate	Date and time this shipment is expected to leave the supplier. Required if type is accept, allDetail, or detail.
deliveryDate	<p>Date and time this shipment is expected to arrive. Required if type is accept, allDetail, or detail.</p> <p>Do not include if its value matches requestedDeliveryDate (if any) in the purchase order.</p> <p>Displayed as "Est. Delivery Date" within Ariba Network.</p>

UnitPrice and Tax are forbidden if ConfirmationStatus type="unknown".

If the purchase order or blanket purchase order has quantity-based pricing for a line item, the ConfirmationStatus element contains the PriceBasisQuantity element. For more information, see ["PriceBasisQuantity Element"](#) on page 126.

In earlier releases of Ariba Network, suppliers could set backordered status only by setting type="unknown" and including a Comments element containing the text "backordered". This method of setting backordered status is no longer supported.

The ConfirmationStatus element also contains the Modifications element. The Modification element contains details of the allowances and charges applicable at the line-item level. For more information, see ["Modifications Element"](#) on page 116.

The following example shows a Modification element:

```

<ConfirmationItem quantity = "3" lineNumber = "1">
  <UnitOfMeasure>DZ</UnitOfMeasure>
  <ConfirmationStatus type = "unknown" quantity = "1">
    <UnitOfMeasure>DZ</UnitOfMeasure>
  </ConfirmationStatus>
  <ConfirmationStatus type = "accept" quantity = "2">
    <UnitOfMeasure>DZ</UnitOfMeasure>
  <UnitPrice>
    <Money currency = "USD">47</Money>
    <Modifications>
      <Modification>
        <OriginalPrice>
          <Money currency = "USD">45.00</Money>
        </OriginalPrice>
        <AdditionalDeduction>
          <DeductionAmount>
            <Money currency = "USD">5.00</Money>
          </DeductionAmount>
        </AdditionalDeduction>
        <ModificationDetail
          name = "Allowance"
          startDate = "2012-11-19T10:15:00-08:00"
          endDate = "2013-11-30T10:15:00-08:00">
          <Description xml:lang = "en-US">Contract Allowance<
            Description>
          </ModificationDetail>
        </Modification>
      </Modifications>
    </UnitPrice>
    <Tax>
      <Money currency = "USD">7.0</Money>
      <Description xml:lang = "en">Tax</Description>
      <TaxDetail category = "Other">
        <TaxAmount>
          <Money currency = "USD">5.0</Money>
        </TaxAmount>
      </TaxDetail>
      <TaxDetail category = "QST">
        <TaxAmount>
          <Money currency = "USD">2.0</Money>
        </TaxAmount>
      </TaxDetail>
    </Tax>
  </ConfirmationStatus>
</ConfirmationItem>

```

## Implementation Hints and Limitations

The following topics can help suppliers to send order confirmations successfully.

- [Item Substitutions](#)
- [Confirmations and Cancel and Change Orders](#)
- [Order Confirmation Updates](#)

### Item Substitutions

Ariba Network allows suppliers to send ConfirmationRequest documents that substitute ordered items. cXML enables item substitutions by allowing ItemIn elements within ConfirmationStatus elements.

Ariba Network displays only Supplier Part ID, Unit Price, Shipping, and Tax information online. Additional information that might be in ConfirmationRequest documents, such as Supplier Part Auxiliary ID, Manufacturer Part ID, Description, and Comments do not display.

### Confirmations and Cancel and Change Orders

Suppliers must understand how Ariba Network processes order confirmations for cancelled and changed orders.

Cancelled orders:

- Ariba Network rejects ConfirmationRequest documents for cancelled purchase orders.
- Ariba Network allows no updates to ConfirmationRequest documents that were received prior to purchase order cancellation.
- Suppliers cannot send ConfirmationRequest documents to acknowledge order cancellations.

Changed orders:

- Ariba Network does not allow suppliers to reject original orders after it receives changed orders.
- Any ConfirmationRequest posted to order prior to change are persisted, but they are not automatically transferred or linked to the new change order.

A change order appears in a supplier's inboxes as a separate order, with a version indicator (for example -V2). ConfirmationRequest documents should refer to the change order's DocumentReference. Ariba Network sets the status of the original order to "Obsoleted."

Suppliers cannot reject the "changing" of original orders, but they can reject, accept, or change the items in the new order.

### Order Confirmation Updates

Suppliers can update previously set order status by sending additional ConfirmationRequest documents. To set initial status of line items, use operation="new". To revise line items, use operation="update" and list all other line items.

Each ConfirmationRequest document *replaces* the previous version of that document. In other words, the previous document version is discarded; ConfirmationRequest documents are not cumulative. Each updated document must list all line items that appeared in the previous version.

Subsequent ConfirmationRequest documents must use new payloadID and NoticeDate attributes, but use the same confirmID attribute as the first order confirmation. These documents can have one or two DocumentReference elements:

- If operation="new", reference the purchase order with a DocumentReference element at the line item level.
- If operation="update", reference the previous ConfirmationRequest with a DocumentReference at the header level and the purchase order with a DocumentReference element at the header level.

## Example ConfirmationRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/Fulfill.dtd">
<cXML payloadID="March222001_1154am" timestamp="2001-03-22 11:55:38 -0700">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN0100001234</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000006789</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN0100001234</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ecommerce Supplier</UserAgent>
    </Sender>
  </Header>
  <Request>
    <ConfirmationRequest>
      <ConfirmationHeader operation="new" confirmID="1234"
        type="accept"
        noticeDate="2001-02-28T10:07:00-08:00">
        <Total>
          <Money currency="USD">139.95</Money>
        </Total>
        <Shipping>
          <Money currency="USD">5.00</Money>
          <Description xml:lang="en-US">FedEx 2-day</Description>
        </Shipping>
        <Tax>
          <Money currency="USD">12.45</Money>
          <Description xml:lang="en-US">CA State Tax</Description>
        </Tax>
        <Comments>Header Comments</Comments>
      </ConfirmationHeader>
      <OrderReference orderID="PC066">
        <DocumentReference
          payloadID="985274930687.1374859706.109.7733@zimbuyer.com">
        </DocumentReference>
      </OrderReference>
      <ConfirmationItem lineNumber="1" quantity="1">
        <UnitOfMeasure>EA</UnitOfMeasure>
        <ConfirmationStatus quantity="1" type="accept" shipmentDate="2001-03-30T08:39:29-08:00">
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Comments>Order has been accepted. Will ship ASAP</Comments>
        </ConfirmationStatus>
      </ConfirmationItem>
    </ConfirmationRequest>
  </Request>
</cXML>
```

```
        </ConfirmationStatus>
      </ConfirmationItem>
    </ConfirmationRequest>
  </Request>
</cXML>
```

## ShipNoticeRequest Documents

---

Suppliers can set the shipping status of purchase order line items by sending cXML ShipNoticeRequest documents.

ShipNoticeRequest documents are advance ship notices that provide detailed shipping information for items in purchase orders. Suppliers can send these documents before or after shipping items. Ariba Network receives ShipNoticeRequest documents, validates them against the cXML DTDs, and applies the information in them to the affected line items.

Buying organizations can see updated line item status. For more information, see “[Visibility of Order Status](#)” on page 141.

## Implementation Hints and Limitations

The following topics can help suppliers to send ship notices successfully.

- [ShipNoticeRequest payloadID Attribute](#)
- [DocumentReference payloadID Attribute](#)
- [shipmentID Attribute](#)
- [ServiceLevel Element](#)
- [ShipControl Element](#)
- [PackageIdentification Element](#)
- [Packaging Element](#)
- [ShipNoticeItem Element](#)
- [TermsofDelivery Element](#)
- [Ship Notices and Cancel and Change Orders](#)
- [Limitations](#)

### ShipNoticeRequest payloadID Attribute

The payloadID in the header of the ShipNoticeRequest must be unique. Refer to the *cXML User's Guide* at [www.cXML.org](http://www.cXML.org) for payloadID format.

### DocumentReference payloadID Attribute

Similar to the ConfirmationRequest transaction, the DocumentReference payloadID must match the payloadID of the OrderRequest. Ariba Network uses this value to match documents, not the orderID attribute. The orderID is an optional attribute.

**shipmentID Attribute**

The shipmentID attribute is required.

**ServiceLevel Element**

The ServiceLevel element is optional. If you include it, one or more language-specific strings are required.

**ShipControl Element**

The ShipControl element describes travel segments under control of a single carrier. It provides tracking information buying organizations can use to retrieve information about the shipment. The Route element describes transit segments (modes) under a carrier's control.

If suppliers specify a carrier name recognized by Ariba Network, it displays a hyperlink for tracking the shipment at the carrier's website. Ariba Network uses carrier names in the companyName domain:

```
<ShipControl>
  <CarrierIdentifier domain="companyName">FedEx</CarrierIdentifier>
  <ShipmentIdentifier>8202 8261 1194</ShipmentIdentifier>
</ShipControl>
```

Ariba Network recognizes the following carrier names:

- Airborne Express
- Consolidated Freightways
- DHL
- EGL Eagle Global Logistics
- EmeryWorldwide
- FedEx
- Menlo/IBM
- Purolator
- Purolator Courier
- Roadway Express
- UAL Cargo
- UPS
- US Postal Service
- Velocity Express
- Yellow Freight

The ShipControl element also stores the ShipmentIdentifier element to store the tracking number of the shipment.

**PackageIdentification Element**

This element resides within the ShipControl element. Ship notices have just one ShipmentIdentifier per carrier. The PackageIdentification contains an inclusive range of numbers that appears on portions (for example, boxes, pallets, or skids) of the shipment.



**Packaging Element**

This element resides within the `ShipNoticeItem` element. It contains both packaging codes (zero or more locale-specific strings) and dimensions for the package of an item. The only supported dimensions are length, width, height, weight and volume.

**ShipNoticeItem Element**

This element stores the quantity and line number of the line item.

**TransportInformation Element**

For more information, see “[TransportInformation Element](#)” on page 112

**TermsofDelivery Element**

You can add the `TermsofDelivery` element to the `ShipNoticeItem` element to specify terms of delivery at the line-item level. For more information, see “[TermsofDelivery Element](#)” on page 114.

**Ship Notices and Cancel and Change Orders**

If purchase orders are marked as Shipped or Partially Shipped, Ariba Network rejects any cancel or change orders for them.

**Limitations**

Observe the following Ariba Network limitations when using `ShipNoticeRequest`.

- `operation="update"` and `operation="delete"` are not currently supported.
- Multiple `ShipControl` elements in a `ShipNoticeRequest` are not currently supported. Send a separate `ShipNoticeRequest` for each shipment.
- Multiple `ShipNoticePortion` elements in a `ShipNoticeRequest` are not currently supported. If a shipment contains items from multiple orders, a `ShipNoticeRequest` must be sent for each order.

## Example ShipNoticeRequest

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/Fulfill.dtd">
<cXML payloadID="1233444-2004@premier.supplier.com"
xml:lang="en-CA" timestamp="2001-10-14T08:39:29-08:00">
  <Header>
    <From>
      <Credential domain="DUNS">
        <Identity>123456789</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000002792</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="DUNS">
        <Identity>123456789</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ecommerce Supplier</UserAgent>
    </Sender>
  </Header>
  <Request>
    <ShipNoticeRequest>
      <ShipNoticeHeader shipmentType="planned" shipmentID="S89823-123" operation="new"
        noticeDate="2001-10-14"
        shipmentDate="2001-10-14T08:30:19-08:00"
        deliveryDate="2001-10-18T09:00:00-08:00">
        <Contact role="shipFrom">
          <Name xml:lang="en-CA">XYZ Warehouse Inc.</Name>
          <PostalAddress>
            <Street>9966 Mercury Liquid Center</Street>
            <City>Sunnyvale</City>
            <State>CA</State>
            <PostalCode>94086</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Phone>
            <TelephoneNumber>
              <CountryCode isoCountryCode="CA">1</CountryCode>
              <AreaOrCityCode>800</AreaOrCityCode>
              <Number>5555555</Number>
            </TelephoneNumber>
          </Phone>
        </Contact>
        <Comments xml:lang="en-CA">Single shipment.</Comments>
        <Comments type="ReasonForShipment" xml:lang="en-US">Low availability at
          warehouse</Comments>
        <Comments type="TransitDirection" xml:lang="en-US">East</Comments>
        <Comments xml:lang="en-US">As per the terms</Comments>
        <TermsOfDelivery>
          <TermsOfDeliveryCode value="PriceCondition"/>
          <ShippingPaymentMethod value="AdvanceCollect"/>
          <TransportTerms value="Other">Contract Terms</TransportTerms>
          <Address>
            <Name xml:lang="en-US">SNV</Name>
            <PostalAddress name="default">
              <Street>123 Anystreet</Street>
              <City>Sunnyvale</City>
              <State>AL</State>
              <PostalCode>35762</PostalCode>
            </PostalAddress>
          </Address>
        </TermsOfDelivery>
      </ShipNoticeHeader>
    </ShipNoticeRequest>
  </Request>
</cXML>

```

```

        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Address>
    <Comments xml:lang="en-US" type="Transport">As per the contract</Comments>
    <Comments xml:lang="en-US" type="TermsOfDelivery">Delivery at the
      doorstep</Comments></Terms Of Delivery>
  </ShipNoticeHeader>
  <ShipControl>
    <CarrierIdentifier domain="SCAC">FDE</CarrierIdentifier>
    <CarrierIdentifier domain="companyName">FedEx</CarrierIdentifier>
    <ShipmentIdentifier>8202 8261 3294</ShipmentIdentifier>
  </ShipControl>
  <ShipNoticePortion>
    <OrderReference orderID="PC066">
      <DocumentReference
        payloadID="985274930687.1374859706.109.7733@zimbuyer.com">
      </DocumentReference>
    </OrderReference>
    <ShipNoticeItem quantity="1" lineNumber="1">
      <UnitOfMeasure>EA</UnitOfMeasure>
    </ShipNoticeItem>
  </ShipNoticePortion>
</ShipNoticeRequest>
</Request>
</cXML>

```

## Attachments

Suppliers sometimes clarify order confirmation and ship notices with associated memos, faxes, or drawings. They can use Ariba Network or cXML to attach files to these documents using a MIME envelope as described in the *cXML User's Guide*. Buying organizations can configure Ariba Network to include attachments when forwarding order confirmations and ship notices to Ariba Buyer 8.2 or later.

### Attachments on Ariba Network

Ariba Network stores attachments for retrieval by both trading partners. Users can retrieve these files by logging on to their Ariba Network accounts.

Attachments expire 18 months after Ariba Network receives them. Expired attachments are not available online.

### Attachment File Names

Ariba Network supports attachments with file names in all supported character sets, including Chinese, Japanese, and Korean. Ariba Network encodes non-ASCII filenames according to Internet Engineer Task Force (IETF) Request for Comment (RFC) 2047. Filenames in Asian languages can contain up to 15 characters, and filenames in Latin-1 languages can contain up to 21 eight-bit characters or 70 seven-bit characters.

### AttachmentOnline Extrinsic

If buying organizations configure their Ariba Network accounts to send order confirmations and ship notices but not attachments, Ariba Network adds an Extrinsic element named “AttachmentOnline” to the ConfirmationHeader to indicate the existence of an attachment. For example:

```
<Extrinsic name="AttachmentOnline">  
  https://service.ariba.com/ad/shipnoticeDetail?poID=1234&anp=Ariba  
</Extrinsic>
```

Buying organizations can use the URL in this Extrinsic to manually log in and view the order confirmation or ship notice. They can then click on the listed attachments to view or download them.

Ariba Network adds this Extrinsic only for buying organizations that have clicked “Leave attachments online” in their transaction configuration.

## Serial Numbers and Asset Tags

---

Suppliers can provide serial numbers to buying organizations in ship notices. Alternatively, buying organizations can pre-assign an asset number range to each supplier. For example, the supplier might offer to print bar codes for the number range and assign them to the products shipped. If suppliers using PO-Flip enter asset tag or serial number information, Ariba Network places it in the correct element.

Suppliers generate AssetInfo tags in the ShipNoticeRequest document. For example:

```
<ShipNoticeItem lineNumber="10" quantity="2.000">  
  <AssetInfo tagNumber="111" serialNumber="SN5187" location="Bob's Office"/>  
  <AssetInfo tagNumber="112" serialNumber="SN5188" location="Anthony's Office"/>  
  <UnitOfMeasure>EA</UnitOfMeasure>  
</ShipNoticeItem>
```

---

## Chapter 9 Invoices and Scheduled Payments

- “Basic Invoice Functionality” on page 157
- “Purchase Order Matching” on page 158
- “Invoicing Business Rules” on page 159
- “Implementation Hints” on page 161
- “Transmission to Buying Organizations” on page 180
- “Status and Cancel Invoices” on page 181
- “Example Invoices” on page 182
- “Scheduled Payments” on page 203
- “End Points Integration with Ariba Network Adapters” on page 206

This chapter describes the behavior of Ariba Network and procurement applications when they receive cXML InvoiceDetailRequest documents.

### Basic Invoice Functionality

---

Suppliers can generate invoices through the following methods:

- Manual online invoice generator (PO-Flip)
- cXML InvoiceDetailRequest documents
- EDI ANSI X12 810 or EDIFACT INVOIC documents

Suppliers can use a combination of these methods without making any configuration changes on Ariba Network; however, to avoid confusion, they should use only one method per purchase order.

Ariba Network matches each invoice with its corresponding purchase order and it compares them with the buying organizations’ invoicing business rules.

Invoices appear in suppliers’ online outboxes and in buying organizations’ online inboxes. Both trading partners can view invoices and their up-to-date status as they move through the invoice lifecycle. Invoices provide links to associated documents such as purchase orders, master agreements, and credit/debit memos.

Suppliers can send debit memos through cXML, but Ariba Network does not support creating them online through the **Create Invoice** pages.

Ariba Network can route invoices from suppliers to Ariba Buyer (7.1 and later). It can also route invoice status (StatusUpdateRequest documents) from those organizations to suppliers. Ariba Buyer performs invoice reconciliation to match charges in invoices to items on purchase orders.

---

## Purchase Order Matching

---

Invoices refer to pre-existing purchase orders, master agreements, or credit/debit memos. These pre-existing documents might reside on Ariba Network, or they might be external documents, which are documents that were not routed through Ariba Network. Both Ariba Network and Ariba Buyer attempt to find pre-existing documents by searching for them.

Suppliers insert one of the following elements in invoices to reference pre-existing purchase orders or master agreements:

- **OrderReference**—(preferred method) A **DocumentReference** element containing the purchase order/master agreement **payloadId**.
- **OrderIDInfo**—The purchase order/master agreement **orderId** and its **orderDate**.

### Matching on Ariba Network

Ariba Network performs document matching immediately after cXML validation to find pre-existing purchase orders or master agreements.

- If invoices use **payload ID**, but that value does not match purchase orders on Ariba Network, Ariba Network considers the purchase orders to be external.
- If invoices use only **orderId** and **orderDate**, Ariba Network attempts to match based on the purchase order number and optional order date.
- If invoices use only **orderId** and **orderDate**, and those values match multiple purchase orders, Ariba Network rejects the invoices.
- If invoices use both **payload ID** and **orderId**, Ariba Network makes sure these values match the same purchase orders. If they do not match, Ariba Network considers the purchase orders to be external, even if **payloadID** matches a purchase order.
- If invoices match an obsoleted purchase order, Ariba Network rejects the invoices.

### Successful Matches

If Ariba Network successfully matches invoices with purchase orders or master agreements, it creates online hyperlinks to them available to both trading partners.

Ariba Network then tests invoice contents against the buying organization's business rules (see "**Invoicing Business Rules**" on page 159.)

### Unsuccessful Matches

If Ariba Network cannot match a purchase order, it treats the invoice as belonging to an external purchase order, which is any purchase order that was not routed through Ariba Network.

Buying organizations configure their Ariba Network accounts to either allow or reject invoices against external purchase orders. For more information, see "**Invoicing Business Rules**" on page 159.

## Matching in Ariba Buyer

By default, Ariba Buyer finds pre-existing purchase orders or master agreements by using the `OrderReference` payloadID from invoices. If invoices do not contain that value, it matches by `OrderIDInfo` (purchase order number).

If Ariba Buyer cannot find referenced purchase orders, it generates *unmatched invoice exceptions*. These exceptions list invoice details and allow requisitioners to manually match invoices to purchase orders. If requisitioners cannot find matches, they can reject the invoices, and Ariba Buyer sends cXML `StatusUpdateRequest` documents to Ariba Network to set invoice status to “rejected.”

Buying organizations can configure Ariba Buyer to automatically reject invoices that do not match purchase orders. They can also customize the logic that Ariba Buyer uses for document matching.

## Invoicing Business Rules

---

When buying organizations enable their Ariba Network accounts for invoicing, they specify invoicing business rules. These rules specify basic requirements of their invoicing process, for example:

- Allow invoices that contain detailed service information for both goods and service items. These are invoices with `InvoiceDetailServiceItem` elements (introduced in cXML 1.2.009). For service line items, the name attribute can be `IsShippingServiceItem`, `IsSpecialHandlingServiceItem`, `ServiceLocation` (contains contact element) or an arbitrary text string.
- Allow invoices against external (non-Ariba Network) purchase orders.
- Allow invoices against PCard/credit card orders.
- Allow invoices to change currency, unit price, unit of measure, part number, ship to, or bill to.
- Allow invoices to have additional quantities or line items.
- Allow users to view invoices on Ariba Network from your ERP application.

Each buying organization can set these business rules differently.

When Ariba Network receives invoices, it tests their contents against these business rules. Ariba Network does not route invoices that fail to pass. It sends email notifications to suppliers (not to buying organizations) and it does not send cXML status documents.

## Rules that Affect PO-Flip Only

The following business rules affect only PO-Flip invoices (invoices generated online), not invoices generated through cXML, EDI, or CSV:

- Allow a change in Ship To Information
- Allow a change in Bill To Information
- Allow direct entry of sales tax

## Supplier Visibility of Business Rules

Suppliers can view the invoicing business rule settings of their customers through their Ariba Network accounts. Suppliers should understand their customers' capabilities before generating invoices.

To view a customer's invoicing business rules, suppliers go to the Buyers area of their Ariba Network accounts and click the name of the customer. If invoicing rules are not listed, that customer does not accept invoices through Ariba Network.

## Numeric Validation

Ariba Network does not perform any numeric validation on line-items, totals, or costs within cXML invoices.

Ariba Network does not validate math errors. For example, if a cXML invoice lists unit price \$50 and quantity 5, Ariba Network will not check to see if the subtotal is \$250. That discrepancy will not generate a validation error.

Similarly, Ariba Network does not perform numeric validation of amounts in `ConfirmationRequest` documents that refer to invoices. In particular, Ariba Network does not check the `PartialAmount` element. `PartialAmount` enables buying organizations to specify different amounts paid than the amounts specified in invoices.



## Implementation Hints

Ariba applications have specific requirements for invoice contents. This section discusses the following aspects of invoices:

- eSignatures
- IdReference Element
- Contact Roles
- Conversion of Segment to AccountingSegment
- Invoice payloadID Attribute
- SerialNumber Elements
- PaymentTerm Element
- Remittance IDs
- Credits
- Inspection Date Attribute
- PriceBasisQuantity Element
- Tax Exchange Rate Extrinsic
- Tax Invoice Extrinsic
- Tax Detail Category
- Tax on Shipping and Special Handling
- Withholding Tax Support
- Fields for Countries with a VAT System
- Attachments
- Maximum Document Size
- Extrinsic Elements
- Field Lengths
- PDF Invoice Copy Attachments

### eSignatures

Digital signatures (eSignatures) provide an electronic method of authenticating the creators of documents and of ensuring content integrity. They are also used to meet country-specific requirements, including VAT.

Digital signatures enable buying organizations and suppliers to prove that an invoice is unaltered, depending on local laws. If a digitally signed invoice is changed after it is signed, the digital signature is invalid.

Ariba Network can apply a digital signature to invoices generated online, through cXML, EDI, or CSV. Ariba Network compares the originating and destination countries on the invoice to determine whether the invoice should be digitally signed. If both countries require digital signatures, Ariba Network digitally signs the invoice. Some countries require digital signatures for both the originating and the destination countries.

To determine the originating and destination countries, Ariba Network checks the InvoiceDetail cXML for the following information in the order given. If the country cannot be determined, Ariba Network does not digitally sign the invoice.

Order	Originating Country Determined by...	Destination Country Determined by...
1	Tax representative's VAT ID prefix	Buyer's VAT ID prefix
2	Tax representative's country code	"shipTo" element country code (header level)

Order	Originating Country Determined by...	Destination Country Determined by...
3	Supplier's VAT ID prefix	"shipTo" element country code (line item)
4	"From" element country code	

The customer's currency is determined by the country specified in the Ship To field.

Digitally signed invoices are compatible with any version of Ariba Buyer that accepts invoices. Versions of Ariba Buyer before 8.2 accept signed invoices but do not authenticate digital signatures. European buying organizations should allocate disk space to store the additional 3-4 kilobytes per invoice for digital signatures.

On the Invoice Review page, Ariba Network specifies that the invoice will be digitally signed. Ariba Network also signs updated and cancel invoices that you generate. Ariba Network does not apply a digital signature for paper invoices, ICS invoices (invoices sent from an invoice conversion service provider), or invoices marked with the `invoiceSubmissionMethod` extrinsic.

For information on the current invoicing certificate requirements for several European countries, see [knowledge.ariba.com/Ariba\\_eSigning\\_CountryMatrix](http://knowledge.ariba.com/Ariba_eSigning_CountryMatrix).

## Supplier Self-Signed Invoices

Beginning with AN49, Ariba Network supports self-signed invoices. Suppliers who implement their own digital signature solution (via cXML) can create self-signed invoices. This process assumes that the supplier is the issuer of the invoice and can validate on behalf of the buyer. Therefore, the supplier must use the signature validation method specified for canonicalization as described in the W3C specifications *before* signing the invoice.

Suppliers are free to use their established business logic for determining when and how to eSign as follows:

- Which invoices to sign (in some case the supplier may choose not to apply an eSignature at all)
- What country combinations of origin and destination to use
- Which certificates to use for signing invoices

### Limitations

Buyers should fully understand the supplier's process for eSigning and validating on their behalf. Also, if the invoice is not canonicalized before signing, any third party (e.g., Infomosaic, Apache) validation of the signature will fail. Ariba's digital signature feature utilizes TrustWeaver as the third-party digital signature ASP.

**Note:** Using Apache is preferred for signature validation (and not Microsoft.NET). Refer to <http://www.w3.org/TR/xml-c14n> and [http://en.wikipedia.org/wiki/Canonical\\_XML](http://en.wikipedia.org/wiki/Canonical_XML) for more information about XML canonicalization.

For more information about using digital signatures with invoices, see the *Ariba Network Buyer Administration Guide*.

## ICS Invoices

The `invoiceSubmissionMethod` extrinsic indicates the means used to submit the invoice. Ariba Network uses this extrinsic to assign a value to invoices created online for reporting and uses it to determine whether to digitally sign invoices. Invoice conversion service providers use the extrinsic with the value `PaperViaICS`. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.020/InvoiceDetail.dtd">
...
<InvoiceDetailRequestHeader invoiceDate="2008-08-11T01:34:42+01:00" invoiceID="280064482"
operation="new" purpose="standard" ...>
  <Extrinsic name="invoiceSubmissionMethod">PaperViaICS</Extrinsic>
</InvoiceDetailRequestHeader>
...
```

Other values for `invoiceSubmissionMethod` are: `Online`, `EDI`, `cXML`, `PaperInvoice`, `SupplierPunchIn`, and `CSVUpload`.

When an invoice is received from an invoice conversion service provider, the **Submit Method** column in the user interface in the Ariba Network account and the `invoiceSubmissionMethod` element in the cXML contains the following values:

- If the invoice is sent with a value in the `invoiceSubmissionMethod` extrinsic, Ariba Network retains this value and displays it in the cXML and in the Ariba Network account.
- If the invoice is sent without a value in the `invoiceSubmissionMethod` extrinsic, then the `invoiceSubmissionMethod` in the cXML shows “PaperViaICS” and the **Submit Method** column in the Ariba Network account displays, “ICS Paper Invoice.”

**Note:** The `invoiceSubmissionMethod` extrinsic supersedes the legacy `convertedInvoiceData` extrinsic, which indicated how the invoice was created. Use `invoiceSubmissionMethod` instead.

For detailed information on ICS invoices and the invoice conversion process, see the *Ariba Guide to Invoice Conversion*.

## Other Source Documents

The extrinsic invoiceSourceDocument is used for reporting purposes to show the source document referenced in the invoice. The value is one of the following strings: PurchaseOrder, SalesOrder, Contract, ExternalPurchaseOrder, or Time\_ExpenseSheet. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.020/InvoiceDetail.dtd">
...
<InvoiceDetailRequestHeader invoiceDate="2008-08-11T01:34:42+01:00" invoiceID="280064482"
operation="new" purpose="standard" invoiceOrigin="supplier">
  <Extrinsic name="invoiceSourceDocument">Contract</Extrinsic>
  <Extrinsic name="invoiceSubmissionMethod">SupplierPunchin</Extrinsic>
</InvoiceDetailRequestHeader>
...
```

When suppliers send a non-PO invoice or credit memo through cXML or EDI, they must ensure that a value is specified in the invoiceSourceDocument extrinsic. This extrinsic value is used to show the source document referenced in the invoice.

- If the buyer has enabled the rule, “**Require suppliers to provide order information**” in the Ariba Network buyer account, a value is required in one of the Order Information fields for invoices sent through cXML or EDI. The value entered in the invoiceSourceDocument extrinsic is displayed in the **Source Doc** field.
- If the buyer has not enabled the rule, then the value in the Order Information fields are optional and can be left blank for invoices sent through cXML or EDI. The cXML is generated with an empty orderID attribute. However, suppliers should ensure that they enter the string, “NoOrderInformation” in the invoiceSourceDocument extrinsic when there is no value in the Order Information fields. When this string is not provided in the extrinsic, the **Source Doc** field is left blank.

```
<Extrinsic name="invoiceSourceDocument">NoOrderInformation</Extrinsic>
```

**Note:** The string, “NoOrderInformation” is case-sensitive and must be entered exactly as mentioned here

## IdReference Element

Trading partners should understand how to generate and use the IdReference value within InvoicePartner elements.

For example, in most cases, the domains accountID and bankRoutingID should be used as a pair with Contact role remitTo. For example:

```
<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" />
</InvoicePartner>
```

For some trading relationships, the accountID might not refer to a bank account ID. It might be an ID assigned to the supplier by the buying organization.

Additional details about IdReference domains:

- accountPayableID—Buying organization's vendor number for the supplier.
- accountReceivableID—Supplier's customer number for the buying organization.
- vatID—Value Added Tax identifier for organizations that comply with VAT requirements.
- gstID—Goods and Services Tax identifier for Canadian organizations.
- stateTaxID and provincialTaxID—Interchangeable, representing only different globalized descriptions.

## ACH Information

ACH information provides details on where to send payments. For example:

```
<InvoicePartner>
  <Contact role="receivingBank">
    <Name xml:lang="en-US">Wells Fargo Bank</Name>
  </Contact>
  <IdReference identifier="121042882" domain="abaRoutingNumber" />
  <IdReference identifier="Workchairs Savings" domain="accountName" />
  <IdReference identifier="34567890" domain="BankaccountID" />
  <IdReference identifier="Savings" domain="accountType" />
  <IdReference identifier="Sunnyvale" domain="branchName" />
</InvoicePartner>
```

## Wire Information

Wire information provides details on where to send payments. For example:

```
<InvoicePartner>
  <Contact role="WirereceivingBank">
    <Name xml:lang="en-US">BankAmerica</Name>
  </Contact>
  <IdReference identifier="121042882" domain="swiftID" />
  <IdReference identifier="12345678" domain="ibanID" />
  <IdReference identifier="Workchairs Savings" domain="accountName" />
  <IdReference identifier="123456" domain="bankAccountID" />
  <IdReference identifier="Savings" domain="accountType" />
  <IdReference identifier="Sunnyvale" domain="branchName" />
</InvoicePartner>
```

## Contact Roles

Invoices can have the following Contact role values:

- from—Supplier's name and address.
- soldTo—Buyer's name, email, and address.
- billTo—Buyer's name and address.
- remitTo—Supplier's address.
- shipFrom—Supplier's shipping address, with email address, if any.
- shipTo—Buyer's name and address.

The Contact element within InvoiceDetailShipping has the following roles: shipFrom and shipTo. Both roles are required.

The role="remitTo" can optionally have an addressID attribute.

## Conversion of Segment to AccountingSegment

When suppliers generate invoices manually through their Ariba Network accounts, Ariba Network transforms any `Segment` elements from the purchase orders into `AccountingSegment` elements for the invoices.

The `Segment` element was deprecated and replaced by the `AccountingSegment` element in cXML 1.2.005.

## Invoice payloadID Attribute

The `payloadID` in the invoice document header must be a unique value for all documents. If the `payloadID` is not unique, the invoice is not generated and no error message appears. Use the guidelines for formulating `payloadID` in the *cXML User's Guide*.

Invoices should also contain the `payloadID` of the corresponding purchase order or master agreement. For more information, see “[Purchase Order Matching](#)” on page 158.

## SerialNumber Elements

Suppliers can uniquely identify items being invoiced by specifying their serial numbers. Suppliers can insert one or more `SerialNumber` elements within `InvoiceDetailItemReference` elements:

```
<SerialNumber>45993823469876</SerialNumber>
<SerialNumber>45993823469877</SerialNumber>
<SerialNumber>45993823469878</SerialNumber>
```

The `serialNumber` attribute was deprecated in cXML 1.2.009 and replaced by the `SerialNumber` element. Ariba Network displays both `serialNumber` attributes and `SerialNumber` elements.

## PaymentTerm Element

The `PaymentTerm` element allows suppliers to specify discounts or penalties based on payment date. Payment terms can come from corresponding purchase orders.

Buying organizations can allow or disallow changes to payment terms on Ariba Network. Regardless of whether they allow suppliers to edit payment terms, Ariba Network accepts changes in payment terms for purchase orders that do not specify these terms.

Standard invoices and debit memos can have payment terms; however, Ariba Network rejects credit memos and line-item credit memos that have payment terms.

## Remittance IDs

Buying organizations might require suppliers to consistently identify remittance addresses with a unique ID value. Suppliers should populate the Contact addressID attribute with that ID.

For example:

```
<InvoicePartner>
  <Contact role="remitTo" addressID="1234">
    <Name xml:lang="en">Joan Bill</Name>
    <PostalAddress name="billing department">
      <DeliverTo>Joan Bill</DeliverTo>
      <Street>16 Castro Street</Street>
      <City>Mountain View</City>
      <State>CA</State>
      <PostalCode>95035</PostalCode>
      <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
    .
    .
    .
  </Contact>
  .
  .
  .
</InvoicePartner>
```

The buying organization can assign a unique ID value for each supplier address. The supplier must use the ID appropriate for the buying organization.

## Credits

Suppliers can issue credits to their customers by sending credit memos and line-item credit memos. To issue a credit, suppliers should send a header-level credit memo or a line-item credit memo with a negative amount. Your customers must configure their Ariba Network account to accept line-item credit memos. If your customers do not configure their accounts, Ariba Network treats a line-item credit memo as a standard invoice.

The following example shows the InvoiceDetailRequestHeader element of a line-item credit memo:

```
<InvoiceDetailRequest>
  <InvoiceDetailRequestHeader invoiceID="CRD012042"
    purpose="lineLevelCreditMemo" operation="new"
    invoiceDate="2001-12-04T18:00:00-07:00">
    <InvoiceDetailHeaderIndicator isHeaderInvoice="no"/>
    <InvoiceDetailLineIndicator/>
    <Comments xml:lang="en-US">
      Buyer ordered 5 Computer Video Cables and 3 Wireless Keyboards. We invoiced the
      customer for 8 Cables by mistake.
    </Comments>
  </InvoiceDetailRequestHeader>
</InvoiceDetailRequest>
```

**Note:** In order for the line-item credit memo to contain a link to the original invoice, suppliers need to provide the DocumentReference element with the PayloadID. For example, <DocumentReference payloadID="<PreviousInvoicePayloadId>" />. For non-cXML suppliers, InvoiceIDInfo is an acceptable alternative. The InvoiceDetailRequestHeader element example shows only the invoiceID provided in the credit memo.

## Inspection Date Attribute

Some countries, such as Japan, mandate that invoices specify when the transfer of products or the delivery of services occurs.

Suppliers specify inspection dates using the `inspectionDate` attribute, which can be used with the `InvoiceDetailItem`, `InvoiceDetailServiceItem`, and `InvoiceDetailOrderSummary` elements. For example:

```
<InvoiceDetailItem
  invoiceLineNumber="1"
  quantity="5"
  inspectionDate="2004-01-01T00:01:23+00:00">
```

Ariba Network automatically adds this attribute when suppliers generate invoices manually in their Ariba Network accounts if both trading partners' addresses are in Japan.

## PriceBasisQuantity Element

This element contains the quantity-based pricing for a line item. For more information, see [“PriceBasisQuantity Element”](#) on page 126.

This element is also available in invoices with `InvoiceDetailServiceItem` elements.

## itemType Attribute

The `itemType` attribute specifies if the line item is a grouped item having child items or an independent line item. It can contain two values: “`composite`” to identify an item group or “`item`” to identify an independent line item. In `InvoiceDetailItem` and `InvoiceDetailServiceItem`, the attribute `parentInvoiceLineNumber` refers to the line number of the parent line item. Buyers can group child line items under an item group.

For example:

```
<InvoiceDetailItem invoiceLineNumber="1" quantity="10" itemType="composite">
  <UnitOfMeasure>EA</UnitOfMeasure>
  <UnitPrice><Money currency="USD">11.11</Money></UnitPrice>
  <InvoiceDetailItemReference lineNumber="1">
    <ItemID><SupplierPartID>KYBD101E</SupplierPartID></ItemID>
    <Description xml:lang="en-US">This is an item group.
      </Description>
  </InvoiceDetailItemReference>
  <SubtotalAmount><Money currency="USD">111.10</Money></SubtotalAmount>
  <GrossAmount><Money currency="USD">111.10</Money></GrossAmount>
  <NetAmount><Money currency="USD">111.10</Money></NetAmount>
</InvoiceDetailItem>
<InvoiceDetailItem invoiceLineNumber="2" quantity="5" parentInvoiceLineNumber="1">
  <UnitOfMeasure>EA</UnitOfMeasure>
  <UnitPrice><Money currency="USD">11.11</Money></UnitPrice>
  <InvoiceDetailItemReference lineNumber="1001">
    <ItemID><SupplierPartID>KYBD101E</SupplierPartID></ItemID>
    <Description xml:lang="en-US">This is a child line</Description>
  </InvoiceDetailItemReference>
  <SubtotalAmount><Money currency="USD">55.55</Money></SubtotalAmount>
  <GrossAmount><Money currency="USD">55.55</Money></GrossAmount>
  <NetAmount><Money currency="USD">55.55</Money></NetAmount>
</InvoiceDetailItem>
```



## Tax Exchange Rate Extrinsic

Ariba Network uses exchange rates from Bloomberg to calculate tax amounts. Some countries, such as Spain, require exchange rates to be retrieved from Spain's officially published rates. To accommodate this requirement, Ariba Network allows suppliers to enter exchange rates instead of automatically converting tax amounts to the specified currency.

Suppliers can indicate tax exchange rates by inserting the following Extrinsic element in the InvoiceDetailRequestHeader element:

```
<Extrinsic name="taxExchangeRate">1.2</Extrinsic>
```

Ariba Network allows a tax to be specified in one currency only for online invoicing. The currency of the buying organization is calculated according to the country specified in the Ship To address. The online invoice form does not allow suppliers to create invoice lines shipping to multiple countries if this rule is turned on. The supplier is prompted to enter the exchange rate if the currency of tax amounts is different from the currency of the Ship To address country.

Ariba Network also validates that the incoming cXML has tax amounts specified in the currency of the Ship To address. If the tax information is specified at the line item level, Ariba Network validates the tax information specified in the currency of the Ship To information at the line item level, if applicable. Otherwise it uses the Ship To information from the invoice header for validation.

If the tax information is specified at the invoice header level, and there are multiple Ship To countries at the line item level, Ariba Network does not validate the local tax currencies.

## Tax Invoice Extrinsic

Some countries, such as Australia, mandate an indicator on invoices that contain tax charges in order for buying organizations to receive tax credits.

Suppliers can indicate tax invoices by inserting the following Extrinsic element in the InvoiceDetailRequestHeader element:

```
<Extrinsic name="TaxInvoice">This is a tax invoice</Extrinsic>
```

The name must be "TaxInvoice", but the element's contents are ignored. Ariba Network automatically adds this element when suppliers generate invoices manually in their Ariba Network accounts if they have an Australian address.

## Tax Detail Category

The Generate Invoice page on Ariba Network enables suppliers to specify values for the TaxDetail category attribute. For example:

```
<TaxDetail purpose="tax" category="gst">
```

Possible values for invoices generated on Ariba Network are:

category Attribute	Description
sales	Sales Tax
vat	Value Added Tax

category Attribute	Description
gst	Goods and Services Tax
pst	Provincial Sales Tax
qst	Quebec Sales Tax
hst	Harmonized Sales Tax
usage	Usage Tax
other	Other Tax Category

## Tax on Shipping and Special Handling

In some locales, suppliers must indicate tax on shipping or special handling. They add an additional `TaxDetail` element and indicate whether it is line, shipping, or special handling tax with `purpose="tax"`, `purpose="shippingTax"`, or `purpose="specialHandlingTax"`. They indicate tax category independently (for example, `category="sales"` or `category="vat"`).

There can be only one shipping amount and one special handling amount per line, so there is no need to indicate which shipping or special handling amount a `TaxDetail` element applies to. Suppliers can specify individual taxes only if they specify both tax and shipping inline, not in the header. If they specify shipping inline but tax in the header, they can still have one `TaxDetail` element in the header that applies to the total of all shipping charges.

The following example specifies line tax, special handling tax, and shipping tax for a single invoice item:

```
<InvoiceDetailItem invoiceLineNumber="1" quantity="10">
  <UnitOfMeasure>EA</UnitOfMeasure>
  <UnitPrice>
    <Money currency="USD">500</Money>
  </UnitPrice>
  ...
  <SubtotalAmount>
    <Money currency="USD">5000</Money>
  </SubtotalAmount>
  <Tax>
    <!-- Tax breakdown:
      5000 * 0.05 = 250 (5% tax on $5000 goods)
      40 * 0.05 = 2 (5% tax on $40 shipping)
      80 * 0.05 = 4 (5% tax on $80 special handling)
      ==
      256 total tax
    -->
    <Money currency="USD">256</Money>
    <TaxDetail purpose="tax" category="sales" percentageRate="5">
      <TaxableAmount>
        <Money currency="USD">5000</Money>
      </TaxableAmount>
      <TaxAmount>
        <Money currency="USD">250</Money>
      </TaxAmount>
      <Description xml:lang="en">Sales tax on goods</Description>
    </TaxDetail>
    <TaxDetail purpose="shippingTax" category="sales" percentageRate="5">
      <TaxableAmount>
        <Money currency="USD">40</Money>
      </TaxableAmount>
```

```

    <TaxAmount>
      <Money currency="USD">2</Money>
    </TaxAmount>
    <Description xml:lang="en">Sales tax on shipping</Description>
  </TaxDetail>
  <TaxDetail purpose="specialHandlingTax" category="sales" percentageRate="5">
    <TaxableAmount>
      <Money currency="USD">80</Money>
    </TaxableAmount>
    <TaxAmount>
      <Money currency="USD">4</Money>
    </TaxAmount>
    <Description xml:lang="en">Sales tax on special handling</Description>
  </TaxDetail>
</Tax>
<InvoiceDetailLineSpecialHandling>
  <Money currency="USD">80</Money>
</InvoiceDetailLineSpecialHandling>
<InvoiceDetailLineShipping>
  <InvoiceDetailShipping>
...
    </InvoiceDetailShipping>
    <Money currency="USD">40</Money>
  </InvoiceDetailLineShipping>
...
  <!-- Gross breakdown:
    5000 subtotal
    + 40 shipping
    + 80 special handling
    + 256 all taxes
    ====
    5376 gross
  -->
  <GrossAmount>
    <Money currency="USD">5376</Money>
  </GrossAmount>
...
</InvoiceDetailItem>

```

As a best practice, use negative numbering (such as “-1,” “-2”) for `invoiceLineNumber` in `InvoiceDetailServiceItem` elements for shipping and special handling charges. Use “0” for `lineNumber` in `InvoiceDetailServiceItemReference` elements for the corresponding `InvoiceDetailServiceItem` for shipping and special handling charges.

If buying organizations use Ariba Network Adapter, they might specify line-item tax in purchase orders and require suppliers to use that tax information in invoices. Suppliers can provide header-level special handling tax or shipping tax information in the `InvoiceDetailSummary` element:

```

<InvoiceDetailSummary>
...
  <TaxDetail purpose="shippingTax" category="sales" percentageRate="6">
    <TaxAmount>
      <Money currency="USD">344</Money>
    </TaxAmount>
    <Description xml:lang="en-US">Sales tax on shipping</Description>
  </TaxDetail>
</Tax>
...
</InvoiceDetailSummary>

```

## Withholding Tax Support

Ariba Network allows suppliers and buyers to specify the withholding tax amount at the line level for invoices for services or goods. This type of tax is paid by the buying organization to the state or government tax authorities on behalf of the supplier.

Two extrinsic elements can be added to the cXML Tax element:

- withholdingTax (for total withholding taxes), and
- withholdingTaxTotal (for total taxes minus withholding tax)

The Money element in Tax element is used to capture totals for all tax amounts and is not used for invoice display. The use of negative rates ensures backward compatibility of existing cXML applications.

The following is an example of how withholding tax data in an invoice is displayed when exported to cXML. A withholding tax rate of -2% was applied to the invoice that was submitted via Ariba Network:

```
<Tax><Money currency="USD">1273.35</Money><Description xml:lang="en-US"></Description>
  <TaxDetail category="withholdingTax" percentageRate="-2">
    <TaxableAmount><Money currency="USD">25466.90</Money></TaxableAmount>
    <TaxAmount><Money currency="USD">-509.34</Money></TaxAmount>
    <Description xml:lang="en-US"></Description>
  </TaxDetail>
  <TaxDetail category="vat" percentageRate="7" taxPointDate="2011-03-11T00:00:00-08:00">
    <TaxableAmount><Money currency="USD">25466.90</Money></TaxableAmount>
    <TaxAmount><Money currency="USD">1782.69</Money></TaxAmount>
    <Description xml:lang="en-US"></Description>
  </TaxDetail>
  <Extrinsic name="withholdingTaxTotal"><Money currency="USD">-509.34</Money></Extrinsic>
  <Extrinsic name="taxTotal"><Money currency="USD">1782.69</Money></Extrinsic>
</Tax>
```

## Fields for Countries with a VAT System

Buying organizations might require invoices to contain data mandated by countries using VAT systems. Suppliers should check their customers' invoice rules on Ariba Network to see whether these fields are mandatory.

### Mandatory Line Item Description

The following example shows a line item description:

```
<InvoiceDetailItemReference lineNumber="1">
  ...
  <Description xml:lang="en">Blue Ballpoint Pens, Retractable</Description>
</InvoiceDetailItemReference>
```

### Mandatory Bill To

The following example shows a Bill To contact:

```
<InvoicePartner>
  <Contact role="billTo" addressID="4319">
    <Name xml:lang="en">Mike Smith</Name>
```

```

    <PostalAddress name="default">
      <DeliverTo>Mike Smith</DeliverTo>
      <Street>15 Rue Des Fleurs</Street>
      <City>Libourne</City>
      <PostalCode>33506</PostalCode>
      <Country isoCountryCode="FR">France</Country>
    </PostalAddress>
    <Email name="default">msmith@buyer.com</Email>
    <Phone name="work">
      <TelephoneNumber>
        <CountryCode isoCountryCode="FR">33</CountryCode>
        <AreaOrCityCode>562</AreaOrCityCode>
        <Number>35820</Number>
      </TelephoneNumber>
    </Phone>
  </Contact>
</InvoicePartner>

```

### Mandatory Remit To

The following example shows a Remit To contact:

```

<InvoicePartner>
  <Contact role="remitTo" addressID="Billing">
    <Name xml:lang="en">Lisa King</Name>
    <PostalAddress name="billing department">
      <DeliverTo>Lisa King</DeliverTo>
      <Street>16 Rue De L'ecole</Street>
      <City>Paris</City>
      <PostalCode>91250</PostalCode>
      <Country isoCountryCode="FR">France</Country>
    </PostalAddress>
    <Email name="default">lking@supplier.com</Email>
    <Phone name="work">
      <TelephoneNumber>
        <CountryCode isoCountryCode="FR">33</CountryCode>
        <AreaOrCityCode>1</AreaOrCityCode>
        <Number>99900</Number>
      </TelephoneNumber>
    </Phone>
  </Contact>
  <IdReference identifier="00000-11111" domain="accountReceivableID">
    <Creator xml:lang="en">Supplier's Back Office System</Creator>
  </IdReference>
  <IdReference identifier="123456789" domain="bankRoutingID">
    <Creator xml:lang="en">First National Bank of Paris</Creator>
  </IdReference>
</InvoicePartner>

```

### Mandatory Ship From and Ship To

The following example shows a Ship To and Ship From contact if the isShippingInLine attribute is not set to yes:

```

<InvoiceDetailShipping>
  <Contact role="shipFrom" addressID="1000487">
    <Name xml:lang="en">Supplier DotCom, Paris</Name>
    <PostalAddress name="default">
      <Street>16 Rue De L'ecole</Street>
      <City>Paris</City>
    </PostalAddress>
  </Contact>

```

```

        <PostalCode>91250</PostalCode>
        <Country isoCountryCode="FR">France</Country>
    </PostalAddress>
    <Email name="default">lking@supplier.com</Email>
    <Phone name="work">
        <TelephoneNumber>
            <CountryCode isoCountryCode="FR">33</CountryCode>
            <AreaOrCityCode>1</AreaOrCityCode>
            <Number>99900</Number>
        </TelephoneNumber>
    </Phone>
</Contact>
<Contact role="shipTo" addressID="1000488">
    <Name xml:lang="en">Libourne Headquarters</Name>
    <PostalAddress name="default">
        <DeliverTo>Mike Smith</DeliverTo>
        <Street>15 Rue Des Fleurs</Street>
        <City>Libourne</City>
        <PostalCode>33506</PostalCode>
        <Country isoCountryCode="FR">France</Country>
    </PostalAddress>
    <Email name="default">msmith@buyer.com</Email>
    <Phone name="work">
        <TelephoneNumber>
            <CountryCode isoCountryCode="FR">33</CountryCode>
            <AreaOrCityCode>562</AreaOrCityCode>
            <Number>35820</Number>
        </TelephoneNumber>
    </Phone>
</Contact>
</InvoiceDetailShipping>

```

### Mandatory Buyer VAT ID and Supplier VAT ID

The supplier's VAT ID is always required. The buyer's VAT ID is required for intra-European Union transactions. Intra-EU is defined as trade within the European Union, where the From and To country codes are in the European Union and are different countries. The following example shows buyer and supplier VAT IDs:

```

<InvoiceDetailRequestHeader>
    ...
    <Extrinsic name="buyerVatID">SE123456789087</Extrinsic>
    <Extrinsic name="supplierVatID">CH987654321</Extrinsic>
</InvoiceDetailRequestHeader>

```

### Mandatory Supplier Information

In some countries, the tax authorities requires suppliers to include their commercial registration ID and legal details in all electronic invoices. The commercial registration ID is not the same as the supplier VAT ID. Legal details are used to capture legal information about the supplier, such as "CEO" or another corporate credential.

```

<InvoiceDetailRequestHeader>
    ...
    <Extrinsic name="supplierCommercialIdentifier">1234567890</Extrinsic>
    <Extrinsic name="supplierCommercialCredentials">CEO</Extrinsic>
</InvoiceDetailRequestHeader>

```

**Note:** Ariba Buyer, Ariba Procure-to-Pay and Ariba Invoice Professional support the following maximum number of characters. Any field values exceeding the limit will be truncated to:

- `supplierCommercialIdentifier`: 50 characters
- `supplierCommercialCredentials`: 100 characters.

### Mandatory Supply Date

VAT entries must have a `TaxDetail` element with a `category="vat"` attribute. The supply date is mandatory if it differs from the invoice date, so VAT entries must have a `taxPointDate` attribute. Buyers can request the supply date using an invoice rule. The following example shows VAT information and a supply date:

```
<Tax>
  <Money currency="EUR">799.60</Money>
  <Description xml:lang="en-GB">Value Added Tax</Description>
  <TaxDetail category="vat"
    percentageRate="8"
    taxPointDate="2005-04-20T23:59:45+01:00">
    <TaxableAmount>
      <Money currency="EUR">9995.00</Money>
    </TaxableAmount>
    <TaxAmount>
      <Money currency="EUR">799.60</Money>
    </TaxAmount>
    <TaxLocation xml:lang="en-GB">Europe</TaxLocation>
  </TaxDetail>
</Tax>
```

### Mandatory VAT in Both Buyer's and Supplier's Local Currency

VAT entries must specify values in both the buyer's and the supplier's local currency. The following example shows how to use the `alternateAmount` and `alternateCurrency` attributes to specify an amount in the buying organization's local currency:

```
<Tax>
  <Money alternateAmount="5.28" alternateCurrency="GBP"
    currency="EUR">10.00</Money>
  <Description xml:lang="en-US"></Description>
  <TaxDetail category="vat" percentageRate="10">
    <TaxableAmount>
      <Money currency="EUR">100.00</Money>
    </TaxableAmount>
    <TaxAmount>
      <Money alternateAmount="5.28" alternateCurrency="GBP"
        currency="EUR">10.00</Money>
      </TaxAmount>
    <Description xml:lang="en-US"></Description>
  </TaxDetail>
</Tax>
```

### Mandatory Explanation of Zero-Value VAT Entries

When the VAT is 0% for certain goods or services in the invoice, suppliers can specify if the VAT is exempt or zero rated. The `exemptDetail` attribute in the `TaxDetail` element is required if a buying organization enables the invoice rule **"Require explanation for zero-rate VAT."**

For example:

```

Exempt Tax Detail
<Tax>
  <Money currency="EUR"0.00</Money>
  <Description xml:lang="en-US"></Description>
  <TaxDetail category="vat" percentageRate="0"exemptDetail="zeroRated">
    <TaxableAmount>
      <Money currency="EUR">100.00</Money>
    </TaxableAmount>
    <TaxAmount>
      <Money currency="EUR">0.00</Money>
    </TaxAmount>
  </Tax>

```

## Attachments

Suppliers sometimes clarify invoices with associated memos, faxes, or drawings. They can use Ariba Network or cXML to attach files to invoices using a MIME envelope as described in the *cXML User's Guide*. Buying organizations can configure Ariba Network to include attachments when forwarding invoices to Ariba Buyer 8.2 or later.

### Attachments on Ariba Network

Ariba Network stores invoice attachments for retrieval by both trading partners. Users can retrieve these files by logging on to their Ariba Network accounts.

Attachments expire 18 months after Ariba Network receives them. Expired attachments are not available online.

### Attachment File Names

Ariba Network supports attachments with file names in all supported character sets, including Chinese, Japanese, and Korean. Ariba Network conforms to Multipurpose Internet Mail Extensions (MIME) standards by encoding non-ASCII filenames according to Internet Engineer Task Force (IETF) Request for Comment (RFC) 2047. Filenames in Asian languages can contain up to 15 characters, and filenames in Latin-1 languages can contain up to 21 eight-bit characters or 70 seven-bit characters.

### AttachmentOnline Extrinsic

If buying organizations configure their Ariba Network accounts to send invoices but not attachments, Ariba Network adds an Extrinsic element named “AttachmentOnline” to the `InvoiceDetailRequestHeader` to indicate the existence of attachments. For example:

```

<Extrinsic name="AttachmentOnline">
  https://service.ariba.com/ad/invoiceDetail?poID=1234&amp;anp=Ariba
</Extrinsic>

```

Buying organizations can use the URL in this Extrinsic to manually log in and view the invoice. They can then click on the listed attachments to view or download them.

Ariba Network adds this Extrinsic only for buying organizations that have selected “Do not download attachments with invoices” in their transaction configuration.



## Maximum Document Size

The maximum size of cXML documents is 4 MB or 3000 line items. You can add attachments up to 10 MB. The 10 MB limit applies to the *total size* of all attachments associated with the document (for example, a purchase order with attachments or an invoice with attachment), which means that the total size for documents with attachments is 14 MB.

If you attempt to add an attachment that exceeds this limit, Ariba Network displays a warning and does not allow you to add the attachment.

Ariba Network can display purchase orders and invoices that contain up to 1,000 line items or are up to 1 MB in size in buying organizations' inboxes and suppliers' outboxes.

## Extrinsic Elements

Suppliers can include Extrinsic elements in invoices.

### Custom Extrinsics

Buying organizations can request that invoices contain custom Extrinsic elements. Suppliers enter values for these Extrinsic elements in the Ariba Network invoice wizard. cXML suppliers that do not use the invoice wizard should contact their customers to see what Extrinsic elements they expect.

### Extrinsics for Clickable Links

Ariba Network can render URLs as clickable links in online purchase orders and invoices. These URLs are produced by Extrinsic elements of the form:

```
<Extrinsic name="anyname">  
  <URL name="click me">http://www.bigcompany.com/info</URL>  
</Extrinsic>
```

The above example produces the following online text:

anyname:[click me](http://www.bigcompany.com/info)

The words “click me” are underlined and are clickable, and the link destination is <http://www.bigcompany.com/info>. If the URL element has no name attribute, Ariba Network displays the URL and makes it clickable.

### Extrinsic for Enabling SSO and ERP

The rule “Allow users to view invoices on Ariba Network from your application” enables users within your organization to access invoices submitted through your ERP when they log in to the Ariba account. Ariba Network can include a URL in the invoice using the following Extrinsic element:

```
<Extrinsic name="AribaNetwork.InvoiceDisplay">  
  <URL name="click me">http://www.bigcompany.com/info</URL>  
</Extrinsic>
```

Contact your Ariba Customer Support representative to assist you with activating this feature.

## Extrinsic for Service Line Items

InvoiceDetailItem can be applied to both goods and service line items in invoices. In the case of service line items, the name attribute can be IsShippingServiceItem or IsSpecialHandlingServiceItem, or ServiceLocation (which must contain a contact element).

## Extrinsic for Polish Document Titles in the Invoice cXML

Buyers can specify the rule, “**Require suppliers to include the legal invoice title for invoices and credit memos in EDI or cXML invoices**” to allow Polish suppliers (supplier country is Poland or VAT ID starts with “PL”) to send invoices and credit memos having titles in the Polish language through cXML, CSV, or EDI.

Suppliers must ensure that they include the following extrinsic while sending invoices and credit memos through cXML to the buyer:

- cXML invoices:

```
<InvoiceDetailRequestHeader>
    <Extrinsic name="InvoiceTitle">Faktura</Extrinsic>
</InvoiceDetailRequestHeader>
```

- cXML credit memos:

```
<InvoiceDetailRequestHeader>
    <Extrinsic name="InvoiceTitle">Faktura Korygująca</Extrinsic>
</InvoiceDetailRequestHeader>
```

**Note:** When the rule is enabled by the buyer, and the supplier sends the invoice or credit memo without the required extrinsic or incorrect value, Ariba Network displays an error.

When Polish suppliers create an invoice or credit memo through the user interface on Ariba Network, Ariba Network automatically includes the Polish titles in the invoices and credit memos.

For more information about Polish document titles, see the *Ariba Network Guide to Invoicing*.

## Field Lengths

The field lengths for invoice elements and attributes are as follows:

Element or Attribute	Field Description
invoiceDate attribute	Fixed datetime format, such as: 2004-01-23T00:55:14-04:00
invoiceID attribute	128 characters
invoiceLineNumber attribute	<2,000,000,000
Name element	256 characters
Street element	256 characters
City element	256 characters
State element	256 characters
PostalCode element	256 characters

Element or Attribute	Field Description
Country element	256 characters
isoCountryCode attribute	2 characters
agreementID attribute	128 characters
orderID attribute	128 characters
orderDate attribute	Fixed datetime format, such as: 2004-01-23T00:55:14-04:00
quantity attribute	precision 30, scale 15 (30, 15)
UnitOfMeasure element	24 characters
currency attribute	3 characters
Money element	(30, 15)
lineNumber attribute	<2,000,000,000
SupplierPartID element	256 characters
Description element	1000 characters
ShortName element	256 characters
percentageRate attribute	(10, 5)

## PDF Invoice Copy Attachments

In many cases, invoice exception handlers or accounts payable personnel require a rendered format of the invoice data representing the supplier view of the invoice in Ariba Network. If your account is enabled for PDF invoice copy attachments, Ariba Network generates a PDF invoice copy for each invoice and adds it to the cXML invoice as a MIME attachment as follows:

```
<InvoiceDetailRequestHeader>
  <Extrinsic name="invoicePDF"><Attachment>
    <URL>cid:18040725.1344960046396@cxml.org</URL></Attachment></Extrinsic>
</InvoiceDetailRequestHeader>
```

PDF invoice copies are generated for the following supplier-submitted invoice types:

- Invoices submitted through EDI or cXML that are not self-signed by the supplier
- Manually created PO-based or non-PO invoices
- Invoices created through CSV import

PDF copies of invoices are not generated for the following invoices:

- Invoice Conversion Services (ICS) invoices
- Supplier self-signed invoices

For these invoices, the provider or supplier is expected to include a PDF invoice copy as an invoice attachment if required.

The file name for the invoice copy PDF file attachment is a randomized file name, for example: kBAIGmcwmN502a7638191182029.pdf.

## Transmission to Buying Organizations

Ariba Buyer Uses the same GetPending transaction to fetch InvoiceDetailRequest documents as it does for other communication with Ariba Network. The polling interval is determined by Ariba Buyer. For a description of this transmission, see “[Transaction Flow](#)” on page 143.

Ariba Network allows invoices to route only for buying organizations that have enabled their Ariba Network accounts for invoicing. For more information, see “[Invoicing Business Rules](#)” on page 159.

## Status and Cancel Invoices

Buying organizations can set invoice status. Suppliers can cancel invoices.

### Invoice Status

Buying organizations send StatusUpdateRequest documents to set invoice status to:

Invoice Status	Description
processing	The invoice was received by the buying organization and is undergoing reconciliation.
canceled	The invoice was received by the buying organization and was canceled.
reconciled	(Also called “approved”) All amounts in the invoice were matched against amounts in a purchase order or a contract.
paying	The invoice is in the payment process or has been partially paid. This status applies only if the buying organization uses invoices to trigger payment.
paid	The invoice was paid. This status applies only if the buying organization uses invoices to trigger payment.
rejected	The invoice was received by the buying organization and was rejected.

cXML-enabled suppliers can receive these documents. Ariba Network posts these documents to the URL that suppliers return in their cXML ProfileResponse. After a buying organization reconciles an invoice, suppliers cannot change or cancel that invoice. Suppliers cannot invoice against canceled purchase orders.

StatusUpdateRequest documents can identify invoices in two possible ways: through either a DocumentReference or an InvoiceIDInfo element. Buying organizations should use a DocumentReference if they know the invoice’s payloadID, and they should use an InvoiceIDInfo element if they know the invoice’s invoiceID and invoiceDate.

### Cancel Invoices

Suppliers can cancel invoices that do not have an invoice status of “reconciled,” “paying,” or “paid.” They cancel invoices by issuing an invoice with InvoiceDetailRequestHeader operation=“delete”. Ariba Network sets the document status to “obsoleted.” Suppliers cannot cancel an invoice that is already canceled.

Ariba Buyer downloads canceled invoices, but it does not distinguish between new and canceled ones. It continues to operate on the original invoice. It sends invoice status updates for “obsoleted” invoices, but not for cancelling invoices (invoices with `InvoiceDetailRequestHeader operation="delete"`). That is, if invoice B cancels invoice A, Ariba Buyer sends invoice status updates for invoice A, not for invoice B. Some customers modify Ariba Buyer to process cancel invoices differently; suppliers should consult with them to see how cancel invoices are handled.

Procure-to-Pay downloads canceled invoices, evaluates them, and sends a `StatusUpdateRequest` document to Ariba Network. While Procure-to-Pay evaluates canceled invoices, Ariba Network sets their document status to “canceling.” If Procure-to-Pay approves them, Ariba Network sets their document status to “canceled.” If Procure-to-Pay rejects them, Ariba Network sets their document status back to their original state.

## Example Invoices

The following examples illustrate common uses for invoices.

- [Summary Invoice](#)
- [Header-Level Invoice](#)
- [Blanket Purchase Order Invoice](#)
- [Complex Scenario](#)
- [Cancel Invoice](#)

## Summary Invoice

The following example shows a summary invoice against two different purchase orders.

It contains `payloadID` attributes at the line item level, which refers to the `payloadID` of the two original `OrderRequest` documents. The `invoiceLineNumber` attribute refers to the line item number in the invoice. The `lineNumber` attribute refers to the line item number in the original `OrderRequest`.

**Note:** With contract labor invoices, Ariba Network displays each line item of the invoice in the summary.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/InvoiceDetail.dtd">
<cXML payloadID="125multi-supplierxyzkjlkwxx-nju"
  timestamp="2001-04-20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN0100001234</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000006789</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="DUNS">
        <Identity>556376197</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>cXML V1.2 application</UserAgent>
    </Sender>
```

```

</Header>
<Request >
  <InvoiceDetailRequest>
    <InvoiceDetailRequestHeader
      invoiceDate="2001-12-07T00:00:00-07:00"
      invoiceID="MULTIINV1" purpose="standard" operation="new">
    <InvoiceDetailHeaderIndicator></InvoiceDetailHeaderIndicator>
    <InvoiceDetailLineIndicator isTaxInLine="yes"
      isShippingInLine="yes" isAccountingInLine="yes">
    </InvoiceDetailLineIndicator>
    <InvoicePartner>
      <Contact role="soldTo" addressID="B2.4.353">
        <Name xml:lang="en">Buyer 1</Name>
        <PostalAddress name="default">
          <DeliverTo>Buyer 1</DeliverTo>
          <Street>15 Camino del Cerro</Street>
          <City>Los Gatos</City>
          <State>CA</State>
          <PostalCode>95032</PostalCode>
          <Country isoCountryCode="US">United States</Country>
        </PostalAddress>
        <Email name="default">test@buyer.com</Email>
        <Phone name="work">
          <TelephoneNumber>
            <CountryCode isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode>408</AreaOrCityCode>
            <Number>3582000</Number>
          </TelephoneNumber>
        </Phone>
        <Fax name="work">
          <TelephoneNumber>
            <CountryCode isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode>408</AreaOrCityCode>
            <Number>3582100</Number>
          </TelephoneNumber>
        </Fax>
      </Contact>
    </InvoicePartner>
    <InvoicePartner>
      <Contact role="remitTo" addressID="Billing">
        <Name xml:lang="en">Joan Bill</Name>
        <PostalAddress name="billing department">
          <DeliverTo>Joan Bill</DeliverTo>
          <Street>16 Castro Street</Street>
          <City>Mountain View</City>
          <State>CA</State>
          <PostalCode>95035</PostalCode>
          <Country isoCountryCode="US">United States</Country>
        </PostalAddress>
        <Email name="default">jbill@supplierbank.com</Email>
        <Phone name="work">
          <TelephoneNumber>
            <CountryCode isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode>650</AreaOrCityCode>
            <Number>9990000</Number>
          </TelephoneNumber>
        </Phone>
      </Contact>
      <IdReference identifier="011" domain="accountReceivableID">
        <Creator xml:lang="en">Supplier ERP</Creator>
      </IdReference>
      <IdReference identifier="123456789" domain="bankRoutingID">
        <Creator xml:lang="en">Supplier Bank</Creator>
      </IdReference>
    </InvoicePartner>
  </InvoiceDetailRequest>
</Request>

```

```

</InvoicePartner>
<PaymentTerm payInNumberOfDays="20">
  <Discount>5</Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays="30">
  <Discount>2.5</Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays="40">
  <Discount>-2</Discount>
</PaymentTerm>
</InvoiceDetailRequestHeader>
<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <OrderReference orderID="P0123">
      <DocumentReference
        payloadID="277403463.70.7733@acme.com">
      </DocumentReference>
    </OrderReference>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem invoiceLineNumber="1" quantity="5">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice><Money currency="USD">10.00</Money></UnitPrice>
    <InvoiceDetailItemReference lineNumber="1">
      <ItemID><SupplierPartID>A2</SupplierPartID></ItemID>
      <Description xml:lang="en">Cabinet Locks with 2 Keys
      </Description>
      <SerialNumber>45993823469876</SerialNumber>
      <SerialNumber>45993823469877</SerialNumber>
      <SerialNumber>45993823469878</SerialNumber>
      <SerialNumber>45993823469879</SerialNumber>
      <SerialNumber>45993823469880</SerialNumber>
    </InvoiceDetailItemReference>
    <SubtotalAmount>
      <Money currency="USD">50.00</Money>
    </SubtotalAmount>
    <GrossAmount>
      <Money currency="USD">50.00</Money>
    </GrossAmount>
    <NetAmount>
      <Money currency="USD">50.00</Money>
    </NetAmount>
    <Distribution>
      <Accounting name="DistributionCharge">
        <AccountingSegment id="100">
          <Name xml:lang="en-US">Split Percentage</Name>
          <Description xml:lang="en-US">Percentage
          </Description>
        </AccountingSegment>
        <AccountingSegment id="Machine Resources">
          <Name xml:lang="en-US">Cost Center</Name>
          <Description xml:lang="en-US">Department Name
          </Description>
        </AccountingSegment>
        <AccountingSegment id="Office Supplies">
          <Name xml:lang="en-US">Account</Name>
          <Description xml:lang="en-US">Account Name
          </Description>
        </AccountingSegment>
      </Accounting>
      <Charge>
        <Money currency="USD">50.00</Money>
      </Charge>
    </Distribution>
  </InvoiceDetailItem>

```

```

</InvoiceDetailOrder>
<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <OrderReference orderID="P0123">
      <DocumentReference
        payloadID="1104750653.65.7733@acme.com">
      </DocumentReference>
    </OrderReference>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem invoiceLineNumber="2" quantity="10">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice><Money currency="USD">1.00</Money></UnitPrice>
    <InvoiceDetailItemReference lineNumber="3">
      <ItemID><SupplierPartID>A4</SupplierPartID></ItemID>
      <Description xml:lang="en">Cam Locks without Keys
      </Description>
      <ManufacturerPartID>815-12</ManufacturerPartID>
      <ManufacturerName xml:lang="en-US"></ManufacturerName>
    </InvoiceDetailItemReference>
    <SubtotalAmount>
      <Money currency="USD">10.00</Money>
    </SubtotalAmount>
    <GrossAmount>
      <Money currency="USD">10.00</Money>
    </GrossAmount>
    <NetAmount>
      <Money currency="USD">10.00</Money>
    </NetAmount>
    <Distribution>
      <Accounting name="DistributionCharge">
        <AccountingSegment id="100">
          <Name xml:lang="en-US">Split Percentage</Name>
          <Description xml:lang="en-US">Percentage
          </Description>
        </AccountingSegment>
        <AccountingSegment id="Machine Resources">
          <Name xml:lang="en-US">Cost Center</Name>
          <Description xml:lang="en-US">Department Name
          </Description>
        </AccountingSegment>
      </Accounting>
      <Charge><Money currency="USD">10.00</Money></Charge>
    </Distribution>
  </InvoiceDetailItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>
  <SubtotalAmount>
    <Money currency="USD">60.00</Money>
  </SubtotalAmount>
  <Tax><Money currency="USD">60.00</Money>
    <Description xml:lang="en-US">Sales Tax</Description>
  </Tax>
  <GrossAmount><Money currency="USD">60.00</Money></GrossAmount>
  <NetAmount><Money currency="USD">60.00</Money></NetAmount>
  <DueAmount><Money currency="USD">60.00</Money></DueAmount>
</InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>

```



## Header-Level Invoice

If suppliers invoice purchase orders in full, it is not necessary to provide line item detail information; a header level invoice is sufficient.

The isHeaderInvoice attribute is set to “yes” to indicate the invoice is of type Header.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/InvoiceDetail.dtd">
<cXML timestamp="2001-12-07T16:23:01-07:00"
  payloadID="Oct102001_0447pm98788688">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN0100001234</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000004321</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="DUNS">
        <Identity>556376197</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>cXML V1.2 application</UserAgent>
    </Sender>
  </Header>
  <Request>
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader invoiceDate="2001-12-07T00:00:00-07:00"
        invoiceID="HEADERINV222" purpose="standard" operation="new">
        <InvoiceDetailHeaderIndicator isHeaderInvoice="yes">
        </InvoiceDetailHeaderIndicator>
        <InvoiceDetailLineIndicator isTaxInLine="yes"
          isShippingInLine="yes"/>
        <InvoicePartner>
          <Contact role="billTo">
            <Name xml:lang="en-US">Buyer Headquarters</Name>
            <PostalAddress>
              <Street>123 Main Street</Street>
              <City>Anytown</City>
              <State>TX</State>
              <PostalCode>99999</PostalCode>
              <Country isoCountryCode="US">United States</Country>
            </PostalAddress>
          </Contact>
        </InvoicePartner>
        <InvoicePartner>
          <Contact role="remitTo">
            <Name xml:lang="en-US">Joan Bill</Name>
            <PostalAddress>
              <Street>One Test Avenue</Street>
              <City>Sunnyvale</City>
              <State>CA</State>
              <PostalCode>94087</PostalCode>
              <Country isoCountryCode="US">United States</Country>
            </PostalAddress>
          </Contact>
          <IdReference identifier="StandardHeaderLevelIndividual"
            domain="accountID"></IdReference>
        </InvoicePartner>
      </InvoiceDetailRequestHeader>
    </InvoiceDetailRequest>
  </Request>
</cXML>
```

```

</InvoicePartner>
  <Comments xml:lang="en-US">Sample Header Level Individual
    Invoice</Comments>
</InvoiceDetailRequestHeader>
<InvoiceDetailHeaderOrder>
  <InvoiceDetailOrderInfo>
    <OrderIDInfo orderID="D021756" orderDate="2007-07-01T09:42:30-05:00">
      </OrderIDInfo>
    </InvoiceDetailOrderInfo>
  <InvoiceDetailOrderSummary invoiceLineNumber="1">
    <SubtotalAmount>
      <Money currency="USD">20.00</Money>
    </SubtotalAmount>
    <Tax>
      <Money currency="USD">5.00</Money>
      <Description xml:lang="en-US"></Description>
    </Tax>
    <InvoiceDetailLineSpecialHandling>
      <Money currency="USD">10.00</Money>
    </InvoiceDetailLineSpecialHandling>
    <InvoiceDetailLineShipping>
      <InvoiceDetailShipping>
        <Contact role="shipFrom" addressID="1000487">
          <Name xml:lang="en">Main Shipping</Name>
          <PostalAddress name="default">
            <Street>15 Camino del Cerro</Street>
            <City>Los Gatos</City>
            <State>CA</State>
            <PostalCode>95032</PostalCode>
            <Country isoCountryCode="US">United States
          </Country>
          </PostalAddress>
          <Email name="default">shipping@shipfrm.com</Email>
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1
            </CountryCode>
            <AreaOrCityCode>408</AreaOrCityCode>
            <Number>3582000</Number>
          </TelephoneNumber>
          </Phone>
          <Fax name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1
            </CountryCode>
            <AreaOrCityCode>408</AreaOrCityCode>
            <Number>3582100</Number>
          </TelephoneNumber>
          </Fax>
        </Contact>
        <Contact role="shipTo" addressID="1000487">
          <Name xml:lang="en">Buyer Headquarters</Name>
          <PostalAddress name="default">
            <DeliverTo>Jason Lynch</DeliverTo>
            <Street>34 Castro Street</Street>
            <City>Mountain View</City>
            <State>CA</State>
            <PostalCode>95035</PostalCode>
            <Country isoCountryCode="US">United States
          </Country>
          </PostalAddress>
          <Email name="default">JasonL@shipto.com</Email>
          <Phone name="work">
            <TelephoneNumber>

```

```

        <CountryCode isoCountryCode="US">1
        </CountryCode>
        <AreaOrCityCode>408</AreaOrCityCode>
        <Number>3582000</Number>
    </TelephoneNumber>
</Phone>
<Fax name="work">
    <TelephoneNumber>
        <CountryCode isoCountryCode="US">1
        </CountryCode>
        <AreaOrCityCode>408</AreaOrCityCode>
        <Number>3582100</Number>
    </TelephoneNumber>
</Fax>
</Contact>
</InvoiceDetailShipping>
    <Money currency="USD">20</Money>
</InvoiceDetailLineShipping>
<GrossAmount>
    <Money currency="USD">35.00</Money>
</GrossAmount>
<InvoiceDetailDiscount percentageRate="10%">
    <Money currency="USD">2.00</Money>
</InvoiceDetailDiscount>
<NetAmount><Money currency="USD">33.00</Money></NetAmount>
<Comments>This a Standard Header Level Invoice</Comments>
</InvoiceDetailOrderSummary>
</InvoiceDetailHeaderOrder>
<InvoiceDetailSummary>
    <SubtotalAmount>
        <Money currency="USD">20.00</Money>
    </SubtotalAmount>
    <Tax>
        <Money currency="USD">5.00</Money>
        <Description xml:lang="en-US"></Description>
    </Tax>
    <SpecialHandlingAmount>
        <Money currency="USD">10.00</Money>
    </SpecialHandlingAmount>
    <ShippingAmount>
        <Money currency="USD">5.00</Money>
    </ShippingAmount>
    <GrossAmount><Money currency="USD">40.00</Money></GrossAmount>
    <InvoiceDetailDiscount percentageRate="10%">
        <Money currency="USD">18.00</Money>
    </InvoiceDetailDiscount>
    <NetAmount><Money currency="USD">18.00</Money></NetAmount>
    <DepositAmount>
        <Money currency="USD">33.00</Money>
    </DepositAmount>
    <DueAmount><Money currency="USD">33.00</Money></DueAmount>
</InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>

```

## Blanket Purchase Order Invoice

The following example shows an invoice for a no-release blanket purchase order (a purchase order is not required prior to invoicing):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://svcdev.ariba.com/schemas/cXML/1.2.018/InvoiceDetail.dtd">
<cXML payloadID="INV011-Against-BPO55@payload" timestamp="2007-08-20T08:49:36-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>AN0100001234</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>AN01000006789</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="DUNS">
        <Identity>556376197</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ariba Network V1.1</UserAgent>
    </Sender>
  </Header>
  <Request deploymentMode="production">
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader invoiceDate="2007-08-20T08:49:36-07:00"
        invoiceID="INV011-Against-BPO55" operation="new" purpose="standard">
        <InvoiceDetailHeaderIndicator></InvoiceDetailHeaderIndicator>
        <InvoiceDetailLineIndicator></InvoiceDetailLineIndicator>
        <InvoicePartner>
          <Contact role="remitTo">
            <Name xml:lang="en-US">SUPPLIER</Name>
            <PostalAddress>
              <Street>jUnitDummy</Street>
              <City>Sunnyvale</City>
              <State>CA</State>
              <PostalCode>94089</PostalCode>
              <Country isoCountryCode="US">United States</Country>
            </PostalAddress>
          </Contact>
        </InvoicePartner>
        <InvoicePartner>
          <Contact role="soldTo">
            <Name xml:lang="en-US">BUYER</Name>
            <PostalAddress>
              <Street>jUnitDummy</Street>
              <City>Sunnyvale</City>
              <State>CA</State>
              <PostalCode>94089</PostalCode>
              <Country isoCountryCode="US">United States</Country>
            </PostalAddress>
          </Contact>
        </InvoicePartner>
        <InvoicePartner>
          <Contact role="billTo">
            <Name xml:lang="en-US"></Name>
          </Contact>
        </InvoicePartner>
      </InvoicePartner>
    </InvoiceDetailRequest>
  </Request>
</cXML>
```

```

    <Contact role="from">
      <Name xml:lang="en-US">SUPPLIER</Name>
      <PostalAddress>
        <Street>jUnitDummy</Street>
        <City></City>
        <State>CA</State>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Contact>
  </InvoicePartner>
  <InvoiceDetailShipping>
    <Contact role="shipTo">
      <Name xml:lang="en-US">BUYER</Name>
      <PostalAddress>
        <Street>jUnitDummy</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>94089</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Contact>
    <Contact role="shipFrom">
      <Name xml:lang="en-US">SUPPLIER</Name>
      <PostalAddress>
        <Street>jUnitDummy</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>94089</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
    </Contact>
  </InvoiceDetailShipping>
</InvoiceDetailRequestHeader>
<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <OrderReference orderID="BP055">
      <DocumentReference payloadID="BP055@payloadID"></DocumentReference>
    </OrderReference>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem invoiceLineNumber="1" quantity="1">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice><Money currency="USD">44.00</Money></UnitPrice>
    <InvoiceDetailItemReference lineNumber="1">
      <ItemID><SupplierPartID>AD4BNC13</SupplierPartID></ItemID>
      <Description xml:lang="en">Adapter SUN Monitor 4-BNCF/13W3M</Description>
    </InvoiceDetailItemReference>
    <SubtotalAmount>
      <Money currency="USD">44.00</Money>
    </SubtotalAmount>
    <GrossAmount>
      <Money currency="USD">44.00</Money>
    </GrossAmount>
    <NetAmount>
      <Money currency="USD">44.00</Money>
    </NetAmount>
  </InvoiceDetailItem>
  <InvoiceDetailItem invoiceLineNumber="2" quantity="1">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice><Money currency="USD">34.00</Money></UnitPrice>
    <InvoiceDetailItemReference lineNumber="2">
      <ItemID><SupplierPartID>AD1513</SupplierPartID></ItemID>
      <Description xml:lang="en">Adapter SUN Monitor HD15F/13W3M</Description>
    </InvoiceDetailItemReference>
  </InvoiceDetailItem>
</InvoiceDetailOrder>

```

```

    </InvoiceDetailItemReference>
    <SubtotalAmount>
      <Money currency="USD">34.00</Money>
    </SubtotalAmount>
    <GrossAmount>
      <Money currency="USD">34.00</Money>
    </GrossAmount>
    <NetAmount>
      <Money currency="USD">34.00</Money>
    </NetAmount>
  </InvoiceDetailItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>
  <SubtotalAmount>
    <Money currency="USD">78.00</Money>
  </SubtotalAmount>
  <Tax><Money currency="USD">10.00</Money>
    <Description xml:lang="en-US"></Description>
  </Tax>
  <GrossAmount><Money currency="USD">78.00</Money></GrossAmount>
  <NetAmount><Money currency="USD">78.00</Money></NetAmount>
  <DueAmount><Money currency="USD">78.00</Money></DueAmount>
</InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>

```

For a release BPO (requires purchase orders prior to invoicing), the `OrderRequestHeader` element is present, and the `orderType` is `blanket`, as shown in the following example:

```

<OrderRequest>
  <OrderRequestHeader orderID="D012042"
    orderDate="2008-12-04T15:26:00-07:00"
    type="new" orderType="blanket">
  </OrderRequestHeader>
</OrderRequest>

```

## Complex Scenario

The following cXML documents describe an example complex business scenario:

- 1 Ariba Buyer generates a purchase order containing two line items:
  - Line 1:** Ten items at \$20 = \$200
  - Line 2:** Five items at \$20 = \$100
- 2 The supplier sends a **standard invoice** that includes shipping charges, but mistakenly charges the buying organization for eight items on line 2 instead of five.
- 3 The supplier sends a **credit memo** for \$110, which includes tax charges.
- 4 The supplier realizes that it used incorrect math in the credit memo; it should have credited the buying organization \$66. The supplier then sends a **debit memo** for \$44 (\$110 - \$66).

## Purchase Order

The following purchase order contains two line items:

**Line 1:** Ten items at \$20 = \$200

**Line 2:** Five items at \$20 = \$100

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="1002700953000.152865612.2314.120401002@bigcompany.com"
  timestamp="2001-12-04T15:26:00-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>bettybuyer@bigcompany.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>samsupplier@acme.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>bettybuyer@bigcompany.com</Identity>
        <SharedSecret>OurPassword</SharedSecret>
      </Credential>
      <UserAgent>Ariba Buyer 7.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <OrderRequest>
      <OrderRequestHeader orderID="D012042"
        orderDate="2001-12-04T15:26:00-07:00"
        type="new" orderType="regular">
      </OrderRequestHeader>
      <Total>
        <Money currency="USD">300.00</Money>
      </Total>
      <ShipTo>
        <Address isoCountryCode="US" addressID="1000467">
          <Name xml:lang="en">Bigcompany Headquarters</Name>
          <PostalAddress name="default">
            <DeliverTo>Betty Buyer</DeliverTo>
            <DeliverTo>Bigcompany Headquarters</DeliverTo>
            <Street>1314 Chesapeake Terrace</Street>
            <City>Sunnyvale</City>
            <State>CA</State>
            <PostalCode>94089</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="default">username@ariba.com</Email>
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>800</AreaOrCityCode>
              <Number>5555555</Number>
            </TelephoneNumber>
          </Phone>
          <Fax name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode></AreaOrCityCode>
              <Number></Number>
            </TelephoneNumber>
          </Fax>
        </Address>
      </ShipTo>
    </OrderRequest>
  </Request>
</cXML>
```

```

        </Fax>
      </Address>
    </ShipTo>
    <BillTo>
      <Address isoCountryCode="US" addressID="15">
        <Name xml:lang="en">Bigcompany Headquarters</Name>
        <PostalAddress name="Accounts Payable">
          <Street>1314 Chesapeake Terrace</Street>
          <City>Sunnyvale</City>
          <State>CA</State>
          <PostalCode>94089</PostalCode>
          <Country isoCountryCode="US">United States</Country>
        </PostalAddress>
        <Phone name="work">
          <TelephoneNumber>
            <CountryCode isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode></AreaOrCityCode>
            <Number></Number>
          </TelephoneNumber>
        </Phone>
        <Fax name="work">
          <TelephoneNumber>
            <CountryCode isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode></AreaOrCityCode>
            <Number></Number>
          </TelephoneNumber>
        </Fax>
      </Address>
    </BillTo>
  </OrderRequestHeader>
  <ItemOut quantity="10" lineNumber="1">
    <ItemID>
      <SupplierPartID>BTM00107</SupplierPartID>
    </ItemID>
    <ItemDetail>
      <UnitPrice>
        <Money currency="USD">20.00</Money>
      </UnitPrice>
      <Description xml:lang="en">Computer Audio Cables</Description>
      <UnitOfMeasure>EA</UnitOfMeasure>
      <Classification domain="UNSPSC">43173609</Classification>
      <ManufacturerPartID>JJ11P28</ManufacturerPartID>
      <Extrinsic name="PR No.">PR1026</Extrinsic>
    </ItemDetail>
    <Distribution>
      <Accounting name="DistributionCharge">
        <AccountingSegment id="Production Control">
          <Name xml:lang="en-US">Cost Center</Name>
          <Description xml:lang="en-US">
            Department Name
          </Description>
        </AccountingSegment>
        <AccountingSegment id="Computer Accessories">
          <Name xml:lang="en-US">Account</Name>
          <Description xml:lang="en-US">
            Account Name
          </Description>
        </AccountingSegment>
      </Accounting>
      <Charge>
        <Money currency="USD">200.00</Money>
      </Charge>
    </Distribution>
  </ItemOut>

```



```

<ItemOut quantity="5" lineNumber="2">
  <ItemID>
    <SupplierPartID>BTM00108</SupplierPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">20.00</Money>
    </UnitPrice>
    <Description xml:lang="en">Computer Video Cables</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="UNSPSC">43173610</Classification>
    <ManufacturerPartID>JJ11P29</ManufacturerPartID>
    <Extrinsic name="PR No.">PR1026</Extrinsic>
  </ItemDetail>
  <Distribution>
    <Accounting name="DistributionCharge">
      <AccountingSegment id="Production Control">
        <Name xml:lang="en-US">Cost Center</Name>
        <Description xml:lang="en-US">
          Department Name
        </Description>
      </AccountingSegment>
      <AccountingSegment id="Computer Accessories">
        <Name xml:lang="en-US">Account</Name>
        <Description xml:lang="en-US">
          Account Name
        </Description>
      </AccountingSegment>
    </Accounting>
    <Charge>
      <Money currency="USD">100.00</Money>
    </Charge>
  </Distribution>
</ItemOut>
</OrderRequest>
</Request>
</cXML>

```

## Standard Invoice

The following invoice includes shipping charges, but mistakenly charges the buying organization for eight items instead of five items on line 2.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/InvoiceDetail.dtd">
<cXML payloadID="INVD012042.120402" timestamp="2001-12-04T17:30:00-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>samsupplier@acme.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>bettybuyer@bigcompany.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>samsupplier@acme.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
    </Sender>
  </Header>
  <LineItems>
    <LineItem>
      <ItemOut>
        <ItemID>
          <SupplierPartID>BTM00108</SupplierPartID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency="USD">20.00</Money>
          </UnitPrice>
          <Description xml:lang="en">Computer Video Cables</Description>
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Classification domain="UNSPSC">43173610</Classification>
          <ManufacturerPartID>JJ11P29</ManufacturerPartID>
          <Extrinsic name="PR No.">PR1026</Extrinsic>
        </ItemDetail>
        <Distribution>
          <Accounting name="DistributionCharge">
            <AccountingSegment id="Production Control">
              <Name xml:lang="en-US">Cost Center</Name>
              <Description xml:lang="en-US">
                Department Name
              </Description>
            </AccountingSegment>
            <AccountingSegment id="Computer Accessories">
              <Name xml:lang="en-US">Account</Name>
              <Description xml:lang="en-US">
                Account Name
              </Description>
            </AccountingSegment>
          </Accounting>
          <Charge>
            <Money currency="USD">100.00</Money>
          </Charge>
        </Distribution>
      </ItemOut>
    </LineItem>
    <LineItem>
      <ItemOut>
        <ItemID>
          <SupplierPartID>BTM00108</SupplierPartID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency="USD">20.00</Money>
          </UnitPrice>
          <Description xml:lang="en">Computer Video Cables</Description>
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Classification domain="UNSPSC">43173610</Classification>
          <ManufacturerPartID>JJ11P29</ManufacturerPartID>
          <Extrinsic name="PR No.">PR1026</Extrinsic>
        </ItemDetail>
        <Distribution>
          <Accounting name="DistributionCharge">
            <AccountingSegment id="Production Control">
              <Name xml:lang="en-US">Cost Center</Name>
              <Description xml:lang="en-US">
                Department Name
              </Description>
            </AccountingSegment>
            <AccountingSegment id="Computer Accessories">
              <Name xml:lang="en-US">Account</Name>
              <Description xml:lang="en-US">
                Account Name
              </Description>
            </AccountingSegment>
          </Accounting>
          <Charge>
            <Money currency="USD">100.00</Money>
          </Charge>
        </Distribution>
      </ItemOut>
    </LineItem>
    <LineItem>
      <ItemOut>
        <ItemID>
          <SupplierPartID>BTM00108</SupplierPartID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency="USD">20.00</Money>
          </UnitPrice>
          <Description xml:lang="en">Computer Video Cables</Description>
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Classification domain="UNSPSC">43173610</Classification>
          <ManufacturerPartID>JJ11P29</ManufacturerPartID>
          <Extrinsic name="PR No.">PR1026</Extrinsic>
        </ItemDetail>
        <Distribution>
          <Accounting name="DistributionCharge">
            <AccountingSegment id="Production Control">
              <Name xml:lang="en-US">Cost Center</Name>
              <Description xml:lang="en-US">
                Department Name
              </Description>
            </AccountingSegment>
            <AccountingSegment id="Computer Accessories">
              <Name xml:lang="en-US">Account</Name>
              <Description xml:lang="en-US">
                Account Name
              </Description>
            </AccountingSegment>
          </Accounting>
          <Charge>
            <Money currency="USD">100.00</Money>
          </Charge>
        </Distribution>
      </ItemOut>
    </LineItem>
    <LineItem>
      <ItemOut>
        <ItemID>
          <SupplierPartID>BTM00108</SupplierPartID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency="USD">20.00</Money>
          </UnitPrice>
          <Description xml:lang="en">Computer Video Cables</Description>
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Classification domain="UNSPSC">43173610</Classification>
          <ManufacturerPartID>JJ11P29</ManufacturerPartID>
          <Extrinsic name="PR No.">PR1026</Extrinsic>
        </ItemDetail>
        <Distribution>
          <Accounting name="DistributionCharge">
            <AccountingSegment id="Production Control">
              <Name xml:lang="en-US">Cost Center</Name>
              <Description xml:lang="en-US">
                Department Name
              </Description>
            </AccountingSegment>
            <AccountingSegment id="Computer Accessories">
              <Name xml:lang="en-US">Account</Name>
              <Description xml:lang="en-US">
                Account Name
              </Description>
            </AccountingSegment>
          </Accounting>
          <Charge>
            <Money currency="USD">100.00</Money>
          </Charge>
        </Distribution>
      </ItemOut>
    </LineItem>
    <LineItem>
      <ItemOut>
        <ItemID>
          <SupplierPartID>BTM00108</SupplierPartID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency="USD">20.00</Money>
          </UnitPrice>
          <Description xml:lang="en">Computer Video Cables</Description>
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Classification domain="UNSPSC">43173610</Classification>
          <ManufacturerPartID>JJ11P29</ManufacturerPartID>
          <Extrinsic name="PR No.">PR1026</Extrinsic>
        </ItemDetail>
        <Distribution>
          <Accounting name="DistributionCharge">
            <AccountingSegment id="Production Control">
              <Name xml:lang="en-US">Cost Center</Name>
              <Description xml:lang="en-US">
                Department Name
              </Description>
            </AccountingSegment>
            <AccountingSegment id="Computer Accessories">
              <Name xml:lang="en-US">Account</Name>
              <Description xml:lang="en-US">
                Account Name
              </Description>
            </AccountingSegment>
          </Accounting>
          <Charge>
            <Money currency="USD">100.00</Money>
          </Charge>
        </Distribution>
      </ItemOut>
    </LineItem>
  </LineItems>
</cXML>

```

```

    <UserAgent>Our Invoicing App 2.0</UserAgent>
  </Sender>
</Header>
<Request>
  <InvoiceDetailRequest>
    <InvoiceDetailRequestHeader invoiceID="InvD012042"
      purpose="standard" operation="new"
      invoiceDate="2001-12-04T17:30:00-07:00">
      <InvoiceDetailHeaderIndicator></InvoiceDetailHeaderIndicator>
      <InvoiceDetailLineIndicator isTaxInLine="yes"
        isShippingInLine="yes" isAccountingInLine="yes">
      </InvoiceDetailLineIndicator>
      <InvoicePartner>
        <Contact role="billTo" addressID="Billing">
          <Name xml:lang="en">Bigcompany Headquarters</Name>
          <PostalAddress name="Accounts Payable">
            <DeliverTo>Joe Accountant</DeliverTo>
            <Street>1314 Chesapeake Terrace</Street>
            <City>Sunnayvale</City>
            <State>CA</State>
            <PostalCode>94089</PostalCode>
            <Country isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="default">JoeTheAccountant@bigcompany.com
          </Email>
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>215</AreaOrCityCode>
              <Number>9990000</Number>
            </TelephoneNumber>
          </Phone>
        </Contact>
        <IdReference identifier="11280630"
          domain="accountReceivableID">
          <Creator xml:lang="en">Acme</Creator>
        </IdReference>
      </InvoicePartner>
    </InvoiceDetailRequestHeader>
    <InvoiceDetailOrder>
      <InvoiceDetailOrderInfo>
        <OrderReference orderID="P0123">
          <DocumentReference
            payloadID="1002700953000.152865612.2314.120401002@bigcompany.com">
          </DocumentReference>
        </OrderReference>
      </InvoiceDetailOrderInfo>
      <InvoiceDetailItem invoiceLineNumber="1" quantity="4">
        <UnitOfMeasure>EA</UnitOfMeasure>
        <UnitPrice>
          <Money currency="USD">50.00</Money>
        </UnitPrice>
        <InvoiceDetailItemReference lineNumber="1">
          <ItemID>
            <SupplierPartID>BTM00107</SupplierPartID>
          </ItemID>
          <Description xml:lang="en">Computer Audio Cables</Description>
          <SerialNumber>69876</SerialNumber>
          <SerialNumber>69877</SerialNumber>
          <SerialNumber>69878</SerialNumber>
          <SerialNumber>69879</SerialNumber>
        </InvoiceDetailItemReference>
        <SubtotalAmount>
          <Money currency="USD">200.00</Money>
        </SubtotalAmount>
      </InvoiceDetailItem>
    </InvoiceDetailOrder>
  </InvoiceDetailRequest>
</Request>

```

```

</SubtotalAmount>
<InvoiceDetailLineShipping>
  <InvoiceDetailShipping>
    <Contact role="shipFrom" addressID="1000467">
      <Name xml:lang="en">Acme</Name>
      <PostalAddress name="Acme">
        <Street>123 Main Street</Street>
        <City>Anytown</City>
        <State>PA</State>
        <PostalCode>99999</PostalCode>
        <Country isoCountryCode="US">
          United States</Country>
        </PostalAddress>
        <Email name="default">shipping@acme.com</Email>
        <Phone name="work">
          <TelephoneNumber>
            <CountryCode
              isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode>800</AreaOrCityCode>
            <Number>5555555</Number>
          </TelephoneNumber>
        </Phone>
        <Fax name="work">
          <TelephoneNumber>
            <CountryCode
              isoCountryCode="US">1</CountryCode>
            <AreaOrCityCode></AreaOrCityCode>
            <Number></Number>
          </TelephoneNumber>
        </Fax>
      </Contact>
      <Contact role="shipTo" addressID="1000487">
        <Name xml:lang="en">Betty Buyer</Name>
        <PostalAddress name="default">
          <DeliverTo>Betty Buyer</DeliverTo>
          <Street>1314 Chesapeake Terrace</Street>
          <City>Sunnyvale</City>
          <State>CA</State>
          <PostalCode>94089</PostalCode>
          <Country isoCountryCode="US">
            United States</Country>
          </PostalAddress>
          <Email
            name="default">bettybuyer@bigcompany.com</Email>
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode
                isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>800</AreaOrCityCode>
              <Number>6666666</Number>
            </TelephoneNumber>
          </Phone>
          <Fax name="work">
            <TelephoneNumber>
              <CountryCode
                isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>215</AreaOrCityCode>
              <Number>5555555</Number>
            </TelephoneNumber>
          </Fax>
        </Contact>
      </InvoiceDetailShipping>
      <Money currency="USD">5.00</Money>
    </InvoiceDetailLineShipping>
  
```

```

<GrossAmount>
  <Money currency="USD">205.00</Money>
</GrossAmount>
<NetAmount>
  <Money currency="USD">205.00</Money>
</NetAmount>
<Distribution>
  <Accounting name="Buyer assigned accounting code 1">
    <AccountingSegment id="ABC123456789">
      <Name xml:lang="en">Purchase</Name>
      <Description xml:lang="en">
        Production Control</Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">205.00</Money>
  </Charge>
</Distribution>
</InvoiceDetailItem>
<InvoiceDetailItem invoiceLineNumber="2" quantity="8">
  <UnitOfMeasure>EA</UnitOfMeasure>
  <UnitPrice>
    <Money currency="USD">20.00</Money>
  </UnitPrice>
  <InvoiceDetailItemReference lineNumber="2">
    <ItemID>
      <SupplierPartID>BTM00108</SupplierPartID>
    </ItemID>
    <Description xml:lang="en">
      Computer Video Cables</Description>
  </InvoiceDetailItemReference>
  <SubtotalAmount>
    <Money currency="USD">160.00</Money>
  </SubtotalAmount>
  <InvoiceDetailLineShipping>
    <InvoiceDetailShipping>
      <Contact role="shipFrom" addressID="1000467">
        <Name xml:lang="en">Acme </Name>
        <PostalAddress name="Acme">
          <Street>123 Main Street</Street>
          <City>Anytown</City>
          <State>PA</State>
          <PostalCode>99999</PostalCode>
          <Country>
            isoCountryCode="US">United States</Country>
          </PostalAddress>
          <Email name="default">shipping@acme.com</Email>
          <Phone name="work">
            <TelephoneNumber>
              <CountryCode>
                isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode>800</AreaOrCityCode>
              <Number>5555555</Number>
            </TelephoneNumber>
          </Phone>
          <Fax name="work">
            <TelephoneNumber>
              <CountryCode>
                isoCountryCode="US">1</CountryCode>
              <AreaOrCityCode></AreaOrCityCode>
              <Number></Number>
            </TelephoneNumber>
          </Fax>
        </Contact>
      </InvoiceDetailShipping>
    </InvoiceDetailLineShipping>
  </InvoiceDetailItem>
</Invoice>

```

```

    <Contact role="shipTo" addressID="1000487">
      <Name xml:lang="en">Betty Buyer</Name>
      <PostalAddress name="default">
        <DeliverTo>Betty Buyer</DeliverTo>
        <Street>1314 Chesapeake Terrace</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>94089</PostalCode>
        <Country
          isoCountryCode="US">United States</Country>
        </PostalAddress>
      <Email
        name="default">bettybuyer@bigcompany.com</Email>
      <Phone name="work">
        <TelephoneNumber>
          <CountryCode
            isoCountryCode="US">1</CountryCode>
          <AreaOrCityCode>800</AreaOrCityCode>
          <Number>66666666</Number>
        </TelephoneNumber>
      </Phone>
      <Fax name="work">
        <TelephoneNumber>
          <CountryCode
            isoCountryCode="US">1</CountryCode>
          <AreaOrCityCode>215</AreaOrCityCode>
          <Number>5555555</Number>
        </TelephoneNumber>
      </Fax>
    </Contact>
  </InvoiceDetailShipping>
  <Money currency="USD">16.00</Money>
</InvoiceDetailLineShipping>
<GrossAmount>
  <Money currency="USD">176.00</Money>
</GrossAmount>
<NetAmount>
  <Money currency="USD">176.00</Money>
</NetAmount>
<Distribution>
  <Accounting name="Buyer assigned accounting code 1">
    <AccountingSegment id="ABC123456789">
      <Name xml:lang="en">Purchase</Name>
      <Description xml:lang="en">
        Production Control</Description>
    </AccountingSegment>
  </Accounting>
  <Charge>
    <Money currency="USD">176.00</Money>
  </Charge>
</Distribution>
</InvoiceDetailItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>
  <SubtotalAmount>
    <Money currency="USD">360.00</Money>
  </SubtotalAmount>
  <Tax>
    <Money currency="USD">36.00</Money>
    <Description xml:lang="en">total tax</Description>
    <TaxDetail purpose="tax" category="State sales tax"
      percentageRate="10">
    <TaxableAmount>
      <Money currency="USD">360.00</Money>
    </TaxableAmount>
  </Tax>
</InvoiceDetailSummary>

```

```
        </TaxableAmount>
        <TaxAmount>
          <Money currency="USD">36.00</Money>
        </TaxAmount>
        <TaxLocation xml:lang="en">PA</TaxLocation>
      </TaxDetail>
    </Tax>
    <ShippingAmount>
      <Money currency="USD">21.00</Money>
    </ShippingAmount>
    <GrossAmount>
      <Money currency="USD">417.00</Money>
    </GrossAmount>
    <NetAmount>
      <Money currency="USD">417.00</Money>
    </NetAmount>
    <DueAmount>
      <Money currency="USD">417.00</Money>
    </DueAmount>
  </InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>
```

## Credit Memo

The following credit memo is for \$110, which includes tax charges.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/InvoiceDetail.dtd">
<cXML payloadID="CRD012042-120402" timestamp="2001-12-04T18:00:00-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>samsupplier@acme.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>bettybuyer@bigcompany.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>samsupplier@acme.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Invoicing App 2.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader invoiceID="CRD012042"
        purpose="creditMemo" operation="new"
        invoiceDate="2001-12-04T18:00:00-07:00">
        <InvoiceDetailHeaderIndicator isHeaderInvoice="yes"/>
        <InvoiceDetailLineIndicator/>
        <Comments xml:lang="en-US">
          Buyer ordered 5 Computer Video Cables. We invoiced the
          customer for 8 by mistake.
        </Comments>
      </InvoiceDetailRequestHeader>
      <InvoiceDetailHeaderOrder>
        <InvoiceDetailOrderInfo>
          <OrderReference>
            <DocumentReference
              payloadID="1002700953000.152865612.2314.120401002@ariba.com"/>
            </OrderReference>
          </InvoiceDetailOrderInfo>
          <InvoiceDetailOrderSummary invoiceLineNumber="2">
            <SubtotalAmount>
              <Money currency="USD">-100.00</Money>
            </SubtotalAmount>
          </InvoiceDetailOrderSummary>
        </InvoiceDetailHeaderOrder>
        <InvoiceDetailSummary>
          <SubtotalAmount>
            <Money currency="USD">-100.00</Money>
          </SubtotalAmount>
          <Tax>
            <Money currency="USD">-10</Money>
            <Description xml:lang="en">total tax</Description>
          </Tax>
          <NetAmount>
            <Money currency="USD">-110.00</Money>
          </NetAmount>
          <DueAmount>
            <Money currency="USD">-110.00</Money>
          </DueAmount>
        </InvoiceDetailSummary>
      </InvoiceDetailHeaderOrder>
    </InvoiceDetailRequest>
  </Request>
</cXML>
```

```

        </DueAmount>
      </InvoiceDetailSummary>
    </InvoiceDetailRequest>
  </Request>
</cXML>

```

## Debit Memo

The supplier realizes that it used incorrect math in the credit memo; it should have credited the buying organization \$66.

The following debit memo is for \$44 (\$110 - \$66).

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/InvoiceDetail.dtd">
<cXML payloadID="DBD012042-120402" timestamp="2001-12-04T18:50:00-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>samsupplier@acme.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>bettybuyer@bigcompany.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>samsupplier@acme.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Invoicing App 2.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader invoiceID="DBD012042"
        purpose="debitMemo" operation="new"
        invoiceDate="2001-12-04T18:50:00-07:00">
        <InvoiceDetailHeaderIndicator isHeaderInvoice="yes"/>
      </InvoiceDetailRequestHeader>
      <InvoiceDetailLineIndicator/>
      <Comments xml:lang="en-US">
        Bigcompany ordered five Computer Video Cables at 20.00 a piece.
        We invoiced the customer for eight by mistake. We sent out a
        credit memo for 100.00 instead of 60.00, so now we are issuing
        this debit memo in the amount of 2 X 20 = 40.00. The customer
        received all 5 items. Sorry for the confusion.
      </Comments>
    </InvoiceDetailRequestHeader>
    <InvoiceDetailHeaderOrder>
      <InvoiceDetailOrderInfo>
        <OrderReference>
          <DocumentReference
            payloadID="1002700953000.152865612.2314.120401002@ariba.com"/>
        </OrderReference>
      </InvoiceDetailOrderInfo>
      <InvoiceDetailOrderSummary invoiceLineNumber="2">
        <SubtotalAmount>
          <Money currency="USD">40.00</Money>
        </SubtotalAmount>
      </InvoiceDetailOrderSummary>
    </InvoiceDetailHeaderOrder>
  </Request>
</cXML>

```



```

</InvoiceDetailHeaderOrder>
<InvoiceDetailSummary>
  <SubtotalAmount>
    <Money currency="USD">40.00</Money>
  </SubtotalAmount>
  <Tax>
    <Money currency="USD">4.00</Money>
    <Description xml:lang="en">total tax</Description>
  </Tax>
  <NetAmount>
    <Money currency="USD">44.00</Money>
  </NetAmount>
  <DueAmount>
    <Money currency="USD">44.00</Money>
  </DueAmount>
</InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>

```

## Cancel Invoice

This example illustrates a cancel invoice.

As always, this document has a unique payloadID. InvoiceDetailRequestHeader element contains the attribute operation="delete" and the InvoiceID attribute reflects that this is a cancel invoice. There is also a DocumentReference element containing the payloadID of the original invoice, as required by the cXML DTDs.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/InvoiceDetail.dtd">
<cXML payloadID="INV123.DELETE" timestamp="2001-12-05T10:22:00-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>supplier@supplier.com </Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>buyer@buyer.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>supplier@supplier.com </Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Mega Invoicing App 1.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <InvoiceDetailRequest>
      <InvoiceDetailRequestHeader purpose="standard"
        operation="delete" invoiceID="INV123.DELETE"
        invoiceDate="2001-12-05T09:30:00-07:00">
        <InvoiceDetailHeaderIndicator></InvoiceDetailHeaderIndicator>
        <InvoiceDetailLineIndicator isTaxInLine="yes"
          isShippingInLine="yes" isAccountingInLine="yes">
        </InvoiceDetailLineIndicator>
        <InvoicePartner>

```

```

    <Contact role="billTo" addressID="Billing">
      <Name xml:lang="en">Test User</Name>
      <PostalAddress name="billing department">
        <DeliverTo>Test User</DeliverTo>
        <Street>16 Castro Street</Street>
        <City>Mountain View</City>
        <State>CA</State>
        <PostalCode>95035</PostalCode>
        <Country isoCountryCode="US">United States</Country>
      </PostalAddress>
      <Email name="default">testuser@company.com</Email>
      <Phone name="work">
        <TelephoneNumber>
          <CountryCode isoCountryCode="US">1</CountryCode>
          <AreaOrCityCode>215</AreaOrCityCode>
          <Number>9990000</Number>
        </TelephoneNumber>
      </Phone>
    </Contact>
    <IdReference identifier="11280630"
      domain="accountReceivableID">
      <Creator xml:lang="en">Test Supplier</Creator>
    </IdReference>
  </InvoicePartner>
  <DocumentReference payloadID="INV123">
  </DocumentReference>
</InvoiceDetailRequestHeader>
<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <OrderReference orderID="P0123">
      <DocumentReference payloadID="1002700953000.152865612">
      </DocumentReference>
    </OrderReference>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem invoiceLineNumber="1" quantity="10">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice>

```

---

## Scheduled Payments

---

Buying organizations use scheduled payments (PaymentProposalRequest documents) to tell Ariba Network and their suppliers about planned payments. These documents list scheduled payment dates, discount amounts, and net amounts. Ariba Buyer 8.2 and later supports them.

This section discusses the following topics:

- “For Information Only” on page 203
- “EFT Payment” on page 203
- “Discount Management” on page 203
- “Example Scheduled Payment” on page 204

### For Information Only

By default, scheduled payments are for information only. After approving invoices from suppliers, the procurement application sends scheduled payments to Ariba Network. Ariba Network forwards them to suppliers through email or displays them in their online Inboxes.

### EFT Payment

Buying organizations can use scheduled payments for automated payment through Electronic Funds Transfer (EFT) on Ariba Network. After the procurement application approves invoices from suppliers, it sends scheduled payments to Ariba Network, which forwards them to suppliers. On the scheduled payment date, Ariba Network instructs the buying organization’s bank to pay the supplier’s bank.

Scheduled payments for EFT payment contain the `isNetworkPayment` attribute:

```
<PaymentProposalRequest
  paymentDate="2005-01-14T14:34:45-07:00"
  operation="new"
  paymentProposalID="PAY123"
  isNetworkPayment="yes">
```

### Discount Management

Scheduled payments can be used for buyer-initiated discounts (formerly called Dynamic Discounting), supplier-initiated discounts, and standing early payment term offers, which allow buying organizations and suppliers to negotiate early settlement in return for discounts.

After a buying organization approves an invoice, their invoicing application sends a scheduled payment to Ariba Network, which displays it to suppliers. If the buying organization activates discount management, Ariba Network allows suppliers to request an accelerated payment, using a discount rate configured by the buying organization. If the supplier activates discount management, Ariba Network allows the buying organization to review and either accept or reject the early settlement terms. If necessary, the supplier can revise the offer.

Scheduled payments for discount management contain a special Extrinsic element named `immediatepay`:

```
<PaymentProposalRequest
  paymentDate="2005-04-20T23:59:20-07:00"
  operation="new"
  paymentProposalID="PAY123">
  ...
  <Extrinsic name="immediatepay">yes</Extrinsic>
</PaymentProposalRequest>
```

If a supplier selects a scheduled payment for early settlement, Ariba Network sends a `CopyRequest` document containing the attached scheduled payment back to the buying organization with an updated net amount and payment date. This `PaymentProposalRequest` document contains the attribute `operation="update"`.

The following example shows the `earlyPaymentDiscount` extrinsic. This amount is used to offset the early payment discount included in the discount portion of the payment proposal to make sure that double-dipping does not occur. The value indicates the actual discount amount.

```
<PaymentProposalRequest
  paymentDate="2005-04-20T23:59:20-07:00"
  operation="new"
  paymentProposalID="PAY123">
  ...
  <Extrinsic name="earlyPaymentDiscount"><Money currency='USD'>100</Money></Extrinsic>
</PaymentProposalRequest>
```

## Example Scheduled Payment

The following scheduled payment is for an ACH payment. It instructs Ariba Network to present the payment to suppliers for discount management.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.014/PaymentRemittance.dtd">
<cXML payloadID="123@bigbuyer.com" timestamp="2005-04-20T23:59:45-07:00">
  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>joe@bigbuyer.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="AribaNetworkUserId">
        <Identity>judy@workchairs.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>joe@bigbuyer.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Super Procurement Application 1.2</UserAgent>
    </Sender>
  </Header>
  <Request>
    <PaymentProposalRequest
      paymentProposalID="proposal123"
      operation="new"
      paymentDate="2005-05-20T23:59:20-07:00">
      <Extrinsic name="immediatepay">yes</Extrinsic>
      <PayableInfo>
```

```

    <PayableInvoiceInfo>
      <InvoiceReference invoiceID="ABC">
        <DocumentReference payloadID="25510.10.81.231"/>
      </InvoiceReference>
      <PayableOrderInfo>
        <OrderReference orderID="DEF">
          <DocumentReference payloadID="25510.10.81.002"/>
        </OrderReference>
      </PayableOrderInfo>
    </PayableInvoiceInfo>
  </PayableInfo>
  <PaymentMethod type="ach"/>
  <Contact role="remitTo" addressID="Billing">
    <Name xml:lang="en">Lisa Dollar</Name>
    <PostalAddress name="billing department">
      <DeliverTo>Lisa Dollar</DeliverTo>
      <Street>100 Castro Street</Street>
      <City>Mountain View</City>
      <State>CA</State>
      <PostalCode>95035</PostalCode>
      <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
    <Email name="default">ldollar@workchairs.com</Email>
    <Phone name="work">
      <TelephoneNumber>
        <CountryCode isoCountryCode="US">1</CountryCode>
        <AreaOrCityCode>650</AreaOrCityCode>
        <Number>9990000</Number>
      </TelephoneNumber>
    </Phone>
  </Contact>
  <GrossAmount>
    <Money currency="USD">3000.00</Money>
  </GrossAmount>
  <DiscountAmount>
    <Money currency="USD">160.00</Money>
  </DiscountAmount>
  <AdjustmentAmount>
    <Money currency="USD">30.00</Money>
  </AdjustmentAmount>
  <NetAmount>
    <Money currency="USD">2810.00</Money>
  </NetAmount>
</PaymentProposalRequest>
</Request>
</cXML>

```

## End Points Integration with Ariba Network Adapters

The cXML Adapter supports end points integration for the following solutions:

- Ariba Network Adapter for Oracle® Fusion Middleware
- Ariba Network Adapter for SAP NetWeaver®
- Ariba® Network Transport Adapter
- Ariba Procure-to-Pay™
- Ariba Procure-to-Order™

The Ariba Network Adapters allow data exchange between your external ERP system and Ariba Network. If you have configured an end point for your external ERP system in Ariba Network, then you must maintain the end point information in the channel configuration file of the respective Ariba Network Adapter.

**Note:** An end point is applicable only to *outbound* documents that are sent from your ERP system to Ariba Network.

When your Ariba Network Adapter receives an outbound document from your external ERP system, the network adapter appends the end point information to the cXML and then posts the document to Ariba Network. For example, when creating the cXML header for a purchase order, your network adapter checks if the end point information is maintained in the channel configuration file.

If you create a purchase order using an Ariba procurement solution and send that to a supplier through Ariba Network, the end point information is appended to the cXML and the document is then posted to the Ariba Network. Ariba checks to see if the end point information is maintained as part of your site profile.

**Note:** If you configure an end point for your Ariba procurement solution in Ariba Network, be sure to work with your Ariba Customer Support representative to maintain the end point information as part of your organization's Site Profile.

The network adapter or procurement solution includes the end point in the cXML header as part of the **From** element as explained below:

```
<Header>
  <From>
    <Credential domain="NetworkID">
      <Identity>AN02000076701</Identity>
    </Credential>
    <Credential domain="SystemID">
      <Identity>ARB2</Identity>
    </Credential>
    <Credential domain="EndPointID">
      <Identity>ERPEndPoint2</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="NetworkID">
      <Identity>AN02000076720</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkID">
      <Identity>AN02000076701</Identity>
      <SharedSecret>welcome1a</SharedSecret>
    </Credential>
    <UserAgent>Buyer</UserAgent>
  </Sender>
```

```
</Header>
<Request>
<OrderRequest>.....
```

**ERPEndPoint1** is the value that you have configured in the communication channel in SAP NetWeaver or channel configuration file.

**Note:** In a single ERP configuration, you do not need to include in the System ID in the <From> credential.

## Enabling End Point Integration with Ariba Network Adapters

A new configuration parameter called **endpoint** is added to the channel configuration file to capture the end point information that you configured in Ariba Network for your external ERP system. This parameter is optional and must be set only if you have configured an end point for your external ERP system in Ariba Network.

**Note:** The following procedure is applicable only to the Ariba Network Adapter for Oracle Fusion Middleware and Ariba Network Transport Adapter.

### ▼ To enable end point integration for outbound documents:

- 1 Open the Configuration Channel file for the respective adapter.
- 2 Specify values for the relevant parameters.
- 3 Save the Configuration Channel file.

### ▼ To enable end point integration for Ariba Network Adapter for SAP NetWeaver:

In the **Model Configurator**, open the communication channel relevant to the outbound transaction you are configuring. Under **Communication Channel**, double-click **ERPEndPoint1** is the value that you have configured in the communication channel in SAP NetWeaver or channel configuration file.

**Note:** In a single ERP configuration, you do not need to include in the System ID in the <From> credential.

## End Points Integration in a Multiple ERP Setup

If your account has been enabled for multiple external ERPs and end point support in Ariba Network, then you must create system IDs corresponding to the different ERP systems, as well as an end point for each system ID. For documents that are sent from Ariba Network to the ERP, the network adapter uses the system ID to identify the ERP systems that receive those documents. Therefore, it is sufficient to create only one end point with the same name for all system IDs.

When creating the cXML header for a purchase order, your network adapter checks if the end point information is maintained in the channel configuration file or communication channel in SAP NetWeaver. If it is, then the network adapter includes the end point in the cXML header along with the system ID as illustrated below:

```
<Header>
  <From>
    <Credential domain="NetworkId">
      <Identity>AN02000076701</Identity>
    </Credential>
    <Credential domain="SystemId">
      <Identity>ARB2</Identity>
    </Credential>
    <Credential domain="EndPointID">
      <Identity>ERPEndPoint1</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="NetworkID">
      <Identity>AN02000076720</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkID">
      <Identity>AN02000076701</Identity>
      <SharedSecret>welcome1a</SharedSecret>
    </Credential>
    <UserAgent>Buyer</UserAgent>
  </Sender>
</Header>
<Request>
  <OrderRequest>.....
```

**ERPEndPoint1** is the value that you have configured in the communication channel in SAP NetWeaver or channel configuration file and **ARB2** is the system ID.



---

## Chapter 10 Receipts

- “[ReceiptRequest Routing](#)” on page 209
- “[cXML ReceiptRequest Document](#)” on page 209
- “[Example Receipt](#)” on page 214

### ReceiptRequest Routing

---

Receipts indicate that your organization received ordered products or services. Your shipping clerks or requisitioners create them and send them to Ariba Invoice Professional through Ariba Network. Accounts payable personnel use them (in addition to purchase orders and invoices) to determine when and how much to pay suppliers.

Your organization generates receipts and addresses them to suppliers. However, Ariba Network routes them to Ariba Invoice Professional, not to suppliers. The To credential in the document header specifies the supplier, but Ariba Network does not send them to suppliers.

Ariba Network routes receipts immediately to Ariba Invoice Professional and sends copies to the **Receipts** page in your online Outbox tab. It does not match receipts to corresponding documents, so you can send them for expired purchase orders or master agreements, or for purchase orders that were not routed through Ariba Network.

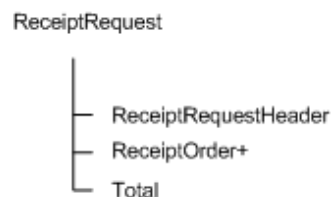
In Ariba Invoice Professional, users use the receipts (if required) and purchase orders to ensure the invoices they receive from suppliers are correct. See “[Receiving Types in Ariba Invoice Professional](#)” on page 123 for information configuring receipt requirements for order items

### cXML ReceiptRequest Document

---

The cXML ReceiptRequest document represents receipts. It defines the receipt information of a purchase order or master agreement with item details. A receipt line is an `ReceiptItem` element with a receipt line number.

This document has the following high-level structure:



The plus (+) indicates an element that can repeat.

Each ReceiptRequest document represents details about a receipt against a purchase order or a master agreement sent from a buying organization to a supplier. You can use it for any portion of all or selected line items from single or multiple purchase orders.

This document is defined in the cXML Private.dtd, not in the public DTDs.

## ReceiptRequestHeader

The ReceiptRequestHeader element contains header information for this receipt.

This element has the following attributes:

receiptID (required)	A buyer-generated unique identifier for this receipt.
receiptDate (required)	The date and time the items or services were received by the buying organization. This date and time should be earlier than the document's timestamp.
operation	The operation described by this receipt document. Must be "new".

### Comments

Any comments for the receipt.

### Extrinsic

Additional information related to this receipt. This information should not duplicate any information in the ReceiptRequest document.

## ReceiptOrder

The ReceiptOrder element defines information related to a purchase order or master agreement.

This element has the following attribute:

closeForReceiving	Flag that indicated that the underlying order or master agreement needs should be closed for further receiving on approval of this receipt. It is false (no), by default.  If this receipt is against a purchase order or release order, this flag indicates that the corresponding order should be closed for receiving. If this receipt is against a no-release master agreement, this flag indicates that the master agreement should be closed for receiving.
-------------------	---

### ReceiptOrderInfo

The ReceiptOrderInfo element contains the reference information of the purchase order or master agreement. The various content options are, in order of preference: OrderReference, MasterAgreementReference, MasterAgreementIDInfo, or OrderIDInfo.

### OrderReference

A reference to the purchase order containing the items or services being received.

**MasterAgreementReference**

A reference to the master agreement containing the items or services being received.

**OrderIDInfo**

The buyer ID of the corresponding purchase order.

**MasterAgreementIDInfo**

The buyer ID of the corresponding master agreement.

**ReceiptItem**

The `ReceiptItem` element defines a receipt line item using information from the purchase order or master agreement. If this receipt is against a release order, specify both the release order and the master agreement. If the receipt is against a no-release master agreement, specify only the master agreement. If the receipt is against a purchase order, specify the purchase order.

This element has the following attributes:

<code>receiptLineNumber</code> (required)	A buyer-defined ID for the current line item. It must be unique across all receipt line items of the same <code>ReceiptRequest</code> document.
<code>quantity</code> (required)	<p>The quantity received for the current receipt line.</p> <p>Negative values in the <code>quantity</code> attribute indicate corrective action against an order that was received or returned with errors. For example, if there is an order that has one line item with the quantity as 20, and you originally specified the received quantity as 20; later found out the received quantity was only 15 and not 20, you can do a negative receiving to correct this mistake.</p> <pre>&lt;ReceiptItem receiptLineNumber="1" quantity="-5" type="received"&gt;</pre>
<code>type</code>	<p>To indicate if the buyer has received or returned the items from the supplier. This attribute can have the following values:</p> <ul style="list-style-type: none"> <li><b>received:</b> Indicates the goods have been received by the buyer. For example: <pre>&lt;ReceiptItem receiptLineNumber="1" quantity="20" type="received"&gt;</pre></li> <li><b>returned:</b> Indicates the goods have been returned by the buyer. For example: <pre>&lt;ReceiptItem receiptLineNumber="1" quantity="10" type="returned"&gt;</pre></li> </ul>

**ReceiptItemReference**

The references related to this line item.

`ReceiptItemReference` has the following attribute:

<code>lineNumber</code> (required)	Line number of the line item, copied from the <code>OrderRequest</code>
---------------------------------------	---

**ItemID**

The supplier part number of current line item, copied from the OrderRequest.

**Description**

The line item description, copied from the OrderRequest.

**ManufacturerPartID**

The manufacturer part number.

**ManufacturerName**

The name of the manufacturer.

**ShipNoticeReference**

Reference to the supplier's ShipNoticeRequest document.

**ShipNoticeIDInfo**

ID of the ShipNoticeRequest. This ID is used when the ShipNoticeReference element is omitted.

**UnitRate**

The amount to be paid per unit of specified measure.

**ReceivedAmount**

Money amount of goods or services received for the receipt line item. The total received amount must equal to quantity x UnitRate.

**AssetInfo**

The AssetInfo element contains optional asset data for each quantity of each receipt line item.

This element has the following attributes:

tagNumber	Internal tag number for this asset
serialNumber	Manufactures serial number for this asset
location	Location of this asset

**Comments**

Textual comments for this line item.

**Total**

Summary total amount of all receipt line item amounts.

**Money**

Specifies the total money amount represented by the receipt.

## Example Receipt

The following receipt is for two computer monitors. It closes the corresponding purchase order for further receiving.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://svcdev.ariba.com/schemas/cXML/1.2.015/Private.dtd">
<cXML xml:lang="en-US" payloadID="xyz-u44pdate@buyer.com" timestamp="2005-11-13T23:00:00-08:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN13000001260</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN12000002259</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN13000001260</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Receiving Application, V1.2</UserAgent>
    </Sender>
  </Header>
  <Request>
    <ReceiptRequest>
      <ReceiptRequestHeader receiptID="RC1234"
        receiptDate="2005-11-13T22:00:00-08:00"
        operation="new"/>
      <ReceiptOrder closeForReceiving="yes">
        <ReceiptOrderInfo>
          <OrderReference orderID="D019">
            <DocumentReference payloadID="112.334.44.=90"/>
          </OrderReference>
        </ReceiptOrderInfo>
        <ReceiptItem receiptLineNumber="1" quantity="2" type="received">
          <ReceiptItemReference lineNumber="1">
            <ItemID>
              <SupplierPartID>RCA15</SupplierPartID>
            </ItemID>
            <Description xml:lang="en">Computer Monitor</Description>
            <ManufacturerPartID>718177215</ManufacturerPartID>
            <ManufacturerName>RCA</ManufacturerName>
          </ReceiptItemReference>
          <UnitRate>
            <Money currency="USD">150.50</Money>
            <UnitOfMeasure>EA</UnitOfMeasure>
          </UnitRate>
          <ReceivedAmount>
            <Money currency="USD">301.00</Money>
          </ReceivedAmount>
          <AssetInfo serialNumber="SER20201" tagNumber="tag000005"
            location="Sunnyvale"/>
          <AssetInfo serialNumber="SER20202" tagNumber="tag000006"
            location="Sunnyvale"/>
        </ReceiptItem>
      </ReceiptOrder>
    </Request>
  </Total>
  <Money currency="USD">301.00</Money>
</Total>
```

```
    </ReceiptRequest>  
  </Request>  
</cXML>
```





---

## Chapter 11 Catalog Upload Transaction

- “Introduction to Catalog Upload” on page 217
- “Sending a CatalogUploadRequest Document” on page 219
- “Receiving the Response” on page 222
- “Receiving Later Catalog Status” on page 223

---

### Introduction to Catalog Upload

The cXML Catalog Upload transaction enables suppliers to programmatically upload and publish catalogs on Ariba Network. The transaction gives you an alternative to logging in to Ariba Network 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.

Ariba Network supports any catalog in CIF or cXML format. A CIF catalog can be either a text file saved with a .cif extension or an Excel file saved with a .xls extension. If you use the Catalog Upload transaction, the catalog cannot exceed 10 MB.

Ariba Network allows only suppliers, not buying organizations, to upload catalogs.

Before uploading an Excel file as your catalog, be sure the file does not exceed 1 MB. Compress any Excel file that exceeds that limit before trying to upload it.

### Transaction Overview

The Catalog Upload transaction consists of two cXML documents:

cXML Document	Purpose
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 Ariba Network to acknowledge the receipt of a CatalogUploadRequest.

These documents use the standard cXML transport mechanism to travel between your cXML-enabled catalog-management application and Ariba Network:

- 1 Your catalog-management application opens an HTTPS connection to Ariba Network and perform a POST to send the CatalogUploadRequest document.
- 2 Ariba Network receives the document, validates it, and parses it.
- 3 After verifying your credential, Ariba sends a Response document through the same HTTPS connection.
- 4 Your application receives the Response document and closes the HTTPS connection.

You can receive further status regarding the catalog through email or cXML.

## Prerequisites

Before using the Catalog Upload transaction, ensure you meet the following prerequisites.

### Ariba Network Account

You must have a valid Ariba Network supplier account, and it must be fully enabled, not a preview account. In addition, you must have account administrator privileges to change your account settings, such as cXML authentication method.

### cXML-Enabled Application

You must have a cXML-enabled application that can initiate HTTPS POST operations.

cXML-enabled applications must support sending and receiving the ProfileRequest transaction. This transaction is important for the Catalog Upload transaction because:

- You will use it to find out URL to which to post the CatalogUploadRequest document.
- Ariba Network will use it to find out the URL to which to post StatusUpdateRequest documents to you for later catalog-status notification.

### cXML-Documents Authentication

You must have configured your Ariba Network account and your cXML-enabled application to authenticate received cXML documents. For more information, see “[cXML Document Authentication](#)” on page 27.

### Valid Catalogs

You must have valid catalogs. Catalogs can be in CIF 2.1, CIF 3.0, cXML, or Excel format.

To debug your catalogs as you develop them, use your Ariba Network account interactively to upload them. Ariba Network automatically validates them and flags any syntactic errors. Your catalogs will be validated when you use the Catalog Upload transaction, but solving catalog problems is easier in interactive sessions. Interactive sessions also allow you to use the order tester to exercise catalogs and generate test purchase orders.

Compress (zip) large catalogs before uploading; for more information, see “[Compressing Catalogs](#)” on page 222.

The following information is available in the Product Documentation > For Users section of the Learning Center..

For information about...	See...
Creating CIF and cXML catalogs	<i>Ariba Catalog Format Reference</i>
Creating Excel format catalogs	<i>Creating and Managing Catalogs for Ariba Network Suppliers</i>
Uploading, validating, and testing catalogs	<i>Creating and Managing Catalogs for Ariba Network Suppliers</i>

## Sending a CatalogUploadRequest Document

The following example shows a Catalog Upload document. It contains two documents enclosed in a MIME envelope: a CatalogUploadRequest document and a CIF 3.0 catalog.

```
--kdf1kajfdksadjfklsadjfk1jdfdsfdkf
Content-type: text/xml; charset=UTF-8
Content-ID: <part0.PC028.975529413484@saturn.workchairs.com>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.023/cXML.dtd">
<cXML timestamp="2011-12-28T16:56:03-08:00"
payloadID="123456669138--123456789955556789@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>MyHomemadeCatalogManager</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.PC028.975529413154@saturn.workchairs.com</URL>
      </Attachment>
      <Commodities>
        <CommodityCode>52</CommodityCode>
      </Commodities>
      <AutoPublish enabled="true"/>
      <Notification>
        <Email>judy@workchairs.com</Email>
        <URLPost enabled="true"/>
      </Notification>
    </CatalogUploadRequest>
  </Request>
</cXML>

--kdf1kajfdksadjfklsadjfk1jdfdsfdkf
Content-type: text/plain; charset=US-ASCII
Content-Disposition: attachment; filename=PremiereCatalog.cif
Content-ID: <part2.PC028.975529413154@saturn.workchairs.com>
Content-length: 364

CIF_I_V3.0
LOADMODE: F
CODEFORMAT: UNSPSC
CURRENCY: USD
SUPPLIERID_DOMAIN: DUNS
```

```

ITEMCOUNT: 3
TIMESTAMP: 2006-01-15 15:25:04
DATA
942888710,34A11,C11,"Eames Chair",11116767,400.00,EA,3,"Fast MFG",,,400.00
942888710,56A12,C12,"Eames Ottoman",11116767,100.00,EA,3,"Fast MFG",,,100.00
942888710,78A13,C13,"Folding Chair",11116767,25.95,EA,3,"Fast MFG",,,25.95
ENDOFDATA

--kdf1kajfdksadjfk1asdjfk1jdfdsfdkf--

```

## To Element

The To element can specify either Ariba Network or the buying organization.

If you address the document to Ariba Network, Ariba Network performs generic catalog validation, not buyer-specific catalog validation. If you address the document to a buying organization, Ariba Network performs validation against the customer's catalog rules.

## CatalogUploadRequest Element

CatalogUploadRequest has an Operation attribute that specifies the type of upload to perform:

- **new**—Uploads a new catalog. A catalog with the same name must not exist.
- **update**—Overwrites an existing catalog. A catalog with the same name must exist.

CatalogUploadRequest contains the following elements.

### CatalogName Element

CatalogName specifies the name of the uploaded catalog. This value is the user-visible name, not the file name of the catalog.

CatalogName has an `xml:lang` attribute that specifies the language used for the catalog name. Language codes are defined in the XML 1.0 Specification (at [www.w3.org/TR/1998/REC-xml-19980210.html](http://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. 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.

### Description

Description briefly describes the catalog contents. Buying organizations can search and view this information.

Description has an `xml:lang` attribute that specifies the language of the description text.

### Attachment Element

Attachment specifies the URL of the attached catalog. It contains one URL element with the scheme "cid".

For more information about attachments, see "[Attaching Your Catalog](#)" on page 222.

### Commodities Element

Commodities specifies the top-level commodity codes for the items in your catalog. Buying organizations use these codes to search for new catalogs. It contains one or more `CommodityCode` elements.

Use **two-digit** UNSPSC (United Nations Standard Products and Services Code) segment codes. For more information, see Appendix A, “[Recommended Coding Systems](#).”

### AutoPublish Element

AutoPublish automatically publishes the catalog to buying organizations after upload. You can automatically publish only if the following requirements are met:

- A previous version of the catalog exists in your account and you are performing an update operation.
- The previous version is in the “published” state. It must have been published private (with a list of buying organizations) or public.
- The `CatalogUploadRequest` (the `To` element) is addressed to one of your customers, not Ariba Network.

AutoPublish has an `Enabled` attribute that 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 Element

Notification sends catalog-status notifications through email or cXML POST. For examples of these messages, see “[Receiving Later Catalog Status](#)” on page 223.

Notification contains either one `Email` element or one `URLPost` element, or both elements.

### Email Element

Email specifies the mailbox to which Ariba Network emails status messages. You can use only one `Email` element, and it can contain only one email address.

### URLPost Element

URLPost specifies whether Ariba Network sends catalog status messages as cXML `StatusUpdateRequest` documents.

The URL destination of the `StatusUpdateRequest` is determined by your Website’s response to Ariba Network’s `ProfileRequest` transaction. For more information about `ProfileRequest`, see Chapter 4, “[Profile Transaction](#).”

URLPost has an `Enabled` attribute that specifies whether Ariba 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 Multipurpose Internet Mail Extensions (MIME) 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 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:”. Do not use any encoding for the catalog, except the appropriate character set encoding (for example, UTF-8). Do not base64 encode the catalog.

For more information about attachments in cXML, see the discussion of the `Attachment` element in the *cXML User's Guide*.

### Compressing Catalogs

Before attaching catalogs larger than 3 MB, compress them with a Zip utility, such as WinZip.

Catalog files must have a `.xml`, `.cif`, `.xls`, or `.zip` file extension.

---

## Receiving the Response

After you send a `CatalogUploadRequest`, Ariba Network replies with a standard cXML Response document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.024/cXML.dtd">
<cXML payloadID="980306507433-6714998277961341012@10.10.83.39"
timestamp="2011-01-23T19:21:47-08:00">
  <Response>
    <Status code="201" text="Accepted">The catalog upload request is processing</Status>
  </Response>
</cXML>
```

The following table lists possible status codes:

Status Code	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 (email 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 catalog exceeds the 10 MB limit.
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.
564 Upload Disabled	Catalog uploading is temporarily unavailable due to scheduled maintenance. Try again later.

For other possible status codes, see the *cXML User's Guide*.

## Receiving Later Catalog Status

If you include the `Notification` element to request later catalog-status notification, Ariba Network sends a message when the catalog reaches its final status. The possible final catalog states are:

State	Meaning
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).

This notification does not appear in your Ariba Network Inbox. You receive it only through email or URLPost.

## Email

The following example shows an email status notification:

From: AribaNetworkAdmin@ariba.com  
 To: judy@workchairs.com  
 Subject: Status of CatalogUploadRequest for Winter Prices from the Ariba Network

Catalog Name: Winter Prices  
 Status: Published  
 Payload ID: 123456669138--1234567899555556789@10.10.83.39

Please log in to your Ariba Network account to view or edit the catalog file.

Thank you,  
 Ariba Network Customer Service

## URLPost

The following example shows a StatusUpdateRequest notification sent by Ariba Network:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/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--1234567899555556789@10.10.83.39"></DocumentReference>
        <Status text="Success" code="200">
          Validated
        </Status>
      </StatusUpdateRequest>
    </Request>
  </cXML>
```



The possible status codes are:

<b>Status Code</b>	<b>Meaning</b>
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 12 Get Pending/Data Download Transaction

- “Overview of Polling and Downloading” on page 227
- “GetPendingRequest” on page 228
- “GetPendingResponse” on page 230
- “DataRequest” on page 231
- “DataResponse” on page 232

### Overview of Polling and Downloading

---

Ariba Buyer and procurement applications receive documents from Ariba Network by polling for them and downloading them. Suppliers that have integrated their Ariba Network accounts with their external ERP system using cXML can also receive documents from Ariba Network by polling and downloading. They use the cXML get pending transaction (GetPendingRequest/GetPendingResponse) to poll for waiting documents. They use the cXML data download transaction (DataRequest/DataResponse) to retrieve those documents.

Polling and downloading are the only ways Ariba Buyer, procurement, and supplier applications receive data from Ariba Network. Ariba Network does not transmit documents to procurement or supplier applications; instead it queues them, by document type, for downloading initiated by them.

**Note:** In Ariba Procure-to-Pay, Ariba Network sends the document to Ariba Procure-to-Pay by posting the document. By posting the document, customers can immediately view order confirmations, ship notices, invoices, and other documents sent by suppliers. Ariba Procure-to-Pay does not use the cXML get pending and download transactions

### Polling Frequency

Procurement and supplier applications should poll for new documents and download them at least once per hour. They can poll for StatusUpdateRequest documents more often than once per hour for the current status of purchase orders.

### Polling Time

For better response from Ariba Network, procurement and supplier applications should schedule polling and downloading for a random offset from the top of the hour. For example, if polling and downloading once per hour, choose a random time from 0 to 59 minutes past the hour (ideally, randomize the seconds, too). Polling and downloading can occur at the same time every hour, but avoid times such as zero or thirty minutes past the hour. This technique spaces downloading so all buying and supplier organizations do not request downloads at the same time.

## Download All Waiting Documents

The get pending transaction has a `maxMessages` attribute that allows procurement and supplier applications to limit the number of documents returned by Ariba Network. Use this attribute to prevent Ariba Network, procurement, or supplier applications from timing out during downloading.

For example, if there are 99 waiting documents, the procurement or supplier application can download them in batches of 20 (`maxMessages="20"`):

Poll Iteration	Documents Returned
get pending/data download 1	20
get pending/data download 2	20
get pending/data download 3	20
get pending/data download 4	20
get pending/data download 5	19
get pending 6	0

Every time a procurement or supplier application polls and downloads, it should completely empty the pending document queue on Ariba Network. It must initiate multiple get pending and download transactions until Ariba Network indicates that there are no documents waiting. It should not stop polling and downloading after a predetermined number of get pending and download transactions.

## Confirm the Receipt of Documents

After downloading `ConfirmationRequest`, `ShipNoticeRequest`, or `InvoiceDetailRequest` documents, procurement applications should send `StatusUpdateRequest` documents to Ariba Network to indicate that it received them. Supplier applications must also send the `StatusUpdateRequest` documents to Ariba Network after receiving the documents. If required, the `Extrinsic` element can be used for additional information about the status of the document being updated.

Procurement applications should not send a `StatusUpdateRequest` document to confirm a `StatusUpdateRequest` document, because that action could lead to circular confirmations.

## GetPendingRequest

---

Procurement and supplier applications use the cXML `GetPendingRequest` document to poll Ariba Network for waiting documents. Generate a `GetPendingRequest` document for each type of cXML document that the procurement or supplier application can download:

- `StatusUpdateRequest`
- `ConfirmationRequest`
- `ShipNoticeRequest`
- `InvoiceDetailRequest`
- `SupplierChangeMessage`
- `SalesOrderRequest`
- `CopyRequest.PaymentProposalRequest`
- `CopyRequest.PaymentRemittanceRequest`

- CopyRequest.PaymentRemittanceStatusUpdateRequest
- SubscriptionChangeMessage
- OrganizationChangeMessage

Ariba Network suppliers must configure their Ariba Network accounts to retrieve cXML documents through the pending queue using the GetPending/Data Download transaction method. For more information, see the *Configuring Document Routing* topic in the Product Documentation > For Administrators section of the Learning Center.

The following example polls for cXML StatusUpdateRequest documents:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="1105574416.19583@hydra.buyer.com"
  timestamp="2005-01-13T00:00:16+00:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN13000000259</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN13000000259</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Procurement System 3.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <GetPendingRequest lastReceivedTimestamp="2005-01-12T00:00:25+00:00"
      maxMessages="20">
      <MessageType>StatusUpdateRequest</MessageType>
    </GetPendingRequest>
  </Request>
</cXML>
```

The `lastReceivedTimestamp` attribute specifies the oldest message to be returned by Ariba Network. Use the cXML `timestamp` attribute of the last received `GetPendingResponse` document as the `lastReceivedTimestamp` attribute in the next `GetPendingRequest` document for that document type. When Ariba Network receives a `GetPendingRequest`, it discards waiting messages referenced in previous `GetPendingResponse` documents with timestamps equal to or earlier than the specified `lastReceivedTimestamp`. For more information about how to use this attribute, see [“lastReceivedTimeStamp and Waiting Documents”](#) on page 233.

The `maxMessages` attribute specifies the maximum number of documents to return. This attribute helps prevent Ariba Network, procurement, or supplier applications from timing out if there are many waiting documents. Use a value between 10 and 100 for this attribute. Procurement and supplier applications should continue using the get pending/data download transactions until the waiting document queue on Ariba Network empties.

If your organization is enabled for multiple ERP systems, you can include a `SystemID` that equates to one of your ERP accounts. Ariba Network searches for and returns all documents for the specified ERP system. If a `SystemID` is not specified, Ariba Network returns documents for all of your ERP accounts.

The following example polls for cXML documents for a single ERP account:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="1105574416.19583@hydra.buyer.com"
  timestamp="2005-01-13T00:00:16+00:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN13000000259</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="SystemID">
        <Identity>ERP01</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN13000000259</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Procurement System 3.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <GetPendingRequest lastReceivedTimestamp="2005-01-12T00:00:25+00:00"
      maxMessages="20">
      <MessageType>StatusUpdateRequest</MessageType>
    </GetPendingRequest>
  </Request>
</cXML>
```

## GetPendingResponse

Ariba Network returns a Response document that either contains or does not contain a GetPendingResponse document in the same HTTP connection. If there is no GetPendingResponse, no documents are waiting. If there is a GetPendingResponse, there are documents waiting.

The following example indicates that one or more documents are waiting:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML timestamp="2005-01-12T16:00:18-08:00"
  payloadID="1105574420906--451266344000288275@10.10.13.125">
  <Response>
    <Status code="200" text="OK"/>
    <GetPendingResponse>
      <cXML timestamp="2005-01-12T16:00:18-08:00"
        payloadID="1105574420141-977399960268715709@10.10.13.125">
        <Header>
          <From>
            <Credential domain="NetworkID">
              <Identity>AN01000000001</Identity>
            </Credential>
          </From>
          <To>
            <Credential domain="NetworkID">
              <Identity>AN13000000259</Identity>
            </Credential>
          </To>
        </Header>
      </cXML>
    </GetPendingResponse>
  </Response>
</cXML>
```

```

    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
      </Credential>
      <UserAgent>ANCXMLDispatcher</UserAgent>
    </Sender>
  </Header>
  <Message>
    <DataAvailableMessage>
      <InternalID domain="PendingMessages">3738</InternalID>
    </DataAvailableMessage>
  </Message>
</cXML>
</GetPendingResponse>
</Response>
</cXML>

```

The DataAvailableMessage element contains an internal ID, which corresponds to one or more documents waiting for download.

## DataRequest

After procurement and supplier applications obtain a DataAvailableMessage, they use its internal ID value to download the waiting documents by sending a cXML DataRequest document.

**Note:** Ariba Network's URL for receiving DataRequest documents is different than the URL for GetPendingRequest documents; the only way to obtain it is by issuing a ProfileRequest document. For more information, see Chapter 4, “[Profile Transaction](#).”

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="1105574421.19583@hydra.buyer.com"
  timestamp="2005-01-13T00:00:21+00:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN13000000259</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN13000000259</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Our Procurement System 3.0</UserAgent>
    </Sender>
  </Header>
  <Request>
    <DataRequest>
      <InternalID domain="PendingMessages">3738</InternalID>
    </DataRequest>
  </Request>
</cXML>

```

## DataResponse

Ariba Network responds to the cXML DataRequest with a DataResponse document and the requested documents together in a MIME envelope in the same HTTP connection. The Content-Type HTTP header defines the MIME boundary.

The following DataResponse document has one StatusUpdateRequest document attached.

Content-Type: multipart/mixed; boundary="-----\_Part\_0\_10550230.1105574425445"

-----\_Part\_0\_10550230.1105574425445

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

Content-ID: <1105574425572.1197583259@cetus.ariba.com>

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML timestamp="2005-01-12T16:00:25-08:00"
  payloadID="1105574425428-5167970095322563427@10.10.13.103">
  <Response>
    <Status code="200" text="OK"/>
    <DataResponse>
      <Attachment>
        <URL>cid:1105574422695.1816707419@cetus.ariba.com</URL>
      </Attachment>
    </DataResponse>
  </Response>
</cXML>
```

-----\_Part\_0\_10550230.1105574425445

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

Content-ID: <1105574422695.1816707419@cetus.ariba.com>

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="1105573919487--7116204576911739136@10.10.13.125"
  timestamp="2005-01-12T15:51:59-08:00">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN12000000259</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN13000000259</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN01000000001</Identity>
      </Credential>
      <UserAgent>AN</UserAgent>
    </Sender>
  </Header>
  <Request deploymentMode="production">
    <StatusUpdateRequest>
      <DocumentReference payloadID="D0123@hydra.buyer.com"/>
      <Status code="200" message="OK"/>
    </StatusUpdateRequest>
  </Request>
</cXML>
-----_Part_0_10550230.1105574425445--
```



## Invoice Attachments

If invoices have attachments, the DataResponse contains InvoiceDetailRequest attachments that also have attachments. Ariba Network sends invoices with attachments in multi-level MIME envelopes.

Downloading of invoice attachments is configurable in buyer Ariba Network accounts. Turning it on affects only newly-submitted invoices, not existing queued invoices.

## lastReceivedTimeStamp and Waiting Documents

Ariba Network deletes the referenced documents from the waiting documents queue when it receives the next GetPendingRequest for that document type. Procurement and supplier applications should not skip waiting documents. After sending a DataRequest document, they must receive the referenced documents before sending another GetPendingRequest document. If DataRequest documents fail, they should retry the transaction. If they fail again, they should notify the procurement and supplier application administrator.

Here is an example of how to use the lastReceivedTimeStamp attribute:

- 1 The procurement application sends a GetPendingRequest:

```
<GetPendingRequest lastReceivedTimeStamp="time1"/>
```

Ariba Network responds with

```
<XML timestamp="time2">
  <GetPendingResponse>
    <DataAvailableMessage>
      <InternalID domain="PendingMessages">idl</InternalID>
    </DataAvailableMessage>
  </GetPendingResponse>
</XML>
```

- 2 The procurement application retrieves the waiting documents by sending a DataRequest:

```
<DataRequest>
  <InternalID domain="PendingMessages">idl</InternalID>
</DataRequest>
```

Ariba Network responds with

```
<DataResponse>...</DataResponse>
```

At this point, the procurement application can download *idl* over and over again.

- 3 Later, the procurement application sends another GetPendingRequest:

```
<GetPendingRequest lastReceivedTimeStamp="time2"/>
```

The procurement application acknowledges receipt of the documents referenced by the first GetPendingResponse by sending the second GetPendingRequest. Ariba Network deletes all waiting documents associated with *idl* and the procurement application can no longer download *idl*.

Ariba Network responds with

```
<XML timestamp="time3">
  <GetPendingResponse>
    <DataAvailableMessage>
      <InternalID domain="PendingMessages">iid2</InternalID>
    </DataAvailableMessage>
  </GetPendingResponse>
</XML>
```

At this point, the procurement application can download *iid2* over and over again.

## **Authenticating Waiting Documents**

Procurement and supplier applications do not need to authenticate documents downloaded through the data download transaction; they come from a trusted source, Ariba Network. They can validate Ariba Network's SSL server certificate when they initiate HTTPS connections.

---

## Chapter 13 cXML Transformations on Ariba Network

- “About Transformations on Ariba Network” on page 235
- “Turning on Transformations” on page 235
- “Moving Extrinsic to the Contact Element” on page 236
- “Transforming ItemOut Extrinsic” on page 237
- “Consolidating ItemOut to OrderRequestHeader” on page 238
- “Mapping Custom Extrinsic to Ariba Network Extrinsic” on page 238
- “Mapping Extrinsic from cXML to EDI” on page 239

---

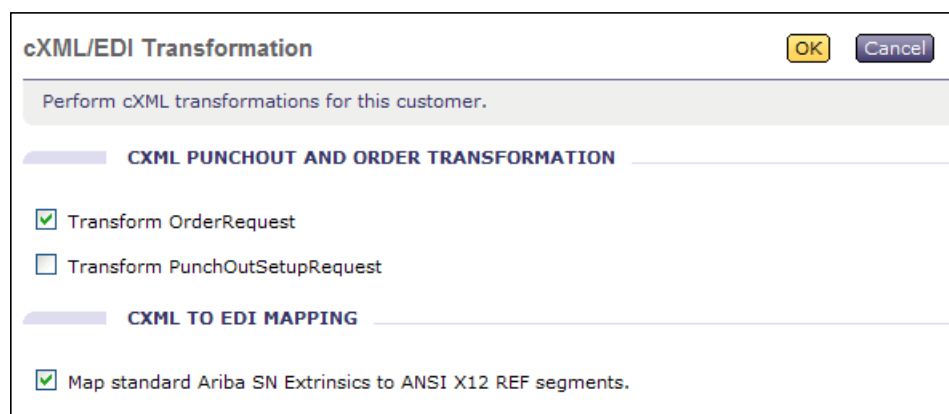
### About Transformations on Ariba Network

When Ariba Network routes cXML OrderRequest or PunchOutSetupRequest documents from buying organizations to suppliers, it can perform transformations that fix common semantic problems. These transformations are needed because Ariba Buyer installations might use extrinsics to specify commonly used data, such as addresses. To be most useful to suppliers, this data should appear in dedicated cXML elements, not extrinsic elements. Ariba Network can also change custom extrinsics to standard Ariba Network extrinsics, which enables suppliers to integrate with multiple customers more easily.

---

### Turning on Transformations

Suppliers can enable transformations on a per-customer basis by clicking **cXML/EDI Transformation** in the Customer Relationships area of their accounts.



The screenshot shows a dialog box titled "cXML/EDI Transformation" with "OK" and "Cancel" buttons in the top right corner. Below the title bar is a light gray instruction bar that says "Perform cXML transformations for this customer." The main content area is divided into two sections by horizontal lines. The first section is titled "CXML PUNCHOUT AND ORDER TRANSFORMATION" and contains two checkboxes: "Transform OrderRequest" (which is checked) and "Transform PunchOutSetupRequest" (which is unchecked). The second section is titled "CXML TO EDI MAPPING" and contains one checked checkbox: "Map standard Ariba SN Extrinsic to ANSI X12 REF segments."

Suppliers can turn on transformations for OrderRequest and PunchOutSetupRequest separately. Note that transformation controls are not available in the main Order Routing pages; they are only available in the cXML/EDI Transformation page. These transformations are described below.

## Moving Extrinsic to the Contact Element

If you activate OrderRequest transformations, Ariba Network moves contact information from a variety of extrinsics to the Contact element.

- 1 **Names** move from a variety of extrinsics to the Name element.

```
<Extrinsic name="Name"
<Extrinsic name="Requester"
<Extrinsic name="RequesterName"
<Extrinsic name="Requisition.Requester.Name">
<Extrinsic name="BuyerName"
<Extrinsic name="OriginalRequester"
```

Ariba Network moves the above data to:

```
<Contact><Name>
```

- 2 **Addresses** move from extrinsics to the PostalAddress element.

```
<Extrinsic name="Street"
<Extrinsic name="City"
<Extrinsic name="State"
<Extrinsic name="Country"
<Extrinsic name="isoCountryCode"
```

Ariba Network moves the above data to:

```
<Contact>
  <PostalAddress>
    <Street>
    <City>
    <State>
    <Country isoCountryCode>
```

A multiline Street extrinsic converts to a multiline Street element; it does not transform into separate Street elements for each line.

- 3 **Deliver To** information moves from a variety of extrinsics to the DeliverTo element within PostalAddress.

```
<Extrinsic name="DeliverTo"
<Extrinsic name="MailStop"
<Extrinsic name="PoleNumber"
```

Ariba Network moves the above data to:

```
<Contact><PostalAddress><DeliverTo>...
```

- 4 **Phone number** moves from a variety of extrinsics to the Phone element.

```
<Extrinsic name="Phone"
<Extrinsic name="Buyer Phone"
<Extrinsic name="Requester Phone Number"
```

Ariba Network moves the above data to:

```
<Contact><Phone>...
```

- 5 **Fax number** moves from an extrinsic to the Fax element.

```
<Extrinsic name="Fax">
```

Ariba Network moves the above data to:

```
<Contact><Fax>...
```

- 6 Email address** is moved from an extrinsic to the Email element.

```
<Extrinsic name="EmailAddress">
```

Ariba Network moves the above data to:

```
<Contact><Email>...
```

## Transforming ItemOut Extrinsic

---

If you activate OrderRequest or PunchOutSetupRequest transformations, Ariba Network moves data from ItemOut Extrinsic elements into intrinsic elements.

- 1 Requisition ID** is moved from a variety of extrinsics to the requisitionID attribute.

```
<ItemOut>
  <Extrinsic name="ReqNumber">value</Extrinsic>
  <Extrinsic name="Requisition #">value</Extrinsic>
  <Extrinsic name="PR No.">value</Extrinsic>
  <Extrinsic name="Requisition.PR">value</Extrinsic>
  <Extrinsic name="No.">value</Extrinsic>
```

Ariba Network moves the above data to:

```
<ItemOut requisitionID="value">...
```

If there are multiple requisition ID extrinsics, Ariba Network transforms only the last one into requisitionID.

- 2 URL** is moved from an extrinsic to the URL element.

```
<Extrinsic name="URL">
```

Ariba Network moves the above data to:

```
<URL>...
```

- 3 Requested Ship Date** is moved from a variety of extrinsics to the requestedDeliveryDate attribute in the ItemOut element.

```
<ItemOut>
  <Extrinsic name="ETA">value</Extrinsic>
  <Extrinsic name="RequestedShipDate">value</Extrinsic>
  <Extrinsic name="RequestedDeliveryDate">value</Extrinsic>
```

Ariba Network moves the above data to:

```
<ItemOut requestedDeliveryDate="value">
```

---

## Consolidating ItemOut to OrderRequestHeader

---

If you activate OrderRequest transformations, Ariba Network reduces redundancy by consolidating duplicate line item information in the OrderRequestHeader element.

- 1 If there are identical **Extrinsic** elements within every ItemOut, Ariba Network deletes them and creates a single Extrinsic at the OrderRequestHeader level. If there is only one ItemOut, Ariba Network moves any Extrinsic elements within it to the OrderRequestHeader level.
- 2 If there are identical **Contact** elements within every ItemOut, Ariba Network deletes them and creates a single Contact at the OrderRequestHeader level.
- 3 If there are identical **requisitionID** attributes within every ItemOut, Ariba Network deletes them and creates a single requisitionID at the OrderRequestHeader level.

---

## Mapping Custom Extrinsics to Ariba Network Extrinsics

---

If you enable **Map customer specific extrinsics to standard Ariba Network Extrinsics**, Ariba Network changes custom extrinsics to standard Ariba Network extrinsics in purchase orders, which allow you to more easily integrate with multiple customers.

**Note:** This control is available only if buying organizations define maps on Ariba Network to change custom extrinsics to standard extrinsics. They must contact Ariba's Global Services Organization (GSO) to define these maps.

Ariba Network has a list of approximately 1,500 standard Ariba Network extrinsics that covers concepts such as "billing account," "mortgage ID," and "docket number." It can map buying organizations' custom extrinsics to these standard extrinsics as it routes purchase orders. See the Ariba Network documentation for the list of standard Ariba Network extrinsics.

For example, Ariba Network can change the custom extrinsic

```
<Extrinsic name="Loan Number">1234</Extrinsic>
```

to the standard Ariba Network extrinsic

```
<Extrinsic name="mortgageIdNumber">1234</Extrinsic>
```

By default, this control is off, so suppliers receive extrinsics exactly as buying organizations send them in purchase orders.

## Mapping Extrinsicics from cXML to EDI

---

If you enable **Map standard Ariba Network extrinsicics to ANSI X12 REF segments**, Ariba Network maps standard Ariba Network extrinsic data to X12 REF segments in purchase orders.

If buying organizations either use standard Ariba Network extrinsicics in purchase orders, or map custom extrinsicics to standard Ariba Network extrinsicics in purchase orders, Ariba Network converts those extrinsicics to equivalent X12 REF elements. See the Ariba Network documentation for the list of standard Ariba Network extrinsicics.

By default, this control is off, so Ariba Network maps extrinsic data to EDI comment elements in purchase orders.





---

## Appendix A Recommended Coding Systems

- “Commodity Codes” on page 241
- “Currency Codes” on page 242
- “Unit of Measure Codes” on page 242
- “Country Codes” on page 243
- “Language Codes” on page 243
- “Dialing Codes” on page 244

This appendix lists standard coding systems recommended for use with Ariba applications.

### Commodity Codes

---

The United Nations Standard Products and Services Code (UNSPSC) standard is the preferred commodity coding system. The UNSPSC is free and contains descriptions of more than 12,000 products and services. Its coding structure is hierarchical, which combines similar items into standardized groups. The finer the granularity of the item description, the more digits in the UNSPSC code, up to 8 digits.

Two organizations manage the UNSPSC:

- Electronic Commerce Code Management Association (ECCMA). For a list of codes, go to:  
[www.eccma.org/unspsc](http://www.eccma.org/unspsc)
- United Nations Development Programme (UNDP). For a list of codes, go to:  
[www.unspsc.org](http://www.unspsc.org)

USPSC codes use a period to separate every two digits (for example 44.12.21.04), but cXML requires codes to contain no punctuation (for example, 44122104).

Examples:

Code	Meaning
44122104	Paper clips
22101522	Track bulldozers
82111502	Manual writing services

## Currency Codes

---

Specify currency names with ISO 4217 three-letter (CodeA) currency codes. For a list of codes (in Microsoft Access database format), go to:

[www.unetrades.net](http://www.unetrades.net)

- 1 Click **Repositories and Codes**.
- 2 Click **Country & Currency Codes**.
- 3 Scroll down to “ISO Country and Currency codes by UNECE.”
- 4 Click **UN/CCCodes R##-## Database**.

Examples:

Code	Meaning
BRL	Brazilian Real
JPY	Japanese Yen
USD	United States Dollar

## Unit of Measure Codes

---

Specify how items are packaged with the United Nations Units of Measure (UNUOM) Common Code system. This standard is also known as:

- United Nations Center for the Facilitation of Procedure and Practices for Administration, Commerce, and Transport (UN/CEFACT) codes
- United Nations Trade Data Elements Directory (UNTDDED) common codes
- United Nations Economic Commission for Europe (UN/ECE) Trade Facilitation Recommendation 20 - Codes for Units of Measurement used in International Trade

For a list of codes, go to:

[www.unece.org/cefact/codesfortrade/rec20.xml](http://www.unece.org/cefact/codesfortrade/rec20.xml)

Examples:

Code	Meaning
AY	Assembly
BX	Box
DZN	Dozen

---

## Country Codes

---

Specify country and region names with ISO 3166-1 two-letter (Alpha-2) country codes. For a list of codes (in Microsoft Access database format), go to:

[www.unetrades.net](http://www.unetrades.net)

- 1 Click **Repositories and Codes**.
- 2 Click **Country & Currency Codes**.
- 3 Scroll down to “ISO Country and Currency codes by UNECE.”
- 4 Click **UN/CCCodes R##-## Database**.

Examples:

Code	Meaning
BR	Brazil
JP	Japan
US	United States

---

## Language Codes

---

Specify language names with Java locale codes. For a list of codes, go to:

[java.sun.com/j2se/1.4.2/docs/guide/intl/locale.doc.html](http://java.sun.com/j2se/1.4.2/docs/guide/intl/locale.doc.html)

Java uses an underbar ( \_ ) within the codes, but cXML uses a dash (-). Examples:

Code	Meaning
pt-BR	Brazilian Portuguese
ja-JP	Japanese
en-US	United States English

In cases where you do not need to specify the geographic region, you can leave off the country portion of the locale code. For example, English text that is the same for US English and UK English can be labeled with the shortened code “en”.

## Dialing Codes

---

Specify international dialing codes with International Telecommunication Union (ITU) Recommendation E.164 country codes. For a list of codes, go to:

[globaltelecom.org/codes.htm](http://globaltelecom.org/codes.htm)

Examples:

Code	Meaning
55	Brazil
81	Japan
1	United States

---

# Index

## A

- accelerated payments 203
- Accounting element 119
- AccountingSegment and Segment
  - conversion on Ariba Network 166
  - from Ariba Buyer 120
- ACH information 165
- acknowledgements
  - blanket purchase orders 140
  - purchase orders 139
- ad hoc (non-catalog) items 133
- AdditionalCost 117
- AdditionalDeduction 116
- Address
  - termsOfDelivery 115
- Address element 110
- address format 72–73
- addressID attribute 72, 167
  - Address element 110
- ad-hoc items 74
- aisle-level PunchOut 82
- allDetail attribute 145, 147
- alternate digital certificate. *See* backup digital certificate
- alternateAmount attribute 18
- alternateCurrency attribute 18
- AN-ID. *See* NetworkID
- ANSI X12 19
- AreaOrCityCode element 24
- Ariba Buyer 7.0.6 defect fixes 136
- Ariba Invoice and Payment Professional 209
- Ariba Network 13
  - client certificate 29
  - production URLs 53
  - test URLs 57
- Ariba Network Adapter 171
- Ariba Ready certification 69
- Ariba Supplier Consulting 62
- Ariba.availableAmount extrinsic 124
- Ariba.collaborationAllowed extrinsic 124
- Ariba.invoicingAllowed extrinsic 124
- AribaNetwork.LegacyOrders extrinsic 136
- AribaNetwork.UserId 33
- asset tags, on ship notices 156
- AttachmentOnline Extrinsic 134, 155, 176
- attachments
  - to invoices 155, 176
  - to order confirmations and ship notices 155
  - to purchase orders 134–135

- Australian tax invoices 169
- authentication 27, 65, 67
- AutoPublish 221

## B

- B2C 69
- backordered status
  - header level 145
  - line-item level 147
- backup digital certificate 29
- blanket purchase order
  - masking 124
- blanket purchase orders 109, 110
  - acknowledgements 140
  - invoices for 188–191
  - limitations 135
- BlanketItemDetail element 124–??
- branding, PunchOut site 63
- BrowserFormPost element 82
- BuyerCookie element 81, 86
- buyerVatID Extrinsic 174

## C

- CA bundle 26
- CA certificate 26
- calling codes. *See* telephone
- cancel orders 74, 129
- canceled requisitions 66
- carrier names 152
- CarrierIdentifier 112
- case sensitivity 35
- Catalog Information Format (CIF) 62, 64
- catalog tester 67
- Catalog Upload transaction 217–225
  - attaching catalogs 222
  - CatalogUploadRequest 217
- catalogs 62
  - maximum size of 217
  - public, private 63
  - PunchOut 74
  - PunchOut compared to static 59
  - static 74
- CC invoice quick enablement 50
- CC invoices 50
- CEFACT units of measure 19, 242
- Certificate Authorities 31

- certificates, client and server 30
- certifying, PunchOut sites 69
- certservice.ariba.com 29
- change orders 74, 130
- checkout process 66
- CIF catalogs 62, 64
  - maximum size of 217
- City element 111
- Classification element 78, 86, 89
  - classification codes, UNSPSC 241
- client certificates 30
- Comments
  - termsofdelivery 115
- commodity code version 89
- commodity codes 73
- common name (CN) 31
- composite 168
- configuration management systems 68
- confirmID attribute 145
- Contact element 72
- contact roles
  - invoices 165
  - purchase orders 131
- cookies 66–67
- copies of purchase orders 132
- CopyRequest
  - for discount management 204
  - for purchase orders 132
- Correspondent element 43
- CostCenter element 72
- country and region codes 243
- Country element 22, 113
- CountryCode element 23
- creating new suppliers 43
- Credential element 65, 79, 80
  - case sensitivity 35
  - credential 37
  - DigitalSignature element 36
  - domains 33
  - test accounts 36
- credit card information 66
- credit memo 199
- credits to buyers 167
- currency attribute
  - Money element 125
- currency code mapping 127
- cXML
  - authentication 27
  - changes 15
  - elements 72
  - Private.dtd 210
  - profile. *See* Profile Transaction
  - and PunchOut sites 59
  - use with PunchOut sites 62
  - versions used by application 14
- cXML catalogs 64
  - maximum size of 217
- cxml-base64 86

- cxml-urlencoded 86

## D

- DataAvailableMessage 231
- DataRequest 231
- DataResponse 232
- date and time format 24
- debit memo 200
- debit memos 157
- DeliverTo element 110, 111
- deliveryDate attribute 147
- deploying, PunchOut sites 68–69
- deployment guides, writing for PunchOut sites 71–75
- Description element 88
- designing, PunchOut sites 63–64
- developing, PunchOut sites 64–67
- dialing codes 244
- digital certificate 27
- digital signatures
  - as alternative to signed paper invoices 161
  - for invoices 161
- DigitalSignature element 36
- discount management 203
- Distinguished Encoding Rules (DER) format 32
- Distinguished Name 29
- Distribution element 119
- document authentication 27
- Document Type Definition (DTD) 62
- DocumentReference element 167, 181
- domain attribute 65
- domain name 31
- DUNS and NetworkID in credentials 35
- D-U-N-S number 33, 77
- duplicate documents 140
- Dynamic Discounting. *See* discount management

## E

- earlyPaymentDiscount extrinsic 204
- ECCMA 241
- EDI 19, 60
- EDI REF segments 239
- effectiveDate attribute 110
- electronic data interchange 19
- Electronic Funds Transfer (EFT). 203
- Email element 114
- encryption strength 31
- ERP systems 73
- eSignatures 161
- European Union
  - regulatory requirements, digital signatures 161
- Excel files
  - uploading 217
- expirationDate attribute 110
- external purchase orders 158
- extrinsic data 66–67

- 
- Extrinsic element 24, 81
    - moved to Contact element 236
    - moved to header 238
  - Extrinsic elements
    - Ariba.availableAmount 124
    - Ariba.collaborationAllowed 124
    - Ariba.invoicingAllowed 124
    - AribaNetwork.LegacyOrders 136
    - AttachmentOnline 134, 155, 176
    - buyerVatID 174
    - CostCenter 24, 81
    - earlyPaymentDiscount 204
    - for clickable links 136, 177
    - immediatepay 204
    - invoiceSourceDocument 164
    - invoiceSubmissionMethod 162, 163
    - paymentServiceName 135
    - supplierCommercialCredentials 174
    - supplierCommercialIdentifier Extrinsic 174
    - supplierVatID 174
    - taxExchangeRate 169
    - TaxInvoice 169
    - UniqueName 24, 81
    - User 24
    - UserEmail 24
  - extrinsic mapping 238, 239
  - F**
    - Fax element 113
    - field lengths, for invoices 178
    - From element 79
  - G**
    - GE vPayment 135
    - GetPendingRequest 228
    - GetPendingResponse 230
    - GMT. *See* UTC
  - H**
    - header-level credit memo 167
    - hideAmount 124
    - hideUnitPrice 124
    - HTTPS 64
    - hyperlinks
      - invoices 177
      - purchase orders 136
  - I**
    - ICS invoices 163
    - identifying remittance addresses 167
    - IdReference element 164
    - immediatepay Extrinsic 204
    - index catalogs
      - Level 2 PunchOut example 101–103
      - punchoutLevel attribute in 94
    - index file format, for Level 2 PunchOut 78
    - initiating, PunchOut session 62
    - inspection dates 168
    - integration manager, for PunchOut sites 62
    - International Telecommunication Union (ITU) dialing codes 244
    - Invoice and Payment Professional 122
    - invoice conversion 43, 163
    - invoice quick enablement 45
    - InvoiceDetailServiceItem 159, 168
    - InvoiceIDInfo element 167, 181
    - invoiceLineNumber, numbering 171
    - InvoicePartner element 164
    - invoices 73
      - ACH information 165
      - attachments 155, 176, 233
      - blanket purchase order 188–191
      - field lengths 178
      - source documents of 164
      - validation for VAT 172
      - wire information 165
    - invoiceSourceDocument extrinsic 164
    - invoiceSubmissionMethod 163
    - invoiceSubmissionMethod extrinsic 162
    - invoiceSubmissionMethod extrinsic 163
    - invoicing 157–202
    - invoicing business rules 159
    - IP address 31
    - isAdHoc flag 133
    - isInternalVersion attribute 130
    - isNetworkPayment
      - for EFT 203
    - ISO
      - country and region codes 243
      - currency codes 242
      - date and time format 24
    - isoCountryCode attribute
      - Country element 113
      - CountryCode element 23, 113
    - Issuer Distinguished Name 29
    - item substitutions 149
    - ItemDetail element 88
    - ItemID element 87
    - ItemOut element 118
    - ItemType 118
    - itemType 168
  - J**
    - Japanese inspection dates 168
    - Java locale codes 243
  - L**
    - language codes 243
-

lastReceivedTimestamp 229, 233  
LeadTime element 89  
legacy purchase orders 136  
lettercase 35  
Level 2 PunchOut 94–103  
    benefits of 94–95  
    catalog format 96  
    example index catalog 101–103  
    index file format 78, 102  
    requirements 97–98  
    typical user experience 95  
line item quantity tolerance 126  
    change purchase order 126  
line-item credit memo 167  
line-item extrinsics moved to header 238  
line-item limit of cXML documents 135, 177  
lineNumber attribute 146  
    defect fixed in Ariba Buyer 7.0.6 136  
    numbering 171  
links  
    invoices 177  
    purchase orders 136  
local currency 18, 175  
locale codes 243

**M**

ManufacturerPartID element 92  
masking blanket purchase order 124  
matching invoices with purchase orders 158–159  
maximum size  
    of attachments 135, 177  
    of catalogs 222  
    of cXML documents 135, 177  
maxMessages 229  
measurement units. *See* units of measure  
message flow 65–66  
MIME 232  
Modification 116  
    itemDetail 125  
ModificationDetail 117  
Modifications 116  
    confirmationRequest 145  
    ConfirmationStatus 147  
Money element 18, 125  
multiple credentials 34  
multiple purchase orders per ConfirmationRequest 144  
multiple purchase orders per ShipNoticeRequest 153

## N

Name element 110  
name="routing" attribute 45  
NetworkID 33, 35  
no release required BPO 124  
non-catalog (ad hoc) items 133  
non-catalog items 74  
no-release blanket purchase order invoice 188–191

noticeDate attribute 145  
Number element 24  
number format 17

## O

obsolete currency codes 127  
operation attribute 80, 145  
operationAllowed attribute 66, 85  
Option name=attachments and changes 57  
order confirmations 141–151  
Order Request document 66  
order status PunchOut 142  
order tracking 152  
orderDate attribute 109, 146  
orderId attribute 109, 146  
OrderIDInfo 186  
OrderIDInfo, for order matching 158  
OrderReference, for order matching 158  
OrderRequest  
    cancel orders 129  
    change orders 130  
    documents 66, 109–140  
    OrderRequestHeader element 109  
    response 138  
orderType attribute 109  
orderVersion attribute 130  
organic supplier growth 43–49

## P

package tracking 152  
parentInvoiceLineNumber 118, 168  
parentLineNumber 87, 118  
passwords, and authentication 67  
path routing 132  
payload ID  
    duplicate documents 140  
    used for order matching 158  
payment proposal quick enablement 48  
Payment Proposals 48  
PaymentProposalRequest documents 203  
paymentServiceName extrinsic  
    for vPayment 135  
PCard element 135  
PCards 75  
phone. *See* telephone  
polling 227  
    frequency 227  
    time 227  
PostalAddress element 21  
PostalCode element 22, 111  
preferredLang attribute 114  
Price-Based Quantity 89, 126  
prices 74  
PriceTolerance 127  
Privacy-Enhanced Mail (PEM) format 32  
private catalogs 63



- Private.dtd 210
- PrivateId 33
- product classification codes (UNSPSC) 241
- product-level PunchOut 82
- Profile transaction 53–57
  - ProfileRequest 55
  - ProfileResponse 56
  - Reset Profile button 54
- ProviderId 33
- province abbreviation 22
- public catalogs 63
- pull communication 227
- PunchOut 59
  - order status PunchOut 142
- PunchOut catalogs
  - compared to static catalogs 59
- PunchOut index catalogs 59, 63, 77
- PunchOut sites
  - address format 72–73
  - analyzing current and future site 60
  - Ariba Supplier Consulting and 62
  - branding 63
  - certification 69
  - checkout process 66
  - commodity codes supported 73
  - customer requirements for 63–64
  - cXML and 62
  - deploying 68–69
  - designing 63–64
  - developing 64–67
  - development considerations 59
  - initiating session 62
  - integration manager and 62
  - key participants 61–62
  - Level 2 94–103
  - logging in to 65
  - message flow 65–66
  - outsourcing development 61
  - planning 59–75
  - PunchOutOrderMessage 85
  - PunchOutSetupRequest document 79
  - PunchOutSetupResponse documents 84
  - retrofitting a website 69–70
  - selecting 65
  - for services 70–71
  - session timeout 103
  - setting up 77–107
  - steps for implementing 60–69
  - technical developer and 61
  - testing 67–68
  - transactions supported 74–75
  - URL for 92
  - writing deployment guide for 71–75
- punchoutLevel attribute 78, 94
- PunchOutOrderMessage document 66, 85
- PunchOutSetupRequest document 62, 65, 79
- PunchOutSetupResponse document 66, 84
- purchase order quick enablement 44

- purchase orders 73, 109–140
  - attachments 134–135
  - cancel orders 129
  - change orders 130
  - in Invoice and Payment Professional 122
  - legacy 136
  - Quick Enablement 44–45

## Q

- quantity attribute 146, 147
- QuantityTolerance 127
- queue 228, 233
- quick enablement 43–49
  - CC invoices 50
  - ICS invoices 45
  - payment proposals 48
- quotes, split 75

## R

- receipts 209–215
- receivePunchoutSetupRequest.asp 103
- Recommendation 20 UOM codes 242
- REF segments 239
- release BPO invoice 191
- release required BPO 124
- releaseRequired attribute 110
- remittance IDs 167
- replace attribute 145
- requestedDeliveryDate defect fixed in Ariba Buyer 7.0.6 137
- requisition approval 66
- requisitions
  - canceled 66
- Reset Profile button 54
- resolveXML.asp 103
- response to OrderRequest 138
- RFC 2047 134, 155, 176
- roles, contact
  - invoices 165
  - purchase orders 131
- routing attribute 45

## S

- scheduled payments 203
- Secure Sockets Layer (SSL) 26
- security 64
- Segment and AccountingSegment
  - conversion on Ariba Network 166
  - from Ariba Buyer 120
- SelectedItem element 72, 82, 94
- self-signed invoices, supplier 162
- Sender element 80
- serial numbers, on ship notices 156
- server certificates 30

- service invoices 159
- service line items 178
- service.ariba.com 29
- services, PunchOut sites for 70–71
- shared secret 27
- SharedSecret element 65
- shelf-level PunchOut 82
- ship notices 151–155
  - serial numbers and asset tags on 156
- shipment tracking 152
- shipmentDate attribute 147
- shipping tax 170, 171
- ShippingPaymentMethod 114
- ship-to address information 66
- shopping cart 66
- size limit
  - of attachments 135, 177
  - of catalogs 222
  - of cXML documents 135, 177
- source documents, of invoices 164
- special handling tax 170, 171
- SSPPriateID 33
- StartPage element 84
- StartPage URL 66
- State element 22, 111
- static catalogs. *See* catalogs
- Status element 84
- store-level PunchOut 82
- Street element 21, 111
- Subject Distinguished Name 29
- substituting items in orders 149
- supplier self-signed invoices 162
- supplierCommercialCredentials Extrinsic 174
- supplierCommercialIdentifier Extrinsic 174
- SupplierID element 77
- SupplierPartAuxiliaryID element 67, 74, 87
- SupplierPartID element 87
- SupplierSetup element 82
- supplierVatID Extrinsic 174
- supply date 175
- System ID 33, 55
- SystemID 229

## T

- T notation for test accounts 36
- tax invoices 169
- tax on shipping or special handling 170
- taxExchangeRate extrinsic element 169
- TCcXMLFormatDTime.asp 106
- technical developer, for PunchOut site 61
- telephone
  - telephone country codes 244
  - TelephoneNumber element 23, 113
- terminating PunchOut sessions 103
- TermsofDelivery 114
- TermsOfDeliveryCode 114
- test account IDs 36

- test accounts 67
- test URLs 57
- testing, PunchOut sites 67–68
- third-party suppliers 74
- time and date format 24
- timeout
  - order response 138
  - PunchOut sessions 103
- To element 79
- Tolerances 126
- Total element 86
- tracking shipments 152
- transactions supported 74–75
- TransportInformation 112
- TransportTerms 115
- trusted Certificate Authorities 31
- type attribute 145, 147

## U

- UN EDIFACT 19
- Unit of Measure encoding 64
- unit price 62
- United Nations
  - country codes 243
  - currency codes 242
  - UNSPSC 241
  - UNUOM 19, 242
- UnitPrice element 88
- units of measure (UOM) 242
- unknown status 147
- UNSPSC 78
- uploading catalogs. *See* Catalog Upload transaction
- URL element 77, 84, 115
- URLs
  - for Ariba Network's cXML services 55
  - for ProfileRequest 53
  - for PunchOut sites 92
  - for testing 57
- User element 72
- UserAgent element 80
- Using 50
- using SSO to view invoices 177
- UTC (Coordinated Universal Time) 24

## V

- VAT 172–176
  - Bill To requirement 172
  - buyer and supplier IDs 174
  - commercial registration ID requirement 174
  - line item description 172
  - local currency requirement 175
  - Remit To requirement 173
  - Ship From and Ship To requirement 173
  - supply date 175
  - validation of invoices 172

- zero-value entry 175
- VendorID 33
- VendorSiteID 33
- versions of cXML 14
- virtual credit cards 135
- vPayment 135

## **W**

- websites, retrofitting to support PunchOut 69–70
- wire information 165
- withholding tax 115, 172

## **X**

- X12 19
- X12 REF segments 239
- XML parsers 62
- XML understanding 62

## **Z**

- zero-value VAT entries 175
- zip codes 22
- zip compression of catalogs 222

